Archives of the Sphere Online Judge

classical problemset

Editors:

Nguyen Dinh Tu
u.swarnaprakash
Andrés Leonardo Rojas
Duarte
Blue Mary
Ivan Metelsky
Adrian Kosowski
Phenomenal
Adrian Kuegel
Spooky
Camilo Andrés Varela León
Paritosh Aggarwal
Jin Bin
Le Đôn Khue
Thanh-Vy Hua
Csaba Noszaly
Roman Sol
Analysis Mode.(Elspeth,
Wanderley Guimarães
Knight-Errant)
Jelani Nelson (Minilek)
Abhilash I
Michał Czuczman
Paul Draper
Daniel Gómez Didier
Nguyen Minh Hieu
Neal Wu
Qu Jun
Prasanna
Fabio Avellaneda
Piotr Łowiec
Hoang Hong Quan
Ngô Minh Đu+´c
Nguyen Van Quang Huy
Robin Nittka
Brian
Ahmed Aly
Adam Dzedzej
Nikola P Borisov
Ajay Somani
Diego Satoba
Pawel Gawrychowski
Rafal
Marco Gallotta
Matthew Reeder
Pavel Kuznetsov
Robert Gerbicz
Simon Gog
Race with time
Michał Małafiejski
Chen Xiaohong
Kashyap KBR
John Rizzo

Bobby Xiao
Mauro Persano
P.Kasthuri Rangan-
Dariusz Dereniowski
Nguyê^~n Hà
Sebastian Kanthak
Du+o+ng
Bartłomiej Kowalski
Rahul Garg
Ivan Alfonso
Neal Zane
Olamendy
Chinh Nguyen
Martin Bader
Pawel Dobrzycki
Piotr Piotrowski
Patryk Pomykalski
Tomasz Goluch
Rahul
Frank Rafael
Radu Grigore
Arteaga
?????
Alfonso2 Peterssen
[FameofLight]
Krzysztof Kluczek
Tomek Czajka
Jose Daniel Rdguez
Łukasz Kuszner
Abel Nieto
abhijith reddy d
Rodríguez
Tomasz Bok
Bogusław K. Osuch
Mir Wasi Ahmed
Gogu Marian
Luka Kalinovcic
Le Trong Dao
andrespérez
Muntasir Azam
Clemente
Khan
Tomasz Niedźwiecki
Nghia Nguyen
Andrés
Hoang
Mejía-Posada
Andres Galvis
Pripoae Toni
Reinier César
Mujica Hdez
Mark Gordon
Lukas Mai
Stephen Merriman
Yash
Robert Rychcicki
VOJ problem setters
?????
Konrad Piwakowski
Zhang Taizhi

Last updated: 2009-07-31 11:11:47

Preface

This electronic material contains a set of algorithmic problems, forming the archives of the Sphere Online Judge (http://www.spoj.pl/), classical problemset. The document can be accessed at the following URLs:

- in PostScript format: http://www.spoj.pl/problems/classical.ps
- in Portable Document Format: http://www.spoj.pl/problems/classical.pdf

These resources are constantly updated to synchronise with the ever-changing hypertext version of the problems, and to include newly added problems. If you have obtained this document from another source, it is strongly recommended that you should download the current version from one of the aforementioned URLs.

Enjoy problem-solving at the Sphere Online Judge!

Remarks concerning this document should be sent to the following e-mail address: contact@spoj.pl.

Table of Contents

1. Problem TEST (1. Life, the Universe, and Everything)
2. Problem PRIME1 (2. Prime Generator)
3. Problem SBSTR1 (3. Substring Check (Bug Funny))
4. Problem ONP (4. Transform the Expression)
5. Problem PALIN (5. The Next Palindrome)
6. Problem ARITH (6. Simple Arithmetics)
7. Problem BULK (7. The Bulk!)
8. Problem CMPLS (8. Complete the Sequence!)
9. Problem DIRVS (9. Direct Visibility)
10. Problem CMEXPR (10. Complicated Expressions)
11. Problem FCTRL (11. Factorial)
12. Problem MMIND (12. The Game of Master-Mind)
13. Problem HOTLINE (13. Hotline)
14. Problem IKEYB (14. I-Keyboard)
15. Problem SHPATH (15. The Shortest Path)
16. Problem TETRA (16. Sphere in a tetrahedron)
17. Problem CRYPTO1 (17. The Bytelandian Cryptographer (Act I))
18. Problem CRYPTO2 (18. The Bytelandian Cryptographer (Act II))
19. Problem CRYPTO3 (19. The Bytelandian Cryptographer (Act III))
20. Problem CRYPTO4 (20. The Bytelandian Cryptographer (Act IV))
21. Problem TRICENTR (22. Triangle From Centroid)
22. Problem PIR (23. Pyramids)
23. Problem FCTRL2 (24. Small factorials)
24. Problem POUR1 (25. Pouring water)
25. Problem BSHEEP (26. Build the Fence)
26. Problem SBANK (27. Sorting Bank Accounts)
27. Problem HMRO (28. Help the Military Recruitment Office!)
28. Problem HASHIT (29. Hash it!)
29. Problem BLINNET (30. Bytelandian Blingors Network)
30. Problem MUL (31. Fast Multiplication)
31. Problem NHAY (32. A Needle in the Haystack)
32. Problem TRIP (33. Trip)
33. Problem RUNAWAY (34. Run Away)
34. Problem EQBOX (35. Equipment Box)
35. Problem CODE1 (36. Secret Code)
36. Problem PROPKEY (37. The Proper Key)
37. Problem LABYR1 (38. Labyrinth)
38. Problem PIGBANK (39. Piggy-Bank)
39. Problem STONE (40. Lifting the Stone)
40. Problem WORDS1 (41. Play on Words)
41. Problem ADDREV (42. Adding Reversed Numbers)
42. Problem BOOKS1 (43. Copying Books)
43. Problem SCYPHER (44. Substitution Cipher)
44. Problem COMMEDIA (45. Commedia dell Arte)
45. Problem SCRAPER (47. Skyscraper Floors)

46. Problem BEADS (48. Glass Beads)
47. Problem HAREFOX (49. Hares and Foxes)
48. Problem INCARDS (50. Invitation Cards)
49. Problem TOUR (51. Fake tournament)
50. Problem JULKA (54. Julka)
51. Problem JASIEK (55. Jasiek)
52. Problem DYZIO (56. Dyzio)
53. Problem SUPPER (57. Supernumbers in a permutation)
54. Problem PICAD (58. Crime at Piccadily Circus)
55. Problem BIA (59. Bytelandian Information Agency)
56. Problem DANCE (60. The Gordian Dance)
57. Problem BRCKTS (61. Brackets)
58. Problem IMP (62. The Imp)
59. Problem SQRBR (63. Square Brackets)
60. Problem PERMUT1 (64. Permutations)
61. Problem BALL1 (65. Ball)
62. Problem CRSCNTRY (66. Cross-country)
63. Problem CUTOUT (67. Cutting out)
64. Problem EXPR1 (68. Expression)
65. Problem MOULDS (69. Moulds)
66. Problem RELATS1 (70. Relations)
67. Problem TREE1 (71. Tree)
68. Problem BAC (73. Bacterial)
69. Problem DIVSUM (74. Divisor Summation)
70. Problem EDIT1 (75. Editor)
71. Problem EDIT2 (76. Editor Inverse)
72. Problem BRICKS (77. New bricks disorder)
73. Problem MARBLES (78. Marbles)
74. Problem EASYPIE (82. Easy Problem)
75. Problem BUNDLE (83. Bundling)
76. Problem SHORTCUT (84. Shortcut)
77. Problem DICE1 (85. Dice Contest)
78. Problem RAIN1 (86. November Rain)
79. Problem FOOTBALL (87. Football)
80. Problem TREE2 (88. Which is Next)
81. Problem HANGLET (89. Hang or not to hang)
82. Problem MINIMAX (90. Minimizing maximizer)
83. Problem TWOSQRS (91. Two squares or not two squares)
84. Problem CUTSQRS (92. Cutting off Squares)
85. Problem MAYA (94. Numeral System of the Maya)
86. Problem STPAR (95. Street Parade)
87. Problem SHOP (96. Shopping)
88. Problem PARTY (97. Party Schedule)
89. Problem DFLOOR (98. Dance Floor)
90. Problem BUS (99. Bus)
91. Problem BABTWR (100. Tower of Babylon)
92. Problem FISHER (101. Fishmonger)

93. Problem LITEPIPE (102. GX Light Pipeline Inc)
94. Problem HIGH (104. Highways)
95. Problem ALICEBOB (105. Alice and Bob)
96. Problem BINSTIRL (106. Binary Stirling Numbers)
97. Problem MAYACAL (107. Calendar of the Maya)
98. Problem MORSE (108. Decoding Morse Sequences)
99. Problem EXCHNG (109. Exchanges)
100. Problem CISTFILL (110. Fill the Cisterns)
101. Problem SEGVIS (112. Horizontally Visible Segments)
102. Problem FAMILY (115. Family)
103. Problem INTERVAL (116. Intervals)
104. Problem RHOMBS (118. Rhombs)
105. Problem SERVERS (119. Servers)
106. Problem SOLIT (120. Solitaire)
107. Problem TTABLE (121. Timetable)
108. Problem STEVE (122. Voracious Steve)
109. Problem PAYING (123. Paying in Byteland)
110. Problem RENT (130. Rent your airplane and make money)
111. Problem SQDANCE (131. Square dance)
112. Problem HELPR2D2 (132. Help R2-D2!)
113. Problem PHONY (134. Phony Primes)
114. Problem MAWORK (135. Men at work)
115. Problem TRANS (136. Transformation)
116. Problem PARTIT (137. Partition)
117. Problem POSTERS (138. Election Posters)
118. Problem MAZE (139. The Long and Narrow Maze)
119. Problem LONER (140. The Loner)
120. Problem GLUE (142. Johnny and the Glue)
121. Problem ALIENS (145. Aliens)
122. Problem MULTIPLY (146. Fast Multiplication Again)
123. Problem TAUT (147. Tautology)
124. Problem MLAND (148. Land for Motorways)
125. Problem FSHEEP (149. Fencing in the Sheep)
126. Problem PLONK (150. Where to Drink the Plonk?)
127. Problem COURIER (151. The Courier)
128. Problem SCALES (153. Balancing the Stone)
129. Problem ROCK (154. Sweet and Sour Rock)
130. Problem PALSEC (160. Choosing a Palindromic Sequence)
131. Problem PAINTTMP (174. Paint templates)
132. Problem POLY1 (175. Polygon)
133. Problem SUM1SEQ (176. Sum of one-sequence)
134. Problem ABWORDS (177. AB-words)
135. Problem ROADNET (178. Road net)
136. Problem WORDEQ (179. Word equations)
137. Problem CONTPACK (180. How to pack containers)
138. Problem SCUBADIV (181. Scuba diver)
139. Problem WINDOW1 (182. Window)

140. Problem ASCIRC (183. Assembler circuits)
141. Problem ATMS (184. Automatic Teller Machines)
142. Problem CHASE1 (185. Chase)
143. Problem LITELANG (186. The lightest language)
144. Problem FLBRKLIN (187. Flat broken lines)
145. Problem RECTNG1 (188. Rectangles)
146. Problem MUSKET (196. Musketeers)
147. Problem EMPTY (199. Empty Cuboids)
148. Problem MONODIG (200. Monodigital Representations)
149. Problem POLYGAME (201. The Game of Polygons)
150. Problem ROCKETS (202. Rockets)
151. Problem POTHOLE (203. Potholers)
152. Problem SLEEP (204. Sleepwalker)
153. Problem ICERINK (205. Icerink)
154. Problem BITMAP (206. Bitmap)
155. Problem THREECOL (207. Three-coloring of binary trees)
156. Problem STORE (208. Store-keeper)
157. Problem MAP (209. The Map)
158. Problem ALTARS (210. The Altars)
159. Problem PRIMIT (211. Primitivus recurencis)
160. Problem WATER (212. Water among Cubes)
161. Problem PANIC (215. Panic in the Plazas)
162. Problem SOPARADE (217. Soldiers on Parade)
163. Problem PHRASES (220. Relevant Phrases of Annihilation)
164. Problem VONNY (224. Vonny and her dominos)
165. Problem JEWELS (226. Jewelry and Fashion)
166. Problem ORDERS (227. Ordering the Soldiers)
167. Problem SHAMAN (228. Shamans)
168. Problem SORTING (229. Sorting is easy)
169. Problem ZEBRA (231. The Zebra Crossing)
170. Problem HOLIDAY1 (234. Getting Rid of the Holidays (Act I))
171. Problem VFMUL (235. Very Fast Multiplication)
172. Problem ROMAN (236. Converting number formats)
173. Problem SUMITR (237. Sums in a Triangle)
174. Problem HOLIDAY2 (238. Getting Rid of the Holidays (Act II))
175. Problem BTOUR (239. Tour de Byteland)
176. Problem BLOCKS (241. Arranging the Blocks)
177. Problem STABLEMP (243. Stable Marriage Problem)
178. Problem SQRROOT (245. Square Root)
179. Problem CHOCOLA (247. Chocolate)
180. Problem CTAIN (260. Containers)
181. Problem TRIPART (261. Triangle Partitioning)
182. Problem CONNECT (262. Connections)
183. Problem PERIOD (263. Period)
184. Problem CORNET (264. Corporative Network)
185. Problem CAVE (272. Cave Exploration)
186. Problem WMELON (274. Johnny and the Watermelon Plantation)

187. Problem WATERWAY (275. The Water Ringroad)
188. Problem CTGAME (277. City Game)
189. Problem BICYCLE (278. Bicycle)
190. Problem INUMBER (279. Interesting number)
191. Problem LIFTS (280. Lifts)
192. Problem MUDDY (282. Muddy Fields)
193. Problem NAPTIME (283. Naptime)
194. Problem SCITIES (286. Selfish Cities)
195. Problem NETADMIN (287. Smart Network Administrator)
196. Problem PON (288. Prime or Not)
197. Problem POLYEQ (290. Polynomial Equations)
198. Problem CUBERT (291. Cube Root)
199. Problem ALIBB (292. Alibaba)
200. Problem OFBEAT (293. Officers on the Beat)
201. Problem TWORK (296. Teamwork is Crucial)
202. Problem AGGRCOW (297. Aggressive cows)
203. Problem CABLETV (300. Cable TV Network)
204. Problem BOOK (301. Booklets)
205. Problem CANTON (302. Count on Cantor)
206. Problem UCUBE (303. The Unstable Cube)
207. Problem RATTERN (309. The Room Pattern)
208. Problem PITPAIR (318. Pythagorean Legacy)
209. Problem WINDMILL (325. The Tall Windmills)
210. Problem PLATON (327. Platon and Socrates)
211. Problem BISHOPS (328. Bishops)
212. Problem CALLS (329. Calls)
213. Problem HARDQ (332. Hard Question)
214. Problem PHDISP (334. The Philosophical Dispute)
215. Problem EOPERA (336. Exchange Operations)
216. Problem SEQ (339. Recursive Sequence)
217. Problem POKER (344. Poker)
218. Problem MIXTURES (345. Mixtures)
219. Problem COINS (346. Bytelandian gold coins)
220. Problem EXPEDI (348. Expedition)
221. Problem AROUND (349. Around the world)
222. Problem LANDSCAP (350. Landscaping)
223. Problem HAN01 (351. Ha-noi!)
224. Problem ACT (359. Alpha Centauri Tennis)
225. Problem IGARB (362. Ignore the Garbage)
226. Problem LISA (364. Pocket Money)
227. Problem PHIDIAS (365. Phidias)
228. Problem FARMER (366. Farmer)
229. Problem EMPODIA (367. Empodia)
230. Problem CSTREET (368. Cobbled streets)
231. Problem MATH1 (369. Math I)
232. Problem ONEZERO (370. Ones and zeros)
233. Problem BENEFACT (372. The Benefactor)

234. Problem GREED (373. Greedy island)
235. Problem MATRIX (374. Count maximum matrices)
236. Problem QTREE (375. Query on a tree)
237. Problem ACS (376. A concrete simulation)
238. Problem TAXI (377. Taxi)
239. Problem PERMUT2 (379. Ambiguous Permutations)
240. Problem BINGO (380. Bullshit Bingo)
241. Problem CHICAGO (381. 106 miles to Chicago)
242. Problem DECORATE (382. Decorate the wall)
243. Problem EUROPEAN (383. European railroad tracks)
244. Problem FOOL (384. Any fool can do it)
245. Problem GAME (385. Game schedule required)
246. Problem HELP (386. Help the problem setter)
247. Problem TOURS (387. Travelling tours)
248. Problem MENU (388. Menu)
249. Problem HOSPITAL (389. Use of Hospital Facilities)
250. Problem BILLIARD (390. Billiard)
251. Problem RAILROAD (391. Railroads)
252. Problem SPIN (392. Spin)
253. Problem HEXAGON (393. Hexagon)
254. Problem ACODE (394. Alphacode)
255. Problem APRIME (395. Anti-prime Sequences)
256. Problem HITOMISS (396. Hit or Miss)
257. Problem CONDUIT (397. I Conduit)
258. Problem RPGAMES (398. Roll Playing Games)
259. Problem TRANK (399. Team Rankings)
260. Problem TOANDFRO (400. To and Fro)
261. Problem TRANSL (401. Translations)
262. Problem HIKE (402. Hike on a Graph)
263. Problem FRACTION (403. Sort fractions)
264. Problem SCANNER (404. Scanner)
265. Problem TCUTTER (405. Tin Cutter)
266. Problem LOGIC (406. Logic)
267. Problem RNUMBER (407. Random Number)
268. Problem JRIDE (408. Jill Rides Again)
269. Problem DELCOMM (409. DEL Command)
270. Problem VHUFFM (410. Variable Radix Huffman Encoding)
271. Problem NUMQDW (411. Number of quite different words)
272. Problem COVER (412. K-path cover)
273. Problem WPUZZLES (413. Word Puzzles)
274. Problem BONFIRE (414. Equatorial Bonfire)
275. Problem DIV15 (416. Divisibility by 15)
276. Problem LAZYPROG (417. The lazy programmer)
277. Problem NECKLACE (418. Necklace)
278. Problem TRANSP (419. Transposing is Fun)
279. Problem AROAD (421. Another Road Problem)
280. Problem TRANSP2 (422. Transposing is Even More Fun)

281. Problem ASSIGN (423. Assignments)
282. Problem HAJIME (425. Kill evil instantly)
283. Problem PARTPALI (428. Particular Palindromes)
284. Problem TCNUMFL (449. Simple Numbers with Fractions Conversion)
285. Problem CLTZ (515. Collatz)
286. Problem ZZPERM (518. Zig-Zag Permutation)
287. Problem DIV (526. Divisors)
288. Problem DIV2 (530. Divisors 2)
289. Problem INCR (598. Increasing Subsequences)
290. Problem QUEST4 (660. Dungeon of Death)
291. Problem QUEST5 (661. Nail Them)
292. Problem SUBS (665. String it out)
293. Problem VOCV (666. Con-Junctions)
294. Problem LSORT (676. Sorting is not easy)
295. Problem BROW (677. A place for the brewery)
296. Problem HANOI07 (681. Building the Tower)
297. Problem PAIRINT (682. Pairs of Integers)
298. Problem ASSIGN4 (684. Another Assignment Problem)
299. Problem SEQPAR (685. Partition the sequence)
300. Problem REPEATS (687. Repeats)
301. Problem SAM (688. Toy Cars)
302. Problem LWAR (693. Lethal Warfare)
303. Problem DISUBSTR (694. Distinct Substrings)
304. Problem UFAST (695. Unite Fast)
305. Problem LIAR (696. Liar Liar)
306. Problem MWORDS (697. Matrix Words)
307. Problem PLHOP (698. Plane Hopping)
308. Problem HKNAP (699. Huge Knap Sack)
309. Problem BPRED (700. Branch Prediction)
310. Problem EXPAND (702. Barn Expansion)
311. Problem SERVICE (703. Mobile Service)
312. Problem PSTRING (704. Remove The String)
313. Problem SUBST1 (705. New Distinct Substrings)
314. Problem TFSETS (707. Triple-Free Sets)
315. Problem NICEDAY (709. The day of the competitors)
316. Problem PRO (726. Promotion)
317. Problem MAXIMUS (729. Move your armies)
318. Problem IVAN (734. Ivan and his interesting game)
319. Problem MDST (735. Minimum Diameter Spanning Tree)
320. Problem TREE (738. Another Counting Problem)
321. Problem NEG2 (739. The Moronic Cowmpouter)
322. Problem TRT (740. Treats for the Cows)
323. Problem STEAD (741. Steady Cow Assignment)
324. Problem LPERMUT (744. Longest Permutation)
325. Problem TEM (757. Thermal Luminescence)
326. Problem CH3D (760. Convex Hull 3D)
327. Problem MIS (764. Delay-noise Analysis)

328. Problem ARCHPLG (780. The Archipelago)
329. Problem TRIOPT (827. Trigonometric optimization)
330. Problem OPTM (839. Optimal Marks)
331. Problem WM06 (850. Soccer Choreography)
332. Problem SWAPS (861. Counting inversions)
333. Problem DNA (866. DNA Translation)
334. Problem CUBES (867. Perfect Cubes)
335. Problem IMPORT (869. Galactic Import)
336. Problem BASE (870. Basically Speaking)
337. Problem SEQUENCE (871. Letter Sequence Analysis)
338. Problem MARKUP (872. Mark-up)
339. Problem TRANSMIT (898. Transmitters)
340. Problem WSCIPHER (899. Ws Cipher)
341. Problem SPLIT (900. Split Windows)
342. Problem INDEXGEN (901. Index Generation)
343. Problem HANGOVER (902. Hangover)
344. Problem DOUBLEVI (903. Double Vision)
345. Problem IMAGE (904. Image Perimeters)
346. Problem MATRIX2 (912. Submatrix of submatrix)
347. Problem QTREE2 (913. Query on a tree II)
348. Problem FTOUR (944. Free Tour)
349. Problem IM (962. Intergalactic Map)
350. Problem EN (964. Entrapment)
351. Problem PB (967. Parking Bay)
352. Problem BIRTHDAY (972. Birthday)
353. Problem MOBILE (987. Mobile)
354. Problem MATRIOSH (999. Generalized Matrioshkas)
355. Problem EQDIV (1000. Equidivisions)
356. Problem BROUL (1001. Babylonian Roulette)
357. Problem UJ (1002. Uncle Jack)
358. Problem QUILT (1003. Little Quilt)
359. Problem POLYCODE (1004. Polygon Encoder)
360. Problem AIBOHP (1021. Aibohphobia)
361. Problem ANGELS (1022. Angels and Devils)
362. Problem COMCB (1024. Complete Chess Boards)
363. Problem FASHION (1025. Fashion Shows)
364. Problem FAVDICE (1026. Favorite Dice)
365. Problem FPOLICE (1027. Fool the Police)
366. Problem HUBULLU (1028. Hubululu)
367. Problem MATSUM (1029. Matrix Summation)
368. Problem EIGHTS (1030. Triple Fat Ladies)
369. Problem UPSUB (1031. Up Subsequence)
370. Problem GSS1 (1043. Can you answer these queries I)
371. Problem CTRICK (1108. Card Trick)
372. Problem SUDOKU (1110. Sudoku)
373. Problem NSTEPS (1112. Number Steps)
374. Problem TOE1 (1161. Tic-Tac-Toe (I))

375. Problem TOE2 (1162. Tic-Tac-Toe (II))
376. Problem JAVAC (1163. Java vs C ++)
377. Problem DEADFR (1166. Dead Fraction)
378. Problem MINCOUNT (1167. Move To Invert)
379. Problem SORTBIT (1182. Sorted bit squence)
380. Problem PALACE (1183. Accomodate the palace)
381. Problem ORIGLIFE (1267. Origin of Life)
382. Problem CNEASY (1268. CN Tower (Easy))
383. Problem CNHARD (1269. CN Tower (Hard))
384. Problem PNTBYNUM (1270. Paint By Numbers)
385. Problem SUMFOUR (1296. 4 values whose sum is 0)
386. Problem PARTSUM (1325. Partial Sums)
387. Problem CHASE (1326. A Chase In WonderLand)
388. Problem KPMATRIX (1329. Matrix)
389. Problem KPMAZE (1335. Maze)
390. Problem CZ_PROB1 (1391. Summing to a Square Prime)
391. Problem EMP (1417. University Employees)
392. Problem CATM (1418. The Cats and the Mouse)
393. Problem NGM (1419. A Game with Numbers)
394. Problem GEOM (1420. Geometry and a Square)
395. Problem FIRM (1421. Goods)
396. Problem KPPOLY (1431. Projections Of A Polygon)
397. Problem KPSUM (1433. The Sum)
398. Problem KPEQU (1434. Equation)
399. Problem PT07X (1435. Vertex Cover)
400. Problem PT07Y (1436. Is it a tree)
401. Problem PT07Z (1437. Longest path in a tree)
402. Problem ARCTAN (1440. Use of Function Arctan)
403. Problem CLEVER (1441. The Clever Typist)
404. Problem CHAIN (1442. Strange Food Chain)
405. Problem DELCOMM2 (1444. DEL Command II)
406. Problem BRCKGAME (1447. A Game of Toy Bricks)
407. Problem COVER2 (1448. 3D Cover)
408. Problem SEQ1 (1451. 01 Sequence)
409. Problem CAKE (1452. Birthday Cake)
410. Problem OPTSUB (1453. Optimal Connected Subset)
411. Problem MEMDIS (1454. Memory Distribution)
412. Problem ANALYSER (1455. Program Analyser)
413. Problem BLUEEQ (1457. Help Blue Mary Please! (Act I))
414. Problem BLUEEQ2 (1458. Help Blue Mary Please! (Act II))
415. Problem AEROLITE (1459. The Secret of an Aerolite)
416. Problem GALAXY (1460. A Simple Calculator in the Galaxy)
417. Problem DRAGON (1461. Greedy Hydra)
418. Problem BARB (1462. Barbarians)
419. Problem ROBOT (1463. Robot Number M)
420. Problem EDIT3 (1464. Editor II)
421. Problem CHRIS (1465. On the Way to Find Chris)

422. Problem CASHIER (1466. Blue Mary Needs Help Again)
423. Problem RAIN2 (1468. Outside it is now raining)
424. Problem SEQ2 (1470. Another Sequence Problem)
425. Problem PRLGAME (1471. A Game of Pearls)
426. Problem TOMJERRY (1472. Tom and Jerry)
427. Problem LEMON (1473. Lemon Tree in the Moonlight)
428. Problem WORMS (1475. VII - Act IV)
429. Problem PROFIT (1476. Maximum Profit)
430. Problem PT07A (1477. Play with a Tree)
431. Problem PT07B (1478. The Easiest Problem)
432. Problem PT07C (1479. The GbAaY Kingdom)
433. Problem PT07D (1480. Let us count 1 2 3)
434. Problem PT07F (1482. A short vacation in Disneyland)
435. Problem PT07G (1483. Colorful Lights Party)
436. Problem PT07H (1484. Search in XML)
437. Problem PT07J (1487. Query on a tree III)
438. Problem PT07K (1488. Balloons of JiaJia)
439. Problem MOLE (1505. Whac-a-Mole)
440. Problem RSORTING (1526. Ranklist Sorting)
441. Problem BLUEEQ3 (1536. Help Blue Mary Please! (Act III))
442. Problem MKJUMPS (1538. Making Jumps)
443. Problem MOBILE2 (1552. Mobiles)
444. Problem BACKUP (1553. Backup Files)
445. Problem ZOO (1554. Zoo)
446. Problem GSS2 (1557. Can you answer these queries II)
447. Problem TREEOI14 (1644. Trees)
448. Problem AMATH (1671. Another Mathematical Problem)
449. Problem GIWED (1672. The Great Indian Wedding)
450. Problem AMBM (1673. Ambitious Manager)
451. Problem EXPLOSN (1674. The Explosion)
452. Problem FUSION (1675. Fusion Cube)
453. Problem GEN (1676. Text Generator)
454. Problem HALLOW (1677. Halloween treats)
455. Problem TREASURY (1678. Royal Treasury)
456. Problem CYLINDER (1681. Cylinder)
457. Problem EXPRESS (1683. Expressions)
458. Problem FREQUENT (1684. Frequent values)
459. Problem GROCERY (1685. Grocery store)
460. Problem LOGIC2 (1687. Logic II)
461. Problem EASYPROB (1688. A Very Easy Problem!)
462. Problem HARDP (1689. Hard Problem)
463. Problem COCONUTS (1693. Coconuts)
464. Problem GRC (1695. Grandpa's Rubik Cube)
465. Problem WIJGT (1696. Will Indiana Jones Get There)
466. Problem OFORTUNE (1697. Ohgas' Fortune)
467. Problem PLSEARCH (1698. Polygonal Line Search)
468. Problem NSYSTEM (1699. Numeral System)

469. Problem TRSTAGE (1700. Traveling by Stagecoach)
470. Problem EOWAMRT (1701. Earth Observation with a Mobile Robot Team)
471. Problem CLEANRBT (1702. Cleaning Robot)
472. Problem ACMAKER (1703. ACM (ACronymMaker))
473. Problem CDOWN (1704. Countdown)
474. Problem GAMEFIL (1705. The Game of Efil)
475. Problem QKP (1706. Queens, Knights and Pawns)
476. Problem RELINETS (1707. Reliable Nets)
477. Problem SQCOUNT (1708. Square Count)
478. Problem SWTHIN (1709. Swamp Things)
479. Problem TWENDS (1710. Two Ends)
480. Problem PRMLX (1712. Permalex)
481. Problem SCALE (1713. Funny scales)
482. Problem NCKLCE (1715. Another Necklace Problem)
483. Problem GSS3 (1716. Can you answer these queries III)
484. Problem RP (1722. Life, the Universe, and Everything II)
485. Problem BMJ (1723. Bee Maja)
486. Problem TRICOUNT (1724. Counting Triangles)
487. Problem IMPORT1 (1725. The Importance)
488. Problem EXCHANGE (1726. Exchange)
489. Problem CPRMT (1728. Common Permutation)
490. Problem TCOUNT2 (1730. Counting Triangles II)
491. Problem TCOUNT3 (1731. Counting Triangles III)
492. Problem EQU2 (1739. Yet Another Equation)
493. Problem GERRYMDR (1740. Gerrymandering)
494. Problem TETRIS3D (1741. Tetris 3D)
495. Problem POLEVAL (1744. Evaluate the polynomial)
496. Problem SEQPAR2 (1748. Sequence Partitioning II)
497. Problem DIVSUM2 (1754. Divisor Summation (Hard))
498. Problem NQUEEN (1771. Yet Another N-Queen Problem)
499. Problem DETER2 (1772. Find The Determinant II)
500. Problem ALL (1774. All Discs Considered)
501. Problem BOOLE (1775. Boolean Logic)
502. Problem DNALAB (1776. DNA Laboratory)
503. Problem ICAMPSEQ (1784. IOICamp Sequence)
504. Problem CODE (1785. Code)
505. Problem DANGER (1786. In Danger)
506. Problem ENCONDIN (1787. Run Length Encoding)
507. Problem FRACTAN (1788. Fractan)
508. Problem GREEDULM (1789. Huffman´s Greed)
509. Problem HEAPULM (1790. Binary Search Heap Construction)
510. Problem GEN2 (1793. Text Generater II)
511. Problem DRAGON2 (1794. Greedy Hydra II)
512. Problem CARD (1797. Cardsharper)
513. Problem ASSIST (1798. Assistance Required)
514. Problem BOTTOM (1799. The Bottom of a Graph)
515. Problem CONTEST (1800. Fixed Partition Contest Management)

516. Problem DRINK (1801. Drink, on Ice)
517. Problem EDGE (1802. Edge)
518. Problem FOLD (1803. Fold)
519. Problem GENETIC (1804. Genetic Code)
520. Problem HISTOGRA (1805. Largest Rectangle in a Histogram)
521. Problem ORZ (1810. Nuclear Plants)
522. Problem LCS (1811. Longest Common Substring)
523. Problem LCS2 (1812. Longest Common Substring II)
524. Problem WA (1815. Problems Collection (Volume X))
525. Problem FTOUR2 (1825. Free tour II)
526. Problem SUDOKU2 (1833. Sudoku)
527. Problem SETSTACK (1835. The SetStack Computer)
528. Problem PIE (1837. Pie)
529. Problem TICKET (1838. Ticket to Ride)
530. Problem BOOKCASE (1839. The Bookcase)
531. Problem PQUEUE (1840. Printer Queue)
532. Problem PPATH (1841. Prime Path)
533. Problem LINELAND (1842. Lineland Airport)
534. Problem LEONARDO (1843. Leonardo Notebook)
535. Problem MICEMAZE (1845. Mice and Maze)
536. Problem PFDEP (1846. Project File Dependencies)
537. Problem NOCHANGE (1847. No Change)
538. Problem MKWAVES (1865. Making Waves)
539. Problem MKPALS (1866. Making Pals)
540. Problem MKMONEY (1868. Making Money)
541. Problem MKMOOM (1869. Making Mountains Out Of Molehills)
542. Problem MKLABELS (1870. Making Labels)
543. Problem MKBUDGET (1871. Making A Budget)
544. Problem ACARGO (1873. Accumulate Cargo)
545. Problem BWHEELER (1874. Burrows Wheeler Precompression)
546. Problem COOLNUMS (1875. Cool Numbers)
547. Problem DRAGONCU (1876. Dragon Curves)
548. Problem EPURSE (1877. Enrich my purse)
549. Problem FCATTLE (1878. Farmers Cattle)
550. Problem GAMETIME (1879. Game Time)
551. Problem HANOICAL (1880. Hanoi Calls)
552. Problem ICODER (1881. Instruction Decoder)
553. Problem RECTANGL (1960. Rectangles)
554. Problem ROMANRDS (1961. Roman Roads)
555. Problem CIRCLES (1962. Circles)
556. Problem IMGPROJ (1963. Image Projections)
557. Problem MMCUT (1964. Tree cut)
558. Problem SETCOV (1965. Set Cover)
559. Problem SKIVALL (1966. Ski Valley)
560. Problem ACFRAC (1991. Another Continuous Fractions Problem)
561. Problem BOX (2000. Boxes (Hard))
562. Problem RNG (2002. Random Number Generator)

563. Problem MINUS (2005. Minus Operation)
564. Problem BALIFE (2006. Load Balancing)
565. Problem COUNT (2007. Another Very Easy Problem! WOW!!!)
566. Problem BACKPACK (2008. Dab of Backpack)
567. Problem CRYPTO (2009. Cryptography)
568. Problem ROLLBALL (2019. The Rolling Ball)
569. Problem PEBBMOV (2021. Moving Pebbles)
570. Problem TRUTHORL (2022. Truth Or Lie)
571. Problem ONEINSTR (2023. One Instruction Computer Simulator)
572. Problem YKH (2031. Please help You-Know-Who)
573. Problem TILING (2038. Rectangle Tiling)
574. Problem REMGAME (2047. Stone Removing Game)
575. Problem CERC07B (2050. Strange Billboard)
576. Problem CERC07C (2051. Cell Phone)
577. Problem CERC07H (2052. Hexagonal Parcels)
578. Problem CERC07K (2053. Key Task)
579. Problem CERC07L (2054. Gates of Logic)
580. Problem CERC07N (2055. Weird Numbers)
581. Problem CERC07P (2056. Rectangular Polygon)
582. Problem CERC07R (2058. Reaux! Sham! Beaux!)
583. Problem CERC07S (2059. Robotic Sort)
584. Problem CERC07W (2060. Tough Water Level)
585. Problem MINDIST (2070. Minimum Distance)
586. Problem CANDY (2123. Candy I)
587. Problem FCTRL4 (2124. Last Non-Zero Digit of Factorials)
588. Problem LABYR2 (2125. Number Labyrinth)
589. Problem PANEL (2126. Panel)
590. Problem RAIN3 (2127. Rain)
591. Problem KROW (2128. K-In-A-Row)
592. Problem CAKE2 (2129. Cake)
593. Problem TROLLS (2130. Trolls)
594. Problem GETBACK (2131. Get Back!)
595. Problem PUZZLE2 (2132. Puzzle)
596. Problem CANDY2 (2136. Candy II)
597. Problem PIB (2138. Pibonacci)
598. Problem GOSSIPER (2139. Gossipers)
599. Problem FAIRONOT (2140. (un)Fair Play)
600. Problem GARDEN (2141. Golden Garden)
601. Problem DEPEND (2143. Dependency Problems)
602. Problem ROOT (2147. Root of a Linear Equation)
603. Problem CANDY3 (2148. Candy III)
604. Problem BAISED (2149. Biased Standings)
605. Problem SUBSEQ (2150. Counting Subsequences)
606. Problem CALCULAT (2151. Digital Calculator)
607. Problem FRACTAL (2152. Hilbert Curve)
608. Problem IMATCH (2153. Internet is Faulty)
609. Problem KRUSKAL (2154. Kruskal)

610. Problem ABSYS (2157. Anti-Blot System)
611. Problem CAKE3 (2159. Delicious Cake)
612. Problem HERE (2160. Here-There)
613. Problem JPIX (2161. Pixel Shuffle)
614. Problem TOWER (2162. Towers of Powers)
615. Problem AMCODES (2171. Ambiguous Codes)
616. Problem EMOTICON (2175. Emoticons)
617. Problem MUSIC (2185. Musical Optimization)
618. Problem MKPAIRS (2189. Making Pairs)
619. Problem TAN1 (2202. Tan and His Interesting Game)
620. Problem BALLOON (2270. Balloons in a Box)
621. Problem UCODES (2271. Undecodable Codes)
622. Problem DESERT (2272. Crossing the Desert)
623. Problem FERRY (2273. Ferries)
624. Problem ISLHOP (2274. Island Hopping)
625. Problem OIL (2275. Toil for Oil)
626. Problem RECTNG2 (2276. Partitions)
627. Problem SSORT (2277. Silly Sort)
628. Problem LEXBRAC (2317. Bracket Sequence)
629. Problem WORDS (2318. Overlapping Words)
630. Problem BIGSEQ (2319. Sequence)
631. Problem DISTANCE (2320. Manhattan)
632. Problem SEGMENTS (2321. Segments)
633. Problem TREEGAME (2322. Tree Game)
634. Problem COMPASS (2323. Broken Compass)
635. Problem MARIOGAM (2324. Mario)
636. Problem STRDIST (2325. String Distance)
637. Problem LIS2 (2371. Another Longest Increasing Subsequence Problem)
638. Problem ARRANGE (2412. Arranging Amplifiers)
639. Problem BUILD (2413. Building Beacons)
640. Problem CCOST (2414. Calculate The Cost)
641. Problem RESIST (2415. Kirchhof Law)
642. Problem DSUBSEQ (2416. Distinct Subsequences)
643. Problem ENEMY (2417. Eliminate The Enemies)
644. Problem FFROG (2418. Flying Frogs)
645. Problem GLGRID (2419. G-Line Grid)
646. Problem HHAND (2420. Hospital at Hands)
647. Problem ININT (2421. Incrementing The Integer)
648. Problem JAZZYJOB (2422. Jazzy Job)
649. Problem MINTRIAN (2423. Minimal Triangulations of Graphs)
650. Problem PLD (2426. Palindromes)
651. Problem RABBIT1 (2450. Counting Rabbits)
652. Problem PHONELIN (2485. Phone Lines)
653. Problem MAGIC4 (2511. Magic Program IV)
654. Problem GNY07A (2523. Mispelling)
655. Problem GNY07B (2524. Conversions)
656. Problem GNY07C (2525. Encoding)

657. Problem GNY07D (2526. Decoding)
658. Problem GNY07E (2527. Flipping Burned Pancakes)
659. Problem GNY07F (2528. Monkey Vines)
660. Problem GNY07G (2529. Model Rocket Height)
661. Problem GNY07H (2530. Tiling a Grid With Dominoes)
662. Problem GNY07I (2531. Spatial Concepts Test)
663. Problem PERMUT3 (2565. Another Permutation Problem)
664. Problem CLK (2631. Chomp)
665. Problem SC1 (2643. Starcraft I)
666. Problem KPARCH (2648. Archiver)
667. Problem KPSORT (2649. Weird sorting)
668. Problem WAR (2658. Art of War)
669. Problem EXAMPLE (2660. Example)
670. Problem ILLUM (2661. Illumination)
671. Problem PUTIN (2662. Put a Point in a Hyperspace)
672. Problem QTREE4 (2666. Query on a tree IV)
673. Problem POLYSSQ (2668. Polygon)
674. Problem MSTS (2670. Count Minimum Spanning Trees)
675. Problem SPP (2699. Recursive Sequence (Version II))
676. Problem UNTITLED (2709. Untitled Problem)
677. Problem COWCAR (2714. Cow Cars)
678. Problem GLASNICI (2715. Glasnici)
679. Problem QUADAREA (2716. Maximal Quadrilateral Area)
680. Problem ARMY (2727. Army Strength)
681. Problem BREAK (2728. Breaking in)
682. Problem INVENT (2731. Inventing Test Data)
683. Problem KEQ (2733. K Equal Digits)
684. Problem LARGE (2734. Large party)
685. Problem RAIL (2735. Simplify the Railroad System)
686. Problem PRHYME (2737. Perfect Rhyme)
687. Problem SUMSUMS (2742. Summing Sums)
688. Problem PRETILE (2743. Prefix Tiling)
689. Problem INCSEQ (2815. Increasing Subsequences)
690. Problem CSUBSEQS (2816. Common Subsequences)
691. Problem INCDSEQ (2817. Distinct Increasing Subsequences)
692. Problem RRSCHED (2826. Round-Robin Scheduling)
693. Problem TLE (2829. Time Limit Exceeded)
694. Problem DETER3 (2832. Find The Determinant III)
695. Problem SDGAME (2833. Super Dice Game)
696. Problem MLE (2835. Memory Limit Exceeded)
697. Problem BROKEN (2852. Broken Keyboard)
698. Problem PDECODE (2853. Decode the Strings)
699. Problem FOREST2 (2855. Forest)
700. Problem HELPBOB (2856. Help Bob)
701. Problem SDGAME2 (2877. Another understanding of Super Dice Game)
702. Problem KNIGHTS (2878. Knights of the Round Table)
703. Problem DOCTOR (2879. The Cow Doctor)

704. Problem WILD (2880. Wild West)
705. Problem CLONE (2881. Find the Clones)
706. Problem WARE (2882. The Warehouse)
707. Problem WIDGET (2883. Widget Factory)
708. Problem MARTIAN (2884. Martian Mining)
709. Problem WORDRING (2885. Word Rings)
710. Problem PARTY2 (2898. Party of Cloaked Killers)
711. Problem VOL (2899. Volunteers)
712. Problem GEOPROB (2901. One Geometry Problem)
713. Problem TRANSP1 (2903. Transportation)
714. Problem NOTATRI (2905. Not a Triangle)
715. Problem GCD2 (2906. GCD2)
716. Problem GSS5 (2916. Can you answer these queries V)
717. Problem QTREE5 (2939. Query on a tree V)
718. Problem UNTITLE1 (2940. Untitled Problem II)
719. Problem SHOOTING (2944. Emmons)
720. Problem ECLIPSE (2946. Eclipse)
721. Problem PAINTBLK (2962. Painting Blocks (Act I))
722. Problem PAINTBLC (2963. Painting Blocks (Act II))
723. Problem ELECTRO (3002. Electrophoretic)
724. Problem FILTER (3003. Median Filter)
725. Problem LIFEGAME (3004. Life Game)
726. Problem LAND (3005. Subdividing a Land)
727. Problem LINE (3006. Connect Line Segments)
728. Problem OILCOMP (3007. Oil Company)
729. Problem RPS (3008. Finding the Top RPS Player)
730. Problem VORONOI (3009. Revenge of Voronoi)
731. Problem WALL (3010. Castle Wall)
732. Problem SOLDIER (3033. Help the soldier)
733. Problem SEQ5 (3070. How many subsequences)
734. Problem MOD (3105. Power Modulo Inverted)
735. Problem DICTSUB (3106. Dictionary Subsequences)
736. Problem ODDDIV (3107. Odd Numbers of Divisors)
737. Problem GRAPHGAM (3108. Charlesbert and Merangelou)
738. Problem STRLCP (3109. Longest Common Prefix)
739. Problem PALNUM (3110. Palindromic Number)
740. Problem STABARDS (3111. Stabards)
741. Problem STSTRING (3112. Strings)
742. Problem GORELIAN (3133. Here We Go(relians) Again)
743. Problem PERMSG (3166. Permutation Exponentiation)
744. Problem LINES (3184. Game of Lines)
745. Problem DOORSPEN (3195. Doors and Penguins)
746. Problem PALIM (3208. Yet Another Longest Palindrome Problem)
747. Problem TYPESET (3249. Typesettin)
748. Problem SLINK (3251. Slink)
749. Problem EDS (3253. Electronic Document Security)
750. Problem GUARD (3254. Guard)

751. Problem RACETIME (3261. Race Against Time)
752. Problem SA04C (3305. Roman Patrollers)
753. Problem SA04D (3306. Very Special Boxes)
754. Problem HEXTILE (3307. Hex Tile Equations)
755. Problem BRIDGES2 (3308. The Bridges of San Mochti)
756. Problem BULLETIN (3309. Bulletin Board)
757. Problem SERIALN (3310. Serial Numbers)
758. Problem UMNOZAK (3314. Umnozak)
759. Problem DOUBLE (3322. Doubled Numbers)
760. Problem HIGHWAY (3347. Cestarine)
761. Problem STACK (3359. Stack)
762. Problem IMGREC2 (3360. Digital Image Recognition)
763. Problem SVADA (3363. Svada)
764. Problem ROUNDT (3372. Round Table)
765. Problem PERMCODE (3373. Permutation Code)
766. Problem SCAVHUNT (3374. Scavenger Hunt)
767. Problem STAMPS (3375. Stamps)
768. Problem PARKINGL (3376. Parking Lot)
769. Problem BUGLIFE (3377. A Bug's Life)
770. Problem MIRRORED (3378. Mirrored Pairs)
771. Problem SSHUFFLE (3379. String Shuffle)
772. Problem TOURIST (3380. Tourist)
773. Problem HIGHWAYS (3381. Highways)
774. Problem MONSTER (3382. Monster Trap)
775. Problem YODA (3385. Yoda Goes Palindromic !)
776. Problem QUALITY (3386. Contest System Quality Assurance Tester)
777. Problem CHMAZE (3387. Changing Maze)
778. Problem DNPALIN (3388. Double Near Palindromes)
779. Problem KNIGHTSR (3389. The Knights of the Round Circle)
780. Problem TRIBE2 (3390. Tribe Council)
781. Problem NOTOKNOT (3393. Knot or Not)
782. Problem LAGRANGE (3394. Lagrange's Four-Square Theorem)
783. Problem SAMER08A (3405. Almost Shortest Path)
784. Problem SAMER08B (3406. Bases)
785. Problem SAMER08C (3407. Candy)
786. Problem SAMER08D (3408. DNA Sequences)
787. Problem SAMER08E (3409. Electricity)
788. Problem SAMER08F (3410. Feynman)
789. Problem SAMER08G (3411. Pole Position)
790. Problem SAMER08H (3412. Higgs Boson)
791. Problem SAMER08I (3413. Traveling Shoemaker Problem)
792. Problem SAMER08J (3414. Bora Bora)
793. Problem SAMER08K (3415. Shrinking Polygons)
794. Problem FALLINGI (3420. Falling Ice)
795. Problem OROSNAKE (3426. Ouroboros Snake)
796. Problem HIST2 (3436. Histogram)
797. Problem LASTDIG (3442. The last digit)

798. Problem CEPC08B (3459. SkyScrapers)
799. Problem RAMP (3462. The Skatepark´s New Ramps)
800. Problem DRIVE (3465. Drive through MegaCity)
801. Problem DEPOSIT (3476. Deposit)
802. Problem BABY (3477. Baby)
803. Problem BEGIN (3483. Begin)
804. Problem CROSSBIT (3484. Crossbits)
805. Problem ELIM (3486. Elimination)
806. Problem TOPCODE (3488. The Top-Code)
807. Problem HIDTRI (3490. Hidden Triangle)
808. Problem BRAILLE (3492. Braille Transcription)
809. Problem NBLTHIEF (3495. The Nobel Thief)
810. Problem MATRICA (3543. Matrica)
811. Problem BST (3544. Binary Search Tree)
812. Problem NAJKRACI (3545. Najkraci)
813. Problem BOYSCOUT (3576. Boy Scouts)
814. Problem PARITY (3577. Parity)
815. Problem HASH (3578. Hashing)
816. Problem DISJPATH (3579. Disjoint Paths)
817. Problem COMPANY (3580. Company)
818. Problem TREESIM (3581. Tree Similarity)
819. Problem RSTAURNT (3582. Restaurant Tab)
820. Problem PATHEADS (3591. Patting Heads)
821. Problem CATTLEB (3678. Cattle Bruisers)
822. Problem MOOPIZZA (3679. Moo University - Emergency Pizza Order)
823. Problem KGSS (3693. Maximum Sum)
824. Problem PROOT (3713. Primitive Root)
825. Problem SNOOKER (3723. Snooker)
826. Problem RAINBOW (3724. Rainbow Ride)
827. Problem TREX (3725. Taming a T-REX)
828. Problem SUBSUMS (3749. Subset Sums)
829. Problem GEORGE (3763. George)
830. Problem STREET (3791. Street)
831. Problem LUBEN (3831. Lubenica)
832. Problem KRUS (3832. Kruska)
833. Problem TRES (3833. Tresnja)
834. Problem VCIRCLES (3863. Area of circles)
835. Problem RELJEF (3865. Reljef)
836. Problem VPALIN (3866. Finding Palindromes)
837. Problem VBOSS (3867. Who is The Boss)
838. Problem VMILI (3870. Military Story)
839. Problem GCDEX (3871. GCD Extreme)
840. Problem VPARTY (3872. Party At School)
841. Problem WHEN (3884. When (You Believe))
842. Problem BOBALLS (3894. Bouncing Balls)
843. Problem BYTESE1 (3920. Lucius Dungeon)
844. Problem BYTESE2 (3921. The Great Ball)

845. Problem BYTESM1 (3922. Mystical River)
846. Problem BYTESM2 (3923. Philosophers Stone)
847. Problem BYTESH1 (3924. Filchs Dilemna)
848. Problem FROGGER (3999. FROGGER)
849. Problem GALLUP (4000. GALLUP)
850. Problem SUBWAYPL (4003. Subway planning)
851. Problem CPU (4004. Exploding CPU)
852. Problem PHONELST (4033. Phone List)
853. Problem CUCKOO (4036. Cuckoo Hashing)
854. Problem KPGAME (4060. A game with probability)
855. Problem MORPH (4069. Morphing is Fun)
856. Problem TWOPROF (4070. Two Professors)
857. Problem EPALIN (4103. Extend to Palindrome)
858. Problem FASTFLOW (4110. Fast Maximum Flow)
859. Problem ELLIPSE (4142. Ellipse)
860. Problem DOMINO2 (4157. Domino)
861. Problem HS08PAUL (4164. A conjecture of Paul Erdős)
862. Problem HS08FOUR (4166. Four colors)
863. Problem SQFREE (4168. Square-free integers)
864. Problem DROOT (4172. Multiplicative digital root)
865. Problem KPURSUIT (4176. A Knightly Pursuit)
866. Problem HERDING (4177. Herding)
867. Problem LATTICE (4178. Distance on a square lattice)
868. Problem TEMPTISL (4179. Temptation Island)
869. Problem FCANDY (4182. Candy (Again))
870. Problem CCCCUBE (4185. Cube)
871. Problem HS08CODE (4186. Break a New RSA system)
872. Problem HS08EQ (4188. Amazing equality)
873. Problem LANDING (4189. Landing)
874. Problem DOMINOES (4197. Dominoes)
875. Problem LEGO (4198. Lego)
876. Problem HAMSTER1 (4200. Hamster flight)
877. Problem RATING (4201. Coder Ratings)
878. Problem BRPAR (4202. Brackets Parade)
879. Problem MATCHING (4206. Fast Maximum Matching)
880. Problem QUEEN (4235. Wandering Queen)
881. Problem TTTABLE (4273. Train TimeTable)
882. Problem AE3A (4305. Drilling)
883. Problem VOTE (4323. Voting Districts)
884. Problem EVERLAST (4324. The fate of the pineapple)
885. Problem EBOXES (4343. Empty Boxes)
886. Problem MATRIX1 (4357. Enter the Matrix)
887. Problem DAGCNT (4407. Counting Arborescence)
888. Problem FENCE1 (4408. Build a Fence)
889. Problem AREA1 (4409. Circle vs Triangle)
890. Problem REPAIR1 (4410. Repair the Door)
891. Problem EXPR3 (4411. Counting Expressions)

892. Problem FACTOR1 (4412. Factorization, Factorization, Factorization)
893. Problem GEM (4413. Gem)
894. Problem HIGHWAY1 (4414. Highway)
895. Problem INTEGER1 (4415. Power of Integer)
896. Problem JUMP1 (4416. Jumping Hands)
897. Problem KPGRAPHS (4420. Counting Graphs)
898. Problem GF2 (4421. Irreducible polynomials over GF2)
899. Problem MIB (4429. Spelling Lists)
900. Problem ARITH2 (4452. Simple Arithmetics II)
901. Problem BOBALLS2 (4453. Bouncing Balls II)
902. Problem BRCKTS2 (4454. Brackets II)
903. Problem MOVIE (4455. Going to the Movies)
904. Problem AIRLINES (4456. Jumbo Airlines)
905. Problem SHOP2 (4457. Shopping II)
906. Problem TITATO (4460. Tic Tac Toe - Best Move)
907. Problem ANTTT (4465. The Ant)
908. Problem PLAYFAIR (4476. Playfair Cracker)
909. Problem EXPR4 (4478. Counting Expressions II)
910. Problem GSS6 (4487. Can you answer these queries VI)
911. Problem PGCD (4491. Primes in GCD Table)
912. Problem MAGIC1 (4492. Magic1)
913. Problem EQUAD1 (4493. Equation)
914. Problem UCI2009B (4523. Binomial Coefficients)
915. Problem UCI2009D (4525. Digger Octaves)
916. Problem FROGS (4528. Frog Wrestling)
917. Problem BANDMATR (4533. Determinant of Banded Matrices)
918. Problem ANARC08A (4546. Tobo or not Tobo)
919. Problem ANARC08B (4549. Adding Sevens)
920. Problem ANARC08C (4551. Match Maker)
921. Problem ANARC08D (4552. Adding up Triangles)
922. Problem ANARC08E (4554. Relax! It is just a game)
923. Problem ANARC08F (4555. Einbahnstrasse)
924. Problem ANARC08G (4556. Think I will Buy Me a Football Team)
925. Problem ANARC08H (4557. Musical Chairs)
926. Problem ANARC08I (4558. I Speak Whales)
927. Problem ANARC08J (4559. A Day at the Races)
928. Problem LQDRECT (4566. Đe^’m hinh chu+~ nhât)
929. Problem CYCLERUN (4574. Riding in cycles)
930. Problem ABCDEF (4580. ABCDEF)
931. Problem GCJ08C (4585. Star Wars)
932. Problem WLOO0707 (4586. Texas Trip)
933. Problem FENCE3 (4587. Electric Fences)
934. Problem NWERC04H (4588. SETI)
935. Problem PMATRIX (4644. Proving Equivalences)
936. Problem CCROSS (4656. Cross Mountain Climb)
937. Problem GASWARS (4657. Gas Wars)
938. Problem HHEMANT (4658. Help Hemant Verma)

SPOJ Problem Set (classical)

1. Life, the Universe, and Everything

Problem code: TEST

Your program is to use the brute-force approach in order to *find the Answer to Life, the Universe, and Everything*. More precisely... rewrite small numbers from input to output. Stop processing input after reading in the number 42. All numbers at input are integers of one or two digits.

Example

Input:

```
1
2
88
42
99
```

Output:

```
1
2
88
```

Added by: Michał Małafiejski
Date: 2004-05-01
Time limit: 3s
Source limit:50000B
Languages: All
Resource: Douglas Adams, The Hitchhiker's Guide to the Galaxy

SPOJ Problem Set (classical)

2. Prime Generator

Problem code: PRIME1

Peter wants to generate some prime numbers for his cryptosystem. Help him! Your task is to generate all prime numbers between two given numbers!

Input

The input begins with the number t of test cases in a single line (t<=10). In each of the next t lines there are two numbers m and n (1 <= m <= n <= 1000000000, n-m<=100000) separated by a space.

Output

For every test case print all prime numbers p such that m <= p <= n, one number per line, test cases separated by an empty line.

Example

Input:

```
2
1 10
3 5
```

Output:

```
2
3
5
7

3
5
```

Warning: large Input/Output data, be careful with certain languages (though most should be OK if the algorithm is well designed)

Added by: Adam Dzedzej
Date: 2004-05-01
Time limit: 6s
Source limit:50000B
Languages: All

SPOJ Problem Set (classical)

3. Substring Check (Bug Funny)

Problem code: SBSTR1

Given two binary strings, A (of length 10) and B (of length 5), output 1 if B is a substring of A and 0 otherwise.

Please note, that the solution may only be submitted in the following languages: Brainf**k, Whitespace and Intercal.

Input

24 lines consisting of pairs of binary strings A and B separated by a single space.

Output

The logical value of: 'B is a substring of A'.

Example

First two lines of input:
1010110010 10110
1110111011 10011

First two lines of output:
1
0

Added by: Adrian Kosowski
Date: 2004-05-01
Time limit: 7s
Source limit:50000B
Languages: WSPC BF ICK

SPOJ Problem Set (classical)

4. Transform the Expression

Problem code: ONP

Transform the algebraic expression with brackets into RPN form (Reverse Polish Notation). Two-argument operators: +, -, *, /, ^ (priority from the lowest to the highest), brackets ( ). Operands: only letters: a,b,...,z. Assume that there is only one RPN form (no expressions like a*b*c).

Input

t [the number of expressions <= **100**]
expression [length <= **400**]
[other expressions]

Text grouped in [ ] does not appear in the input file.

Output

The *expressions* in RPN form, one per line.

Example

```
Input:
3
(a+(b*c))
((a+b)*(z+x))
((a+t)*((b+(a+c))^(c+d)))

Output:
abc*+
ab+zx+*
at+bac++cd+^*
```

Added by: Michał Małafiejski
Date: 2004-05-01
Time limit: 5s
Source limit:50000B
Languages: All
Resource: -

SPOJ Problem Set (classical)

5. The Next Palindrome

Problem code: PALIN

A positive integer is called a *palindrome* if its representation in the decimal system is the same when read from left to right and from right to left. For a given positive integer K of not more than 1000000 digits, write the value of the smallest palindrome larger than K to output. Numbers are always displayed without leading zeros.

Input

The first line contains integer t, the number of test cases. Integers K are given in the next t lines.

Output

For each K, output the smallest palindrome larger than K.

Example

`Input:`
2
808
2133

`Output:`
818
2222

Warning: large Input/Output data, be careful with certain languages

---

Added by: Adrian Kosowski
Date: 2004-05-01
Time limit: 9s
Source limit:50000B
Languages: All

SPOJ Problem Set (classical)

6. Simple Arithmetics

Problem code: ARITH

One part of the new WAP portal is also acalculator computing expressions with very long numbers. To make the output look better, the result is formated the same way as is it usually used with manual calculations.

Your task is to write the core part of this calculator. Given two numbers and the requested operation, you are to compute the result and print it in the form specified below. With addition and subtraction, the numbers are written below each other. Multiplication is alittle bit more complex: first of all, we make apartial result for every digit of one of the numbers, and then sum the results together.

Input

There is asingle positive integer *T* on the first line of input (equal to about 1000). It stands for the number of expressions to follow. Each expression consists of asingle line containing apositive integer number, anoperator (one of +, − and *) and the second positive integer number. Every number has at most 500digits. There are no spaces on the line. If the operation is subtraction, the second number is always lower than the first one. No number will begin with zero.

Output

For each expression, print two lines with two given numbers, the second number below the first one, last digits (representing unities) must be aligned in the same column. Put the operator right in front of the first digit of the second number. After the second number, there must be ahorizontal line made of dashes (−).

For each addition or subtraction, put the result right below the horizontal line, with last digit aligned to the last digit of both operands.

For each multiplication, multiply the first number by each digit of the second number. Put the partial results one below the other, starting with the product of the last digit of the second number. Each partial result should be aligned with the corresponding digit. That means the last digit of the partial product must be in the same column as the digit of the second number. No product may begin with any additional zeros. If aparticular digit is zero, the product has exactly one digit -- zero. If the second number has more than one digit, print another horizontal line under the partial results, and then print the sum of them.

There must be minimal number of spaces on the beginning of lines, with respect to other constraints. The horizontal line is always as long as necessary to reach the left and right end of both numbers (and operators) directly below and above it. That means it begins in the same column where the leftmost digit or operator of that two lines (one below and one above) is. It ends in the column where is the rightmost digit of that two numbers. The line can be neither longer nor shorter than specified.

Print one blank line after each test case, including the last one.

Example

Sample Input:

```
4
12345+67890
324-111
325*4405
1234*4
```

Sample Output:

```
 12345
+67890
------
 80235

 324
-111
----
 213

    325
  *4405
  -----
   1625
     0
 1300
1300
-------
1431625

1234
  *4
----
4936
```

Warning: large Input/Output data, be careful with certain languages.

Added by: Adrian Kosowski
Date: 2004-05-08
Time limit: 5s
Source limit: 50000B
Languages: All
Resource: ACM Central European Programming Contest, Prague 2000

SPOJ Problem Set (classical)

7. The Bulk!

Problem code: BULK

ACM uses anew special technology of building its transceiver stations. This technology is called *Modular Cuboid Architecture* (*MCA*) and is covered by apatent of Lego company. All parts of the transceiver are shipped in unit blocks that have the form of cubes of exactly the same size. The cubes can be then connected to each other. The MCA is modular architecture, that means we can select preferred transceiver configuration and buy only those components we need .

The cubes must be always connected "face-to-face", i.e. the whole side of one cube is connected to the whole side of another cube. One cube can be thus connected to at most six other units. The resulting equipment, consisting of unit cubes is called *The Bulk* in the communication technology slang.

Sometimes, anold and unneeded bulk is condemned, put into astorage place, and replaced with anew one. It was recently found that ACM has many of such old bulks that just occupy space and are no longer needed. The director has decided that all such bulks must be disassembled to single pieces to save some space. Unfortunately, there is no documentation for the old bulks and nobody knows the exact number of pieces that form them. You are to write acomputer program that takes the bulk description and computes the number of unit cubes.

Each bulk is described by its faces (sides). Aspecial X-ray based machine was constructed that is able to localise all faces of the bulk in the space, even the inner faces, because the bulk can be partially hollow (it can contain empty spaces inside). But any bulk must be connected (i.e. it cannot drop into two pieces) and composed of whole unit cubes.

Input

There is asingle positive integer T on the first line of input (equal to about 1000). It stands for the number of bulks to follow. Each bulk description begins with aline containing single positive integer F, $6 <= F <= 250$, stating the number of faces. Then there are F lines, each containing one face description. All faces of the bulk are always listed, in any order. Any face may be divided into several distinct parts and described like if it was more faces. Faces do not overlap. Every face has one inner side and one outer side. No side can be "partially inner and partially outer".

Each face is described on asingle line. The line begins with aninteger number P stating the number of points that determine the face, $4 <= P <= 200$. Then there are $3 \times P$ numbers, coordinates of the points. Each point is described by three coordinates X,Y,Z ($0 <= X,Y,Z <= 1000$) separated by spaces. The points are separated from each other and from the number P by two space characters. These additional spaces were added to make the input more human readable. The face can be constructed by connecting the points in the specified order, plus connecting the last point with the first one.

The face is always composed of "unit squares", that means every edge runs either in X, Y or Z-axis direction. If we take any two neighbouring points X_1,Y_1,Z_1 and X_2,Y_2,Z_2, then the points will always differ in exactly one of the three coordinates. I.e. it is either $X_1 <> X_2$, or $Y_1 <> Y_2$, or $Z_1 <>$

Z_2, other two coordinates are the same. Every face lies in anorthogonal plane, i.e. exactly one coordinate is always the same for all points of the face. The face outline will never touch nor cross itself.

Output

Your program must print asingle line for every test case. The line must contain the sentence `The bulk is composed of` *V* `units.`, where *V* is the volume of the bulk.

Example

Sample Input:

```
2
12
4  10 10 10  10 10 20  10 20 20  10 20 10
4  20 10 10  20 10 20  20 20 20  20 20 10
4  10 10 10  10 10 20  20 10 20  20 10 10
4  10 20 10  10 20 20  20 20 20  20 20 10
4  10 10 10  10 20 10  20 20 10  20 10 10
5  10 10 20  10 20 20  20 20 20  20 15 20  20 10 20
4  14 14 14  14 14 16  14 16 16  14 16 14
4  16 14 14  16 14 16  16 16 16  16 16 14
4  14 14 14  14 14 16  16 14 16  16 14 14
4  14 16 14  14 16 16  16 16 16  16 16 14
4  14 14 14  14 16 14  16 16 14  16 14 14
4  14 14 16  14 16 16  16 16 16  16 14 16
12
4  20 20 30  20 30 30  30 30 30  30 20 30
4  10 10 10  10 40 10  40 40 10  40 10 10
6  10 10 20  20 10 20  20 30 20  30 30 20  30 40 20  10 40 20
6  20 10 20  20 20 20  30 20 20  30 40 20  40 40 20  40 10 20
4  10 10 10  40 10 10  40 10 20  10 10 20
4  10 40 10  40 40 10  40 40 20  10 40 20
4  20 20 20  30 20 20  30 20 30  20 20 30
4  20 30 20  30 30 20  30 30 30  20 30 30
4  10 10 10  10 40 10  10 40 20  10 10 20
4  40 10 10  40 40 10  40 40 20  40 10 20
4  20 20 20  20 30 20  20 30 30  20 20 30
4  30 20 20  30 30 20  30 30 30  30 20 30
```

Sample Output:

```
The bulk is composed of 992 units.
The bulk is composed of 10000 units.
```

Warning: large Input/Output data, be careful with certain languages

---

Added by: Adrian Kosowski
Date: 2004-05-08
Time limit: 7s
Source limit: 50000B
Languages: All
Resource: ACM Central European Programming Contest, Prague 2000

SPOJ Problem Set (classical)

8. Complete the Sequence!

Problem code: CMPLS

You probably know those quizzes in Sunday magazines: given the sequence 1, 2, 3, 4, 5, what is the next number? Sometimes it is very easy to answer, sometimes it could be pretty hard. Because these "sequence problems" are very popular, ACM wants to implement them into the "Free Time" section of their new WAP portal.

ACM programmers have noticed that some of the quizzes can be solved by describing the sequence by polynomials. For example, the sequence 1, 2, 3, 4, 5 can be easily understood as atrivial polynomial. The next number is 6. But even more complex sequences, like 1, 2, 4, 7, 11, can be described by apolynomial. In this case, $1/2.n^2-1/2.n+1$ can be used. Note that even if the members of the sequence are integers, polynomial coefficients may be any real numbers.

Polynomial is anexpression in the following form:

$$P(n) = a_D.n^D+a_{D-1}.n^{D-1}+...+a_1.n+a_0$$

If a_D <> 0, the number D is called a*degree* of the polynomial. Note that constant function $P(n) = C$ can be considered as polynomial of degree 0, and the zero function $P(n) = 0$ is usually defined to have degree -1.

Input

There is asingle positive integer T on the first line of input (equal to about 5000). It stands for the number of test cases to follow. Each test case consists of two lines. First line of each test case contains two integer numbers S and C separated by asingle space, $1 <= S < 100$, $1 <= C < 100$, $(S+C) <= 100$. The first number, S, stands for the length of the given sequence, the second number, C is the amount of numbers you are to find to complete the sequence.

The second line of each test case contains S integer numbers X_1, X_2, ... X_S separated by aspace. These numbers form the given sequence. The sequence can always be described by apolynomial $P(n)$ such that for every i, $X_i = P(i)$. Among these polynomials, we can find the polynomial P_{min} with the lowest possible degree. This polynomial should be used for completing the sequence.

Output

For every test case, your program must print asingle line containing C integer numbers, separated by aspace. These numbers are the values completing the sequence according to the polynomial of the lowest possible degree. In other words, you are to print values $P_{min}(S+1)$, $P_{min}(S+2)$, ... $P_{min}(S+C)$.

It is guaranteed that the results $P_{min}(S+i)$ will be non-negative and will fit into the standard *integer* type.

Example

Sample Input:

```
4
6 3
1 2 3 4 5 6
8 2
1 2 4 7 11 16 22 29
10 2
1 1 1 1 1 1 1 1 1 2
1 10
3
```

Sample Output:

```
7 8 9
37 46
11 56
3 3 3 3 3 3 3 3 3 3
```

Warning: large Input/Output data, be careful with certain languages

---

Added by: Adrian Kosowski
Date: 2004-05-08
Time limit: 5s
Source limit: 50000B
Languages: All
Resource: ACM Central European Programming Contest, Prague 2000

SPOJ Problem Set (classical)

9. Direct Visibility

Problem code: DIRVS

Building the GSM network is avery expensive and complex task. Moreover, after the *Base Transceiver Stations* (*BTS*) are built and working, we need to perform many various measurements to determine the state of the network, and propose effective improvements to be made.

The ACM technicians have aspecial equipment for measuring the strength of electro-magnetic fields, the transceivers' power and quality of the signal. This equipment is packed into ahuge knapsack and the technician must move with it from one BTS to another. Unfortunately, the knapsack have not enough memory for storing all of the measured values. It has asmall cache only, that can store values for several seconds. Then the values must be transmitted to the BTS by aninfrared connection (IRDA). The IRDA needs direct visibility between the technician and the BTS.

Your task is to find the path between two neighbouring BTSes such that at least one of those BTSes is always visible.

Input

There is asingle positive integer T on the first line of input (equal to about 500). It stands for the number of test cases to follow. Each test case consists of atown description. For simplicity, atown is modelled as arectangular grid of P *x* Q square fields. Each field is exactly 1metre wide. For each field, anon-negative integer $Z_{i,j}$ is given, representing the height of the terrain in that place, in metres. That means the town model is made of cubes, each of them being either solid or empty. There are no "half solid" cubes.

The first line of each test case contains two integer numbers P and Q, separated by asingle space, $1 <= P,Q <= 200$. Then there are P lines each containing Q integer numbers separated by aspace. These numbers are $Z_{i,j}$, where $1 <= i <= P$, $1 <= j <= Q$ and $0 <= Z_{i,j} <= 5000$. After the terrain description, there are four numbers R_1, C_1, R_2, C_2 on the last line of each test case. These numbers represent position of two BTSes, $1 <= R_1,R_2 <= P$, $1 <= C_1,C_2 <= Q$. The first coordinate (R) determines the row of the town, the second coordinate determines the column.

The technician is moving in steps (*steps* stands for *Standard Technician's Elementary Positional Shift*). Each step is made between two neighbouring square fields. That means the step is always in North, South, West or East direction. It is not possible to move diagonally. The step between two fields A and B (step from A to B) is allowed only if the height of the terrain in the field B is not very different from the height in the field A. The technician can climb at most 1metre up or descend at most 3metres down in asingle step.

At the end of each step, at least one of the two BTSes must be visible. However, there can be some point "in the middle of the step" where no BTS is visible. This is OK and the data is handled by the cache. The BTS is considered visible, if there is adirect visibility between the unit cube just above the terrain on the BTSes coordinates and the cube just above the terrain on the square field, where the technician is. Direct visibility between two cubes means that the line connecting the centres of the two

cubes does not intersect any solid cube. However, the line can touch any number of solid cubes. In other words, consider both the BTS and the technician being points exactly half metre above the surface and in the centre of the appropriate square field.

Note that the IRDA beam can go between two cubes that touch each other by their edge, although there is no space between them. It is because such abeam touches both of these two cubes but does not intersect any of them. See the last test case of the sample input for anexample of such asituation.

Output

You are to find the shortest possible path from BTS (R_1, C_1) to BTS (R_2, C_2), meeting the above criteria. All steps must be done between neighbouring fields, the terrain must not elevate or descend too much, and at the end of each step, at least one BTS must be visible.

For each test case, print one line containing the sentence `The shortest path is` *M* `steps long.`, where *M* is the number of steps that must be made. If there is no such path, output the sentence `Mission impossible!`.

Example

Sample Input:

```
4
5 5
8 7 6 5 4
2 2 2 2 2
2 2 2 2 2
2 2 2 2 2
2 2 2 2 2
1 1 5 1
5 8
2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2
9 9 9 9 9 9 9 2
2 2 2 2 2 2 2 2
1 2 5 1
5 8
2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2
2 2 2 2 2 2 2 2
9 9 9 9 9 9 9 2
2 2 2 2 2 2 2 2
1 5 5 1
6 12
5 5 5 5 1 5 5 5 5 5 5 5
5 5 5 5 1 5 5 5 5 5 5 5
5 5 5 5 9 5 5 5 5 5 5 5
5 9 1 5 5 5 5 5 5 5 5 5
5 5 9 5 5 5 5 5 5 5 5 5
5 5 9 5 5 5 5 5 5 5 5 5
6 1 3 12
```

Sample Output:

```
The shortest path is 10 steps long.
Mission impossible!
The shortest path is 14 steps long.
The shortest path is 18 steps long.
```

Added by: Adrian Kosowski
Date: 2004-05-08
Time limit: 13s
Source limit: 50000B
Languages: All
Resource: ACM Central European Programming Contest, Prague 2000

SPOJ Problem Set (classical)

10. Complicated Expressions

Problem code: CMEXPR

The most important activity of ACM is the GSM network. As the mobile phone operator, ACM must build its own transmitting stations. It is very important to compute the exact behaviour of electro-magnetic waves. Unfortunately, prediction of electro-magnetic fields is avery complex task and the formulas describing them are very long and hard-to-read. For example, Maxwell's Equations describing the basic laws of electrical engineering are really tough.

ACM has designed its own computer system that can make some field computations and produce results in the form of mathematic expressions. Unfortunately, by generating the expression in several steps, there are always some unneeded parentheses inside the expression. Your task is to take these partial results and make them "nice" by removing all unnecessary parentheses.

Input

There is asingle positive integer T on the first line of input (equal to about 10000). It stands for the number of expressions to follow. Each expression consists of asingle line containing only lowercase letters, operators (`+`, `-`, `*`, `/`) and parentheses (`(` and `)`). The letters are variables that can have any value, operators and parentheses have their usual meaning. Multiplication and division have higher priority then subtraction and addition. All operations with the same priority are computed from left to right (operators are left-associative). There are no spaces inside the expressions. No input line contains more than 250 characters.

Output

Print asingle line for every expression. The line must contain the same expression with unneeded parentheses removed. You must remove as many parentheses as possible without changing the semantics of the expression. The semantics of the expression is considered the same if and only if any of the following conditions hold:

- The ordering of operations remains the same. That means "(a+b)+c" is the same as "a+b+c", and "a+(b/c)" is the same as "a+b/c".
- The order of some operations is swapped but the result remains unchanged with respect to the addition and multiplication associativity. That means "a+(b+c)" and "(a+b)+c" are the same. We can also combine addition with subtraction and multiplication with division, if the subtraction or division is the second operation. For example, "a+(b-c)" is the same as "a+b-c".

You cannot use any other laws, namely you cannot swap left and right operands and you cannot replace "a-(b-c)" with "a-b+c".

Example

Sample Input:

```
8
(a+(b*c))
((a+b)*c)
(a*(b*c))
(a*(b/c)*d)
((a/(b/c))/d)
((x))
(a+b)-(c-d)-(e/f)
(a+b)+(c-d)-(e+f)
```

Sample Output:

```
a+b*c
(a+b)*c
a*b*c
a*b/c*d
a/(b/c)/d
x
a+b-(c-d)-e/f
a+b+c-d-(e+f)
```

Added by: Adrian Kosowski
Date: 2004-05-09
Time limit: 5s
Source limit: 50000B
Languages: All
Resource: ACM Central European Programming Contest, Prague 2000

SPOJ Problem Set (classical)

11. Factorial

Problem code: FCTRL

The most important part of aGSM network is so called *Base Transceiver Station* (*BTS*). These transceivers form the areas called *cells* (this term gave the name to the cellular phone) and every phone connects to the BTS with the strongest signal (in alittle simplified view). Of course, BTSes need some attention and technicians need to check their function periodically.

ACM technicians faced avery interesting problem recently. Given aset of BTSes to visit, they needed to find the shortest path to visit all of the given points and return back to the central company building. Programmers have spent several months studying this problem but with no results. They were unable to find the solution fast enough. After along time, one of the programmers found this problem in aconference article. Unfortunately, he found that the problem is so called "Travelling Salesman Problem" and it is very hard to solve. If we have *N* BTSes to be visited, we can visit them in any order, giving us *N*! possibilities to examine. The function expressing that number is called factorial and can be computed as aproduct 1.2.3.4....*N*. The number is very high even for arelatively small *N*.

The programmers understood they had no chance to solve the problem. But because they have already received the research grant from the government, they needed to continue with their studies and produce at least *some* results. So they started to study behaviour of the factorial function.

For example, they defined the function *Z*. For any positive integer *N*, *Z*(*N*) is the number of zeros at the end of the decimal form of number *N*!. They noticed that this function never decreases. If we have two numbers $N_1 < N_2$, then $Z(N_1) <= Z(N_2)$. It is because we can never "lose" any trailing zero by multiplying by any positive number. We can only get new and new zeros. The function *Z* is very interesting, so we need acomputer program that can determine its value efficiently.

Input

There is asingle positive integer *T* on the first line of input (equal to about 100000). It stands for the number of numbers to follow. Then there are *T* lines, each containing exactly one positive integer number *N*, 1 <= *N* <= 1000000000.

Output

For every number *N*, output asingle line containing the single non-negative integer *Z*(*N*).

Example

Sample Input:

```
6
3
60
100
1024
23456
8735373
```

Sample Output:

```
0
14
24
253
5861
2183837
```

Added by: Adrian Kosowski
Date: 2004-05-09
Time limit: 6s
Source limit: 50000B
Languages: All
Resource: ACM Central European Programming Contest, Prague 2000

SPOJ Problem Set (classical)

12. The Game of Master-Mind

Problem code: MMIND

If you want to buy anew cellular phone, there are many various types to choose from. To decide which one is the best for you, you have to consider several important things: its size and weight, battery capacity, WAP support, colour, price. One of the most important things is also the list of games the phone provides. Nokia is one of the most successful phone makers because of its famous Snake and SnakeII. ACM wants to make and sell its own phone and they need to program several games for it. One of them is Master-Mind, the famous board logical game.

The game is played between two players. One of them chooses a*secret code* consisting of *P*ordered pins, each of them having one of the predefined set of *C*colours. The goal of the second player is to guess that secret sequence of colours. Some colours may not appear in the code, some colours may appear more than once.

The player makes guesses, which are formed in the same way as the secret code. After each guess, he/she is provided with aninformation on how successful the guess was. This feedback is called a*hint*. Each hint consists of *B* black points and *W* white points. The black point stands for every pin that was guessed right, i.e. the right colour was put on the right position. The white point means right colour but on the wrong position. For example, if the secret code is "white, yellow, red, blue, white" and the guess was "white, red, white, white, blue", the hint would consist of one black point (for the white on the first position) and three white points (for the other white, red and blue colours). The goal is to guess the sequence with the minimal number of hints.

The new ACM phone should have the possibility to play both roles. It can make the secret code and give hints, but it can also make its own guesses. Your goal is to write aprogram for the latter case, that means aprogram that makes Master-Mind guesses.

Input

There is asingle positive integer *T* on the first line of input. It stands for the number of test cases to follow. Each test case describes one game situation and you are to make aguess. On the first line of each test case, there are three integer numbers, *P*, *C* and *M*. *P* (1 <= *P* <= 10) is the number of pins, *C* (1 <= *C* <= 100) is the number of colours, and *M* (1 <= *M* <= 100) is the number of already played guesses.

Then there are 2 *x M* lines, two lines for every guess. At the first line of each guess, there are *P* integer numbers representing colours of the guess. Each colour is represented by anumber G_i, 1 <= G_i <= *C*. The second line contains two integer numbers, *B* and *W*, stating for the number of black and white points given by the corresponding hint.

Let's have asecret code S_1, S_2, ... S_P and the guess G_1, G_2, ... G_P. Then we can make aset *H* containing pairs of numbers (*I*,*J*) such that $S_I = G_J$, and that any number can appear at most once on the first position and at most once on the second position. That means for every two different pairs from that set, (I_1,J_1) and (I_2,J_2), we have $I_1 <> I_2$ and $J_1 <> J_2$. Then we denote *B*(*H*) the number

of pairs in the set, that meet the condition $I = J$, and $W(H)$ the number of pairs with $I <> J$.

We define anordering of every two possible sets H_1 and H_2. Let's say $H_1 <= H_2$ if and only if one of the following holds:

- $B(H_1) < B(H_2)$, or
- $B(H_1) = B(H_2)$ and $W(H_1) <= W(H_2)$

Then we can find amaximal set H_{max} according to this ordering. The numbers $B(H_{max})$ and $W(H_{max})$ are the black and white points for that hint.

Output

For every test case, print the line containing P numbers representing P colours of the next guess. Your guess must be valid according to all previous guesses and hints. The guess is valid if the sequence could be asecret code, i.e. the sequence was not eliminated by previous guesses and hints.

If there is no valid guess possible, output the sentence `You are cheating!`. If there are more valid guesses, output the one that is lexicographically smallest. I.e. find such guess G that for every other valid guess V there exists such anumber I that:

- $G_J = V_J$ for every $J<I$, and
- $G_I<V_I$.

Example

Sample Input:

```
3
4 3 2
1 2 3 2
1 1
2 1 3 2
1 1
4 6 2
3 3 3 3
3 0
4 4 4 4
2 0
8 9 3
1 2 3 4 5 6 7 8
0 0
2 3 4 5 6 7 8 9
1 0
3 4 5 6 7 8 9 9
2 0
```

Sample Output

```
1 1 1 3
You are cheating!
9 9 9 9 9 9 9 9
```

Warning: large Input/Output data, be careful with certain languages

---

Added by:	Adrian Kosowski
Date:	2004-05-09
Time limit:	7s
Source limit:	50000B
Languages:	All
Resource:	ACM Central European Programming Contest, Prague 2000

SPOJ Problem Set (classical)

13. Hotline

Problem code: HOTLINE

Every customer sometimes needs help with new and unusual products. Therefore, hotline service is very important for every company. We need asingle phone number where the customer can always find afriendly voice ready to help with anything. On the other hand, many people are needed to serve as hotline operators, and human resources are always very expensive. Moreover, it is not easy to pretend "friendly voice" at 4am and explain to adrunken man that you are really unable to give him the number to House of Parliament. It was also found that some of the questions repeat very often and it is very annoying to answer them again and again.

ACM is amodern company, wanting to solve its hotline problem. They want to decrease the number of human operators by creating acomplex software system that would be able to answer most common questions. The customer's voice is analysed by aspecial Voice Recognition Module (VRM) and converted to aplain text. The text is then taken by an Artificial Automatic Adaptive Answering Algorithm (AAAAA). The most common questions are recognised and answered automatically. The replies are then converted to asound by Text-to-Speech Module (TTS).

You are to write the AAAAA module. Because your algorithm should be adaptive, it has no explicit knowledge base. But it must be able to listen to sentences in English and remember the mentioned facts. Whenever the question is asked about such afact, the system has to answer it properly. The VRM and TTS modules are already implemented, so the input and output of AAAAA will be in the text form.

Input

There is asingle positive integer *T* on the first line of input. It stands for the number of dialogues to follow. Each dialogue consists of zero or more lines, each of them containing one sentence: either statement or question. The statement ends with adot character (`.`), the question ends with aquestion mark (`?`). No statement will appear more than once, however the questions can be repeated. There is one extra line after each dialogue. That line ends with anexclamation mark (`!`).

Sentences can contain words, spaces and punctuation characters (such as commas, colons, semicolons etc.). All words contain only letters of English alphabet and are case-sensitive. That means the same word is always written the same way, usually in lowercase. Acronyms, names and some other words can begin with capital letters. For simplicity, all sentences begin with alowercase letter. Only if the first word should be written with acapital, the sentence begins with acapital letter. There are no unneeded spaces between words. No line will have more than 100 characters. There will be at most 100 statements per each test case.

Statements

Each statement has one of the following two forms (_ denotes aspace):
subject _*predicate*`[s]` [_*object*] **.**
subject _`don't|doesn't` _*predicate* [_*object*] **.**

The square brackets mark anoptional part, the vertical line two possible variants. Subject is asingle word, noun or pronoun in singular. Predicate is averb (single word) denoting some activity. Object can be any text. Object does not contain any dots. Any pair "verb + object" determines unique activity. The same verb with different objects makes different independent activities, i.e. the different and independent meaning of the sentence. Sentence without any object can be considered as sentence with anempty object. The verb without anobject has different and independent meaning than the same verb with any non-empty object.

The first variant of sentence denotes apositive statement. The word "*predicate*[s]" means verb that matches the subject of the sentence. If the subject is "I" or "you", the verb has the same form as the infinitive. With any other subject, the letter "s" is appended on the end of the verb. Assume there are no irregular verbs.

The second variant is anegative statement. Verb "don't" or "doesn't" must also match the subject. The form "don't" is used with either "I" or "you", "doesn't" is used in any other case.

Aspecial generic subject "everybody" can be used. It means the activity holds for any subject. Other special subject is "nobody". Such sentence also holds for any subject, but its meaning is negative. Both of these generic subjects can be used with the first variant only (without "doesn't"). The sentence "nobody likes something" is exactly equal to "everybody doesn't like something", except the latter form will never occur in the input.

Questions

Each question has one of the following three forms:

1. doldoes *_subject _predicate* [_object] ?
2. who *_predicate*s [*_object*] ?
3. what _doldoes *_subject* do ?

The word "doldoes" always matches the subject ("do I?", "do you?", "does any other subject?"). In the second type of question, predicate always matches the word "who", i.e. the "s" is always appended. Generic subjects cannot be used in questions.

Output

For each dialogue, your program must output the line Dialogue #*D*:, where *D* is the sequence number of dialogue, starting with 1. Then print exactly three lines for every question: the first line repeats the question, the second line contains the answer, and the third line is empty. Print nothing for statements. After each dialogue, print the same line with anexclamation mark that was in the input. Then print one extra empty line. Empty line contains anew-line character only.

The answer must be properly formated to be accepted by aTTS module. Only the statements appearing in the input before the answer are used for the corresponding reply. If there is any contradiction among statements, the reply is always I am abroad.. If the question and statements consider the special subject "you", it must be replaced with "I" in the answer. If the question considers special subject "I", it must be replaced with "you" in the answer. The verb must always match the subject of the sentence. The exact form of the correct answer depends on the type of question.

1.does subject predicate [object] ?

If there is any positive statement about the mentioned subject (or generic subject "everybody"), predicate and object, the answer is:
yes, *_subject _predicate*[s] [*_object*] .

If there is any negative statement about the mentioned subject (or generic subject "nobody"), predicate and object, the answer is:
no, *_subject* _don't|doesn't *_predicate* [*_object*] .

Otherwise, the answer is: maybe.

Subject in the answer is always the same subject as the subject of the question.

2.who predicates [object] ?

If there is apositive statement considering any subject, the specified predicate and object, the answer is:
subject _predicate[s] [*_object*] .

If two or more subjects match the activity, replace the subject in the answer with enumeration of all such subjects, in the same order as the corresponding statements have appeared in the input. Subjects are separated with comma and space, last two subjects are separated with the word "and". If "everybody" belongs to the group of enumerated subjects, do not enumerate subjects, and print "everybody" only. If the enumeration contains at least two subjects, the predicate matches the plural subject (i.e. verb is without trailing "s"), otherwise it matches the only subject.
subject1 , _subject2 _and *_subject3 predicate* [*_object*] .

If there is anegative statement considering the generic subject "nobody", the specified predicate and object, the answer is:
*nobody _predicate*s [*_object*] .

Otherwise, the answer is: I don't know.

3.what does subject do ?

If there are one or more sentences (both positive and negative) considering the specified subject (or ageneric subject "everybody" or "nobody"), all verbs and objects from such sentences must be included in areply in the same order as the corresponding sentences have appeared in the input. No verb-object pair can be included more than once (the eventual second appearance must be skipped). The verb-object pairs are separated by acomma followed by aspace, the last verb is separated by acomma and the word "and". Please note the comma is printed here although there was no comma when separating the subjects in the previous type of answer (see above). The negative answers have the same form as the statements, that means the verb "don't" or "doesn't" is used:
subject [_don't|doesn't] *_predicate1*[s] [*_object1*] ,
[_don't|doesn't] *_predicate2*[s] [*_object2*] ,
_and [_don't|doesn't] *_predicate3*[s] [*_object3*] .
subject [_don't|doesn't] *_predicate1*[s] [*_object1*] ,
_and [_don't|doesn't] *_predicate2*[s] [*_object2*] .
subject [_don't|doesn't] *_predicate*[s] [*_object*] .

Otherwise, the answer is: `I don't know.`

Example

Sample Input:

```
1
I like hotdogs.
nobody likes to work.
everybody smiles.
what do I do?
who smiles?
what do you do?
does Joe smile?
do I like to work?
everybody hurts sometimes.
who walks there?
Michal walks there.
who walks there?
what does Michal do?
do you understand?
nobody walks there.
do you understand now?
bye!
```

Sample Output:

```
Dialogue #1:
what do I do?
you like hotdogs, don't like to work, and smile.

who smiles?
everybody smiles.

what do you do?
I don't like to work, and smile.

does Joe smile?
yes, Joe smiles.

do I like to work?
no, you don't like to work.

who walks there?
I don't know.

who walks there?
Michal walks there.

what does Michal do?
Michal doesn't like to work, smiles, hurts sometimes, and walks there.

do you understand?
maybe.

do you understand now?
I am abroad.

bye!
```

Added by: Adrian Kosowski
Date: 2004-05-09
Time limit: 2s
Source limit: 50000B
Languages: All
Resource: ACM Central European Programming Contest, Prague 2000

SPOJ Problem Set (classical)

14. I-Keyboard

Problem code: IKEYB

Most of you have probably tried to type anSMS message on the keypad of acellular phone. It is sometimes very annoying to write longer messages, because one key must be usually pressed several times to produce asingle letter. It is due to alow number of keys on the keypad. Typical phone has twelve keys only (and maybe some other control keys that are not used for typing). Moreover, only eight keys are used for typing 26letters of anEnglish alphabet. The standard assignment of letters on the keypad is shown in the left picture:

1	2 abc	3 def
4 ghi	5 jkl	6 mno
7 pqrs	8 tuv	9 wxyz
*	0 *space*	#

1	2 abcd	3 efg
4 hijk	5 lm	6 nopq
7 rs	8 tuv	9 wxyz
*	0 *space*	#

There are 3 or 4 letters assigned to each key. If you want the first letter of any group, you press that key once. If you want the second letter, you have to press the key twice. For other letters, the key must be pressed three or four times. The authors of the keyboard did not try to optimise the layout for minimal number of keystrokes. Instead, they preferred the even distribution of letters among the keys. Unfortunately, some letters are more frequent than others. Some of these frequent letters are placed on the third or even fourth place on the standard keyboard. For example, `S` is avery common letter in anEnglish alphabet, and we need four keystrokes to type it. If the assignment of characters was like in the right picture, the keyboard would be much more comfortable for typing average English texts.

ACM have decided to put anoptimised version of the keyboard on its new cellular phone. Now they need acomputer program that will find anoptimal layout for the given letter frequency. We need to preserve alphabetical order of letters, because the user would be confused if the letters were mixed. But we can assign any number of letters to asingle key.

Input

There is asingle positive integer *T* on the first line of input (equal to about 2000). It stands for the number of test cases to follow. Each test case begins with aline containing two integers *K*, *L* (1 <= *K* <= *L* <= 90) separated by asingle space. *K* is the number of keys, *L* is the number of letters to be mapped onto those keys. Then there are two lines. The first one contains exactly *K* characters each representing aname of one key. The second line contains exactly *L* characters representing names of letters of analphabet. Keys and letters are represented by digits, letters (which are case-sensitive), or any punctuation characters (ASCII code between 33 and 126 inclusively). No two keys have the same character, no two letters are the same. However, the name of aletter can be used also as aname for

akey.

After those two lines, there are exactly L lines each containing exactly one positive integer F_1, F_2, ... F_L. These numbers determine the frequency of every letter, starting with the first one and continuing with the others sequentially. The higher number means the more common letter. No frequency will be higher than 100000.

Output

Find anoptimal keyboard for each test case. Optimal keyboard is such that has the lowest "price" for typing average text. The *price* is determined as the sum of the prices of each letter. The price of aletter is aproduct of theletter frequency (F_i) and its position on the key. The order of letters cannot be changed, they must be grouped in the given order.

If there are more solutions with the same price, we will try to maximise the number of letters assigned to the last key, then to the one before the last one etc.

More formally, you are to find asequence P_1, P_2, ... P_L representing the position of every letter on aparticular key. The sequence must meet following conditions:

- $P_1 = 1$
- for each i>1, either $P_i = P_{i-1}+1$ or $P_i = 1$
- there are at most K numbers P_i such that $P_i = 1$
- the sum of products S_P = Sum[i=1..l] $F_i.P_i$ is minimal
- for any other sequence Q meeting these criteria and with the same sum $S_Q = S_P$, there exists such M, 1 <= M <= L that for any J, M<J <= L, $P_J = Q_J$, and $P_M > Q_M$.

The output for every test case must start with asingle line saying `Keypad #`I`:`, where I is asequential order of the test case, starting with 1. Then there must be exactly K lines, each representing one letter, in the same order that was used in input. Each line must contain the character representing the key, acolon, one space and alist of letters assigned to that particular key. Letters are not separated from each other.

Print one blank line after each test case, including the last one.

Example

Sample Input:

```
1
8 26
23456789
ABCDEFGHIJKLMNOPQRSTUVWXYZ
3371
589
1575
1614
6212
971
773
1904
2989
```

```
123
209
1588
1513
2996
3269
1080
121
2726
3083
4368
1334
518
752
427
733
871
```

Sample Output:

```
Keypad #1:
2: ABCD
3: EFG
4: HIJK
5: LM
6: NOPQ
7: RS
8: TUV
9: WXYZ
```

Warning: large Input/Output data, be careful with certain languages

---

Added by: Adrian Kosowski
Date: 2004-05-09
Time limit: 5s
Source limit: 50000B
Languages: All
Resource: ACM Central European Programming Contest, Prague 2000

SPOJ Problem Set (classical)

15. The Shortest Path

Problem code: SHPATH

You are given a list of cities. Each direct connection between two cities has its transportation cost (an integer bigger than 0). The goal is to find the paths of minimum cost between pairs of cities. Assume that the cost of each path (which is the sum of costs of all direct connections belongning to this path) is at most 200000. The name of a city is a string containing characters a,...,z and is at most 10 characters long.

Input

```
s [the number of tests <= 10]
n [the number of cities <= 10000]
NAME [city name]
p [the number of neighbours of city NAME]
nr cost [nr - index of a city connected to NAME (the index of the first city is 1)]
          [cost - the transportation cost]
r [the number of paths to find <= 100]
NAME1 NAME2 [NAME1 - source, NAME2 - destination]
[empty line separating the tests]
```

Output

```
cost [the minimum transportation cost from city NAME1 to city NAME2 (one per line)]
```

Example

```
Input:
1
4
gdansk
2
2 1
3 3
bydgoszcz
3
1 1
3 1
4 4
torun
3
1 3
2 1
4 1
warszawa
2
2 4
3 1
2
gdansk warszawa
bydgoszcz warszawa
```

```
Output:
3
2
```

Warning: large Input/Output data, be careful with certain languages

---

Added by: Darek Dereniowski
Date: 2004-05-10
Time limit: 5s
Source limit: 50000B
Languages: All
Resource: DASM Programming League 2003 (problemset 11)

SPOJ Problem Set (classical)

16. Sphere in a tetrahedron

Problem code: TETRA

Of course a Sphere Online Judge System is bound to have some tasks about spheres. So here is one. Given the lengths of the edges of a tetrahedron calculate the radius of a sphere inscribed in that tetrahedron (i.e. a sphere tangent to all the faces).

Input

Number N of test cases in a single line. (N <= 30) Each of the next N lines consists of 6 integer numbers -- the lengths of the edges of a tetrahedron separated by single spaces. The edges are not longer than 1000 and for the tetrahedron WXYZ, the order of the edges is: WX, WY, WZ, XY, XZ, YZ.

Output

N lines, each consisting of a real number given with four digits decimal precision equal to the radius of a sphere inscribed in the given tetrahedron.

Example

```
Input:
2
1 1 1 1 1 1
1000 999 998 5 5 6

Output:
0.2041
1.4189
```

Added by: Adam Dzedzej
Date: 2004-05-11
Time limit: 1s
Source limit:50000B
Languages: All

SPOJ Problem Set (classical)

17. The Bytelandian Cryptographer (Act I)

Problem code: CRYPTO1

Theinfamous Bytelandian Bit-eating Fanatic Organisation (BBFO for short)plans to launch an all-outdenial-of-service attack onthe Bytelandian McDecimal's fast food network by blocking the entrance to every restaurant with a camel (the purpose being to ridthe Organisationof unhealthy competition, obviously). In a sly and perfidious move, the head cryptographer ofBBFOdecided todisclose the date and time of theplanned attack to the management of McDecimal's, but in encrypted form (ha ha). He calculated the number of seconds from midnight 1970.01.01 GMT to the moment of attack, squared it, divided it by 4000000007 and sent the remainder by e-mail to McDecimal's. This made the original date impossible to decode.

Or did it?

* * *

You work as the head algorthimist at McDecimal's HQ and know nothing of what ishappening in Byteland.Things are not going well. You areplaying a quiet game ofhearts against your computerand wondering whyon earth Management are considering making you redundant. Suddenly, the CEObursts into your office, saying:

- Look here, young man[lady]! I have this numberandthose guysclaim it is supposed to be some date. I am giving you one second to tell me what it all means!

I am afraid you have no choice. You can't ask any further questions.
You just have to answer, now.

Input

The encrypted timestamp.

Output

The decryptedGMT time and date of attack, somewhere between 1970 and 2030,using standard 26 character formatting.

Example

Input:
```
1749870067
```

Output:
```
Sun Jun 13 16:20:39 2004
```

Added by:	Adrian Kosowski
Date:	2004-05-13
Time limit:	1s
Source limit:	10000B
Languages:	All
Resource:	;)

SPOJ Problem Set (classical)

18. The Bytelandian Cryptographer (Act II)

Problem code: CRYPTO2

Encouraged by his last successful exploit, the Bytelandian fanatic cryptographer impudently encrypted a three-digit number by subtracting 1 from it.

This time he has **really** overstepped the mark! Soldier, go and beat him, for Burger King & Country! Oh, and remember your good manners, use Brainf**k (no other language is allowed).

Input

An encrypted 3-digit positive integer.

Output

The decrypted value.

Example

Input:

```
699
```

Output:

```
700
```

Added by: Adrian Kosowski
Date: 2004-05-28
Time limit: 1s
Source limit:50000B
Languages: BF
Resource: Sometimes the simplest language is the most pleasing.

SPOJ Problem Set (classical)

19. The Bytelandian Cryptographer (Act III)

Problem code: CRYPTO3

The Bytelandian cryptographer acknowledged he was sorely beaten in Act 2. He renounced his own methods of encryption and decided to return to the classic techniques.

Not knowing what to do next, he went to the cinema to chew the problem over. To his surprise, he found that the cone containing pop-corn was in fact a rolled up page torn from the classic book, *RSA for newbies in 24 seconds*. The page in question contained the entire key-generating and encryption algorithm. Fascinated, he thought up two different prime numbers p and q, and calculated his own public key, and revealed the product p*q to the wide world. Then, he began work on his wicked scheme of encryption.

History repeats. Once more, you receive an encrypted message from the cryptographer. This time you know that without additional information you are beaten, so you decide to use the psychological approach. You phone the Bytelandian cryptographer, and ask him whether he couldn't give you a little hint. What you really want to know is the number u of positive integers which are smaller than p*q and have no common factors with p*q other than 1. But the cryptographer, sensing that this would allow you to decode the message right away, refuses to tell you this number. Eventually, after a lot of asking, he gives you a piece of utterly useless information: he tells you how many positive integers x cannot be represented in the form x=a*p+b*q, regardless of what non-negative integer values a and b assume.

You begin to wonder whether the information you received from the cryptographer is not by any chance enough to find the value of u.
Even if the only languages at your disposal are Brainfk and Intercal...**

Input

The number provided by the cryptographer (a positive integer of at most 99 decimal digits). The input ends with a new line symbol.

Output

The value of u.

Example

Input:
```
1
```

Output:
```
2
```

(This example is possible for p=2, q=3)

Added by:	Adrian Kosowski
Date:	2004-05-29
Time limit:	3s
Source limit:	50000B
Languages:	BF ICK
Resource:	Sadly, the ability to make a simple problem difficult to understand is seldom considered a talent.

SPOJ Problem Set (classical)

20. The Bytelandian Cryptographer (Act IV)

Problem code: CRYPTO4

The Bytelandian Cryptographer has been requested by the BBFO to put forward an ecryption scheme which would allow the BBFO to communicate with its foreign associates. After some intensive studies, he has decided upon the Vigenére cipher. Messages written using 26 upper case characters of the Latin alphabet: A, B, ..., Z which are interpreted as integers 0,1, ..., 25 respectively. The secret cypher for transmitting a message is known to both sides and consists of n integers k_1, k_2,...,k_n. Using this cypher, the i-th number x_i of the input message x is encrypted to the form of the i-th number of the output message y, as follows:
$y_i = (x_i + k_{1+((i-1) \bmod n)}) \bmod 26$.

You are trying to find out the content of a message transmitted by the BBFO. By a lucky stroke of fortune, your spys managed to intercept the message in both its plaintext and encrypted form (x and y respectively). Unfortunately, during their dramatic escape the files they were carrying where pierced by bullets and fragments of messages x and y were inadvertantly lost. Or were they? It is your task to reconstruct as much of message x as you possibly can.

Input

The first line of input contains a single integer t<=200 denoting the number of test cases. t test case descriptions follow.

For each test case, the first line contains one integer m which is some upper bound on the length of the cypher (1<=n<=m<=100000). The second line of input contains the original message x, while the third line contains the encrypted message y. The messages are expressed using characters 'A'-'Z' (interpreted as integers 0-25) and '*' (denoting a single character illegible due to damage). The total length of the input file is not more than 2MB.

Output

For each test case output a single line containing the original message x, with asterisks '*' in place of only those characters whose value cannot be determined.

Example

Input:

```
4
1
A*X*C
**CM*
4
*B***A
AAAAAA
6
*B***A
AAAAAA
```

```
4
*AA*******
AAAAAAAAAA
```

Output:

```
A*XHC
*BA*BA
*B***A
*AA**A****
```

Warning: large Input/Output data, be careful with certain languages.
The time limit is strict for this problem.

Added by: Konrad Piwakowski
Date: 2004-11-16
Time limit: 17s
Source limit: 50000B
Languages: All
Resource: DASM Programming League 2004 (problemset 3)

SPOJ Problem Set (classical)

22. Triangle From Centroid

Problem code: TRICENTR

Given the length of side a of a triangle and the distances from the centroid (the point of concurrence of the medians - red in the picture) to all sides: a, b and c, calculate this triangle's area and the distance (blue line) from the orthocenter (the point of concurrence of the heights - green in the picture) to the centroid.

[IMAGE]

Input

In the first line integer n - the number of test cases (equal to about 1000). The next n lines - 4 floating point values: the length of side a, and distances from the centroid to sides a, b and c.

Output

n lines consisting of 2 floating point values with 3 digits after the decimal point: the area of the triangle and the distance from the orthocenter to centroid.

Example

Input:
```
2
3.0 0.8660254038 0.8660254038 0.8660254038
657.8256599140 151.6154399062 213.5392629932 139.4878846649
```

Output:
```
3.897 0.000
149604.790 150.275
```

Added by: Patryk Pomykalski
Date: 2004-05-22
Time limit: 1s
Source limit: 50000B
Languages: All

SPOJ Problem Set (classical)

23. Pyramids

Problem code: PIR

Recently in Farland, a country in Asia, the famous scientist Mr. Log Archeo discovered ancient pyramids. But unlike those in Egypt and Central America, they have a triangular (not rectangular) foundation. That is, they are tetrahedrons in the mathematical sense. In order to find out some important facts about the early society of the country (it is widely believed that the pyramid sizes are closely connected with Farland's ancient calendar), Mr. Archeo needs to know the volume of the pyramids. Unluckily, he has reliable data about their edge lengths only. Please, help him!

Input

t [number of tests to follow] In each of the next t lines six positive integer numbers not exceeding 1000 separated by spaces (each number is one of the edge lengths of the pyramid ABCD). The order of the edges is the following: AB, AC, AD, BC, BD, CD.

Output

For each test output a real number - the volume, printed accurate to four digits after decimal point.

Example

```
Input:
2
1 1 1 1 1 1
1000 1000 1000 3 4 5

Output:
0.1179
1999.9937
```

Added by: Adam Dzedzej
Date: 2004-05-14
Time limit: 1s
Source limit: 10000B
Languages: All
Resource: ACM ICPC 2002-2003 NEERC, Northern Subregion

SPOJ Problem Set (classical)

24. Small factorials

Problem code: FCTRL2

You are asked to calculate factorials of some small positive integers.

Input

An integer t, 1<=t<=100, denoting the number of testcases, followed by t lines, each containing a single integer n, 1<=n<=100.

Output

For each integer n given at input, display a line with the value of n!

Example

Sample input:

```
4
1
2
5
3
```

Sample output:

```
1
2
120
6
```

Added by: Adrian Kosowski
Date: 2004-05-28
Time limit: 1s
Source limit: 2000B
Languages: All

SPOJ Problem Set (classical)

25. Pouring water

Problem code: POUR1

Given two vessels, one of which can accommodate *a* litres of water and the other - *b* litres of water, determine the number of steps required to obtain exactly *c* litres of water in one of the vessels.

At the beginning both vessels are empty. The following operations are counted as 'steps':

- emptying a vessel,
- filling a vessel,
- pouring water from one vessel to the other, without spilling, until one of the vessels is either full or empty.

Input

An integer *t*, 1<=*t*<=100, denoting the number of testcases, followed by *t* sets of input data, each consisting of three positive integers *a, b, c*, not larger than 40000, given in separate lines.

Output

For each set of input data, output the minimum number of steps required to obtain c litres, or -1 if this is impossible.

Example

Sample input:

```
2
5
2
3
2
3
4
```

Sample output:

```
2
-1
```

Added by: Adrian Kosowski
Date: 2004-05-31
Time limit: 1s
Source limit:50000B
Languages: All
Resource: An ancient problem, formulated in these words by Mr Tadeusz Ratajczak

SPOJ Problem Set (classical)

26. Build the Fence

Problem code: BSHEEP

At the beginning of spring all the sheep move to the higher pastures in the mountains. If there are thousands of them, it is well worthwhile gathering them together in one place. But sheep don't like to leave their grass-lands. Help the shepherd and build him a fence which would surround all the sheep. The fence should have the smallest possible length! Assume that sheep are negligibly small and that they are not moving. Sometimes a few sheep are standing in the same place. If there is only one sheep, it is probably dying, so no fence is needed at all...

Input

t [the number of tests <= 100]
[empty line]
n [the number of sheep <= 100000]
x_1 y_1 [coordinates of the first sheep]
...
x_n y_n
[integer coordinates from -10000 to 10000]
[empty line]
[other lists of sheep]
Text grouped in [ ] does not appear in the input file. Assume that sheep are numbered in the input order.

Output

o [length of circumference, 2 digits precision]
p1 p2 ... pk
[the sheep that are standing in the corners of the fence; the first one should be positioned bottommost and as far to the left as possible, the others ought to be written in anticlockwise order; ignore all sheep standing in the same place but the first to appear in the input file; the number of sheep should be the smallest possible]
[empty line]
[next solutions]

Example

```
Input:
8

5
0 0
0 5
10 5
3 3
10 0
```

```
1
0 0

3
0 0
1 0
2 0

4
0 0
0 0
0 1
1 0

3
0 0
0 1
1 0

6
0 0
-1 -1
1 1
2 2
3 3
4 4

2
10 0
0 0

7
-3 -4
2 -3
4 3
-4 2
0 5
2 -3
-1 4

Output:
30.00
1 5 3 2

0.00
1

4.00
1 3

3.41
1 4 3

3.41
1 3 2

14.14
2 6

20.00
```

```
2 1

26.98
1 2 3 5 4
```

Warning: large Input/Output data, be careful with certain languages

---

Added by: Michał Małafiejski
Date: 2004-06-01
Time limit: 7s
Source limit: 50000B
Languages: All
Resource: -

SPOJ Problem Set (classical)

27. Sorting Bank Accounts

Problem code: SBANK

In one of the internet banks thousands of operations are being performed every day. Since certain customers do business more actively than others, some of the bank accounts occur many times in the list of operations. Your task is to sort the bank account numbers in ascending order. If an account appears twice or more in the list, write the number of repetitions just after the account number. The format of accounts is as follows: **2** control digits, an **8**-digit code of the bank, **16** digits identifying the owner (written in groups of four digits), for example (at the end of each line there is exactly one space):

30 10103538 2222 1233 6160 0142

Banks are real-time institutions and they need FAST solutions. If you feel you can meet the challenge within a very stringent time limit, go ahead! A well designed sorting algorithm in a fast language is likely to succeed.

Input

t [the number of tests <= **5**]
n [the number of accounts<= **100 000**]
[list of accounts]
[empty line]
[next test cases]

Output

[sorted list of accounts with the number of repeated accounts]
[empty line]
[other results]

Example

```
Input:
2
6
03 10103538 2222 1233 6160 0142
03 10103538 2222 1233 6160 0141
30 10103538 2222 1233 6160 0141
30 10103538 2222 1233 6160 0142
30 10103538 2222 1233 6160 0141
30 10103538 2222 1233 6160 0142

5
30 10103538 2222 1233 6160 0144
30 10103538 2222 1233 6160 0142
30 10103538 2222 1233 6160 0145
```

```
30 10103538 2222 1233 6160 0146
30 10103538 2222 1233 6160 0143

Output:
03 10103538 2222 1233 6160 0141 1
03 10103538 2222 1233 6160 0142 1
30 10103538 2222 1233 6160 0141 2
30 10103538 2222 1233 6160 0142 2

30 10103538 2222 1233 6160 0142 1
30 10103538 2222 1233 6160 0143 1
30 10103538 2222 1233 6160 0144 1
30 10103538 2222 1233 6160 0145 1
30 10103538 2222 1233 6160 0146 1
```

Added by: Michał Małafiejski
Date: 2004-06-01
Time limit: 7s
Source limit: 50000B
Languages: All
Resource: -

SPOJ Problem Set (classical)

28. Help the Military Recruitment Office!

Problem code: HMRO

At the end of year 2004, the regional agencies of the Polish Military Recruitment Office (known as WKU in Polish) is sending a call to all boys born in 1984. Every recruit has his personal 11-digit identification number (PESEL, format: YYMMDDXXXXX, where YYMMDD is the date of birth, and XXXXX is a zero-padded integer smaller than 100000). Every agency of the Military Recruitment Office has its own code (MRO, format: a place code consisting of 3 upper case letters and a one-digit number). But this year the army underwent some reforms and not all boys at conscription age are going to be recruited. The list of closed down MRO points is as follows: the code of the closed down MRO is followed by the code of some other MRO, to which all the recruits are now going to be assigned. The list of recruits contains their PESEL codes. Your task is to prepare the complete list of recruits and determine the codes of their new MRO-s.

Input

```
s [the number of tests <= 10]
p [the number of boys at conscription age <= 100000]
PESEL and MRO code
z [the number of closed down MRO points <= 100000]
old_code new_code [old_code - the code of closed down MRO,
new_code - its new MRO code]
p [the number of recruits <= 100000]
PESEL [PESEL code of recruit]
[empty line]
[next tests]
```

Output

```
one PESEL and MRO code per line in the order of input
[empty line between tests]
[other results]
```

Example

```
Input:
1
4
84101011111 GDA1
84010122222 GDA2
84010233333 GDA2
84020255555 GDY1
1
GDA2 GDA1
3
84101011111
84010122222
84020255555
```

```
Output:
84101011111 GDA1
84010122222 GDA1
84020255555 GDY1
```

Warning: large Input/Output data, be careful with certain languages

---

Added by: Michał Małafiejski
Date: 2004-06-01
Time limit: 30s
Source limit:50000B
Languages: All
Resource: -

SPOJ Problem Set (classical)

29. Hash it!

Problem code: HASHIT

Your task is to calculate the result of the hashing process in a table of 101 elements, containing keys that are strings of length at most 15 letters (ASCII codes '*A*',...,'*z*'). Implement the following operations:

- find the index of the element defined by the key (ignore, if no such element),
- insert a new key into the table (ignore insertion of the key that already exists),
- delete a key from the table (without moving the others), by marking the position in table as *empty* (ignore non-existing keys in the table)

When performing find, insert and delete operations define the following function:
integer Hash(string key),
which for a string $key=a_1...a_n$ returns the value:
$Hash(key)=h(key) \bmod 101$, where
$h(key)=19 *(\text{ASCII}(a_1)*1+...+\text{ASCII}(a_n)*n)$.
Resolve collisions using the open addressing method, i.e. try to insert the key into the table at the first free position: $(Hash(key)+j^2+23*j) \bmod 101$, for $j=1,...,19$. After examining of at least 20 table entries, we assume that the insert operation cannot be performed.

Input

t [the number of test cases <= 100]
n_1 [the number of operations (one per line)[<= 1000]
ADD:string
[or]
DEL:string [other test cases, without empty lines betwee series]

Output

For every test case you have to create a new table, insert or delete keys, and write to the output:
the number of keys in the table [first line]
index:key [sorted by indices]

Example

```
Input:
1
11
ADD:marsz
ADD:marsz
ADD:Dabrowski
ADD:z
ADD:ziemii
ADD:wloskiej
ADD:do
```

```
ADD:Polski
DEL:od
DEL:do
DEL:wloskiej


Output:
5
34:Dabrowski
46:Polski
63:marsz
76:ziemii
96:z
```

Added by: Michał Małafiejski
Date: 2004-06-01
Time limit: 3s
Source limit:50000B
Languages: All
Resource: -

SPOJ Problem Set (classical)

30. Bytelandian Blingors Network

Problem code: BLINNET

We have discovered the fastest communication medium Bytelandian scientists announced, and they called it *blingors*. The blingors are incomparably better than other media known before. Many companies in Byteland started to build blingors networks, so the information society in the kingdom of Bytes is fact! The priority is to build the core of the blingors network, joinig main cities in the country. Assume there is some number of cities that will be connected at the beginning. The cost of building blingors connection between two cities depends on many elements, but it has been successfully estimated. Your task is to design the blingors network connections between some cities in this way that between any pair of cities is a communication route. The cost of this network should be as small as possible.

Remarks

- The name of the city is a string of at most 10 letters from *a,...,z*.
- The cost of the connection between two cities is a positive integer.
- The sum of all connections is not greater than 2^{32}-1.
- The number of cities is not greater than 10 000.

Input

```
s [number of test cases <= 10]
n [number of cities <= 10 000]
NAME [city name]
p [number of neigbouring cities to the city NAME]
neigh cost
     [neigh - the unique number of  city from the main list
      cost - the cost of building the blingors connection from NAME to neigh]

[empty line between test cases]
```

Output

[separate lines] *cost* [the minimum cost of building the blingors network]

Example

Input:

```
2

4
gdansk
2
2 1
3 3
bydgoszcz
3
1 1
```

```
3 1
4 4
torun
3
1 3
2 1
4 1
warszawa
2
2 4
3 1

3
ixowo
2
2 1
3 3
iyekowo
2
1 1
3 7
zetowo
2
1 3
2 7
```

Output:

```
3
4
```

Warning: large Input/Output data, be careful with certain languages

---

Added by: Łukasz Kuszner
Date: 2004-06-01
Time limit: 4s
Source limit: 50000B
Languages: All
Resource: PAL

SPOJ Problem Set (classical)

31. Fast Multiplication

Problem code: MUL

Multiply the given numbers.

Input

```
n [the number of multiplications <= 1000]
l1 l2 [numbers to multiply (at most 10000 decimal digits each)]

Text grouped in [ ] does not appear in the input file.
```

Output

```
The results of multiplications.
```

Example

```
Input:
5
4 2
123 43
324 342
0 12
9999 12345

Output:
8
5289
110808
0
123437655
```

Warning: large Input/Output data, be careful with certain languages

---

Added by: Darek Dereniowski
Date: 2004-06-01
Time limit: 2s
Source limit:50000B
Languages: All
Resource: PAL

SPOJ Problem Set (classical)

32. A Needle in the Haystack

Problem code: NHAY

Write a program that finds all occurences of a given pattern in a given input string. This is often referred to as finding a *needle* in a *haystack*.

The program has to detect **all** occurences of the needle in the haystack. It should take the needle and the haystack as input, and output the positions of each occurence, as shown below. The suggested implementation is the KMP algorithm, but this is not a requirement. However, a naive approach will probably exceed the time limit, whereas other algorithms are more complicated... The choice is yours.

Input

The input consists of a number of test cases. Each test case is composed of three lines, containing:

- the length of the needle,
- the needle itself,
- the haystack.

The length of the needle is only limited by the memory available to your program, so do not make any assumptions - instead, read the length and allocate memory as needed. The haystack is **not** limited in size, which implies that your program should not read the whole haystack at once. The KMP algorithm is stream-based, i.e. it processes the haystack character by character, so this is not a problem.

The test cases come one after another, each occupying three lines, with no additional space or line breaks in between.

Output

For each test case your program should output all positions of the needle's occurences within the haystack. If a match is found, the output should contain the position of the first character of the match. Characters in the haystack are numbered starting with zero.

For a given test case, the positions output should be sorted in ascending order, and each of these should be printed in a separate line. For two different test cases, the positions should be separated by an empty line.

Example

```
Sample input:2
na
bananobano
6
foobar
```

```
foo
9
foobarfoo
barfoobarfoobarfoobarfoobarfoo
```

Sample output:

```
2
4

3
9
15
21
```

Note the double empty line in the output, which means that no match was found for the second test case.

Warning: large Input/Output data, be careful with certain languages

---

Added by: Michał Małafiejski
Date: 2004-06-03
Time limit: 5s
Source limit: 50000B
Languages: All
Resource: the problem was phrased and test data was supplied by Mr. Maciej 'hawk' Jarzębski

SPOJ Problem Set (classical)

33. Trip

Problem code: TRIP

Alice and Bob want to go on holiday. Each of them has drawn up a list of cities to be visited in turn. A list may contain a city more than once. As they want to travel together, they have to agree upon a common route. Noone wants to change the order of the cities on his list or add other cities. Therefore they have no choice but to remove some cities from the list. Of course the common route is to involve as much sight-seeing in cities as possible. There are exactly 26 cities in the region. Therefore they are encoded on the lists as lower case letters from 'a' to 'z'.

Input

The first line of input contains a number T <= 10 that indicates the number of test cases to follow. Each test case consists of two lines; the first line is the list of Alice, the second line is the list of Bob. Each list consists of 1 to 80 lower case letters.

Output

The output for each test case should contain all different trips exactly once that meet the conditions described above. There is at least one such trip, but never more than 1000 different ones. You should order the trips in lexicographic order. Print one blank line between the output of different test cases.

Example

Input

```
1
abcabcaa
acbacba
```

Output

```
ababa
abaca
abcba
acaba
acaca
acbaa
acbca
```

Added by: Adrian Kuegel
Date: 2004-06-05
Time limit: 3s
Source limit:50000B
Languages: All
Resource: own problem, used in CEOI 2003

SPOJ Problem Set (classical)

34. Run Away

Problem code: RUNAWAY

One of the traps we will encounter in the Pyramid is located in the Large Room. A lot of small holes are drilled into the floor. They look completely harmless at the first sight. But when activated, they start to throw out very hot java, uh ... pardon, lava. Unfortunately, all known paths to the Center Room (where the Sarcophagus is) contain atrigger that activates the trap. The ACM were not able to avoid that. But they have carefully monitored the positions of all the holes. So it is important to find the place in the Large Room that has the maximal distance from all the holes. This place is the safest in the entire room and the archaeologist has to hide there.

Input

The input consists of *T* test cases. The number of them (*T*) is given on the first line of the input file. Each test case begins with a line containing three integers *X*, *Y*, *M* separated by space. The numbers satisfy conditions: *1 <= X,Y <=10000*, *1 <= M <= 1000*. The numbers *X* and *Y*indicate the dimensions of the Large Room which has arectangular shape. The number *M* stands for the number of holes. Then exactly *M* lines follow, each containing two integer numbers U_i and V_i ($0 <= U_i <= X$, $0 <= V_i <= Y$) indicating the coordinates of one hole. There may be several holes at the same position.

Output

Print exactly one line for each test case. The line should contain the sentence "`The safest point is (P, Q).`" where *P* and *Q*are the coordinates of the point in the room that has the maximum distance from the nearest hole, rounded to the nearest number with exactly one digit after the decimal point (0.05 rounds up to 0.1).

Example

```
Sample Input:
3
1000 50 1
10 10
100 100 4
10 10
10 90
90 10
90 90
3000 3000 4
1200 85
63 2500
2700 2650
2990 100

Sample output:
The safest point is (1000.0, 50.0).
The safest point is (50.0, 50.0).
The safest point is (1433.0, 1669.8).
```

Added by: Adrian Kosowski
Date: 2004-06-06
Time limit: 13s
Source limit: 50000B
Languages: All
Resource: ACM Central European Programming Contest, Prague 1999

SPOJ Problem Set (classical)

35. Equipment Box

Problem code: EQBOX

There is a large room in the Pyramid called *Room-of-No-Return*. Its floor is covered by rectangular tiles of equal size. The name of the room was chosen because of the very high number of traps and mechanisms in it. The ACM group has spent several years studying the secret plan of this room. It has made a clever plan to avoid all the traps. A specially trained mechanic was sent to deactivate the most feared trap called Shattered Bones. After deactivating the trap the mechanic had to escape from the room. It is very important to step on the center of the tiles only; he must not touch the edges. One wrong step and a large rock falls from the ceiling squashing the mechanic like a pancake. After deactivating the trap, he realized a horrible thing: the ACM plan did not take his equipment box into consideration. The box must be laid onto the ground because the mechanic must have both hands free to prevent contact with other traps. But when the box is laid on the ground, it could touch the line separating the tiles. And this is the main problem you are to solve.

Input

The input consists of *T* test cases (T is equal to about 10000). The number of them (*T*) is given on the first line of the input file. Each test case consists of a single line. The line contains exactly four integer numbers separated by spaces: *A*, *B*, *X* and *Y*. *A* and *B*indicate the dimensions of the tiles, *X* and *Y* are the dimensions of the equipment box (*1 <= A,B,X,Y <= 50000*).

Output

Your task is to determine whether it is possible to put the box on asingle tile -- that is, if the whole box fits on asingle tile without touching its border. If so, you are to print one line with the sentence "`Escape is possible.`". Otherwise print the sentence "`Box cannot be dropped.`".

Example

```
Sample Input:

2
10 10 8 8
8 8 10 10

Sample output:

Escape is possible.
Box cannot be dropped.
```

Warning: large Input/Output data, be careful with certain languages

---

Added by: Adrian Kosowski
Date: 2004-06-06
Time limit: 5s
Source limit: 50000B
Languages: All
Resource: ACM Central European Programming Contest, Prague 1999

SPOJ Problem Set (classical)

36. Secret Code

Problem code: CODE1

The Sarcophagus itself is locked by a secret numerical code. When somebody wants to open it, he must know the code and set it exactly on the top of the Sarcophagus. A very intricate mechanism then opens the cover. If an incorrect code is entered, the tickets inside would catch fire immediately and they would have been lost forever. The code (consisting of up to 100 integers) was hidden in the Alexandrian Library but unfortunately, as you probably know, the library burned down completely.

But an almost unknown archaeologist has obtained a copy of the code something during the 18th century. He was afraid that the code could get to the ''wrong people'' so he has encoded the numbers in a very special way. He took a random complex number B that was greater (in absolute value) than any of the encoded numbers. Then he counted the numbers as the digits of the system with basis B. That means the sequence of numbers $a_n, a_{n-1}, ..., a_1, a_0$ was encoded as the number $X = a_0 + a_1 B + a_2 B^2 + ... + a_n B^n$.

Your goal is to decrypt the secret code, i.e. to express a given number X in the number system to the base B. In other words, given the numbers X and Byou are to determine the ''digit'' a_0 through a_n.

Input

The input consists of T test cases (equal to about 100000). The number of them (T) is given on the first line of the input file. Each test case consists of one single line containing four integer numbers X_r, X_i, B_r, B_i ($|X_r|,|X_i| <= 1000000$, $|B_r|,|B_i| <= 16$). These numbers indicate the real and complex components of numbers X and B, i.e. $X = X_r + i.X_i$, $B = B_r + i.B_i$. B is the basis of the system ($|B| > 1$), X is the number you have to express.

Output

Your program must output a single line for each test case. The line should contain the ''digits'' $a_n, a_{n-1}, ..., a_1, a_0$, separated by commas. The following conditions must be satisfied:

- for all i in $\{0, 1, 2, ...n\}$: $0 <= a_i < |B|$
- $X = a_0 + a_1 B + a_2 B^2 + ... + a_n B^n$
- if $n > 0$ then $a_n <> 0$
- $n <= 100$

If there are no numbers meeting these criteria, output the sentence "`The code cannot be decrypted.`". If there are more possibilities, print any of them.

Example

```
Sample Input

4
-935 2475 -11 -15
1 0 -3 -2
93 16 3 2
191 -192 11 -12

Sample output:

8,11,18
1
The code cannot be decrypted.
16,15
```

Warning: large Input/Output data, be careful with certain languages

---

Added by: Adrian Kosowski
Date: 2004-06-06
Time limit: 5s
Source limit: 50000B
Languages: All
Resource: ACM Central European Programming Contest, Prague 1999

SPOJ Problem Set (classical)

37. The Proper Key

Problem code: PROPKEY

Many people think that Tetris was invented by two Russian programmers. But that is not the whole truth. The idea of the game is very old -- even the Egyptians had something similar. But they did not use it as a game. Instead, it was used as avery complicated lock. The lock was made of wood and consisted of a large number of square fields, laid out in regular rows and columns. Each field was either completely filled with wood, or empty. The key for this lock was two-dimensional and it was made by joining square parts of the same size as the fields of the lock. So they had a 2D lock and 2D key that could be inserted into the lock from the top. The key was designed so that it was not possible to move it upwards. It could only fall down and it could slide sideways -- exactly like in a Tetris game. The only difference is that the key could not be rotated. Rotation in Tetris is really a Russian invention.

The entry gate into the Pyramid has such a lock. The ACM archaeologists have found several keys and one of them belongs to the lock with a very high probability. Now they need to try them out and find which one to use. Because it is too time-consuming to try all of them, it is better to begin with those keys that may be inserted deeper into the lock. Your program should be able to determine how deep a given key can be inserted into a given lock.

Input

The input consists of *T* test cases. The number of them (*T*) is given on the first line of the input file. Each test case begins with a line containing two integers *R* and *C* (*1 <= R,C <= 100*) indicating the key size. Then exactly *R* rows follow, each containing *C* characters. Each character is either a hash mark (`#`) or a period (`.`). A hash mark represents one square field made of wood; a period is an empty field. The wooden fields are always connected, i.e. the whole key is made of one piece. Moreover, the key remains connected even if we cut off arbitrary number of rows from its top. There is always at least one non-empty field in the top-most and bottom-most rows and the left-most and right-most columns.

After the key description, there is a line containing two integers *D* and *W* (*1 <= D <= 10000*, *1 <= W <= 1000*). The number *W* is the lock width, and *D* is its depth. The next *D* lines contain *W* characters each. The character may be either a hash mark (representing the wood) or a period (the free space).

Output

Your program should print one line of output for each test case. The line should contain the statement "`The key falls to depth X.`". Replace *X* with the maximum depth to which the key can be inserted by moving it down and sliding it to the left or right only. The depth is measured as the distance between the bottom side of the key and the top side of the lock. If it is possible to move the key through the whole lock and take it away at the bottom side, output the sentence "`The key can fall through.`".

Example

```
Sample Input:

4
2 4
#.##
###.
3 6
#....#
#....#
#..###
2 3
##.
.##
2 7
#.#.#.#
.#.#.#.
1 1
#
1 10
###....###
3 2
##
.#
.#
1 5
#.#.#


Sample output:

The key falls to depth 2.
The key falls to depth 0.
The key can fall through.
The key falls to depth 2.
```

Warning: large Input/Output data, be careful with certain languages

---

Added by: Adrian Kosowski
Date: 2004-06-06
Time limit: 5s
Source limit: 50000B
Languages: All
Resource: ACM Central European Programming Contest, Prague 1999

SPOJ Problem Set (classical)

38. Labyrinth

Problem code: LABYR1

The northern part of the Pyramid contains a very large and complicated labyrinth. The labyrinth is divided into square blocks, each of them either filled by rock, or free. There is also a little hook on the floor in the center of every free block. The ACM have found that two of the hooks must be connected by a rope that runs through the hooks in every block on the path between the connected ones. When the rope is fastened, a secret door opens. The problem is that we do not know which hooks to connect. That means also that the neccessary length of the rope is unknown. Your task is to determine the maximum length of the rope we could need for a given labyrinth.

Input

The input consists of *T* test cases. The number of them (*T*) is given on the first line of the input file. Each test case begins with a line containing two integers *C* and *R* (*3 <= C,R <= 1000*) indicating the number of columns and rows. Then exactly *R* lines follow, each containing *C* characters. These characters specify the labyrinth. Each of them is either a hash mark (`#`) or a period (`.`). Hash marks represent rocks, periods are free blocks. It is possible to walk between neighbouring blocks only, where neighbouring blocks are blocks sharing acommon side. We cannot walk diagonally and we cannot step out of the labyrinth.

The labyrinth is designed in such a way that there is exactly one path between any two free blocks. Consequently, if we find the proper hooks to connect, it is easy to find the right path connecting them.

Output

Your program must print exactly one line of output for each test case. The line must contain the sentence "`Maximum rope length is X.`" where *X*is the length of the longest path between any two free blocks, measured in blocks.

Example

```
Sample Input:

2
3 3
###
#.#
###
7 6
#######
#.#.###
#.#.###
#.#.#.#
#.....#
#######
```

```
Sample output:

Maximum rope length is 0.
Maximum rope length is 8.
```

Warning: large Input/Output data, be careful with certain languages

---

Added by: Adrian Kosowski
Date: 2004-06-06
Time limit: 5s
Source limit: 50000B
Languages: All
Resource: ACM Central European Programming Contest, Prague 1999

SPOJ Problem Set (classical)

39. Piggy-Bank

Problem code: PIGBANK

Before ACM can do anything, abudget must be prepared and the necessary financial support obtained. The main income for this action comes from Irreversibly Bound Money (IBM). The idea behind is simple. Whenever some ACM member has any small money, he takes all the coins and throws them into a piggy-bank. You know that this process is irreversible, the coins cannot be removed without breaking the pig. After a sufficiently long time, there should be enough cash in the piggy-bank to pay everything that needs to be paid.

But there is a big problem with piggy-banks. It is not possible to determine how much money is inside. So we might break the pig into pieces only to find out that there is not enough money. Clearly, we want to avoid this unpleasant situation. The only possibility is to weigh the piggy-bank and try to guess how many coins are inside. Assume that we are able to determine the weight of the pig exactly and that we know the weights of all coins of a given currency. Then there is some minimum amount of money in the piggy-bank that we can guarantee. Your task is to find out this worst case and determine the minimum amount of cash inside the piggy-bank. We need your help. No more prematurely broken pigs!

Input

The input consists of *T* test cases. The number of them (*T*) is given on the first line of the input file. Each test case begins with a line containing two integers *E* and *F*. They indicate the weight of an empty pig and of the pig filled with coins. Both weights are given in grams. No pig will weigh more than 10 kg, that means *1 <= E <= F <= 10000*. On the second line of each test case, there is an integer number *N* (*1 <= N <= 500*) that gives the number of various coins used in the given currency. Following this are exactly *N* lines, each specifying one coin type. These lines contain two integers each, *P*and *W* (*1 <= P <= 50000*, *1 <= W <=10000*). *P* is the value of the coin in monetary units, *W* is it's weight in grams.

Output

Print exactly one line of output for each test case. The line must contain thesentence "`The minimum amount of money in the piggy-bank is X.`" where *X* is the minimum amount of money that can be achieved using coins with the given total weight. If the weight cannot be reached exactly, print aline "`This is impossible.`".

Example

```
Sample Input:

3
10 110
2
1 1
30 50
10 110
```

```
2
1 1
50 30
1 6
2
10 3
20 4

Sample output:

The minimum amount of money in the piggy-bank is 60.
The minimum amount of money in the piggy-bank is 100.
This is impossible.
```

Added by: Adrian Kosowski
Date: 2004-06-06
Time limit: 5s
Source limit: 50000B
Languages: All
Resource: ACM Central European Programming Contest, Prague 1999

SPOJ Problem Set (classical)

40. Lifting the Stone

Problem code: STONE

There are many secret openings in the floor which are covered by a big heavy stone. When the stone is lifted up, a special mechanism detects this and activates poisoned arrows that are shot near the opening. The only possibility is to lift the stone very slowly and carefully. The ACM team must connect a rope to the stone and then lift it using a pulley. Moreover, the stone must be lifted all at once; no side can rise before another. So it is very important to find the centre of gravity and connect the rope exactly to that point. The stone has a polygonal shape and its height is the same throughout the whole polygonal area. Your task is to find the centre of gravity for the given polygon.

Input

The input consists of *T* test cases (equal to about 500). The number of them (*T*) is given on the first line of the input file. Each test case begins with a line containing a single integer *N* (*3 <= N <= 1000000*) indicating the number of points that form the polygon. This is followed by *N* lines, each containing two integers X_i and Y_i ($|X_i|, |Y_i| <= 20000$). These numbers are the coordinates of the *i*-th point. When we connect the points in the given order, we get a polygon. You may assume that the edges never touch each other (except the neighbouring ones) and that they never cross. The area of the polygon is never zero, i.e. it cannot collapse into a single line.

Output

Print exactly one line for each test case. The line should contain exactly two numbers separated by one space. These numbers are the coordinates of the centre of gravity. Round the coordinates to the nearest number with exactly two digits after the decimal point (0.005 rounds up to 0.01). Note that the centre of gravity may be outside the polygon, if its shape is not convex. If there is such a case in the input data, print the centre anyway.

Example

```
Sample Input:
2
4
5 0
0 5
-5 0
0 -5
4
1 1
11 1
11 11
1 11

Sample output:

0.00 0.00
6.00 6.00
```

Added by: Adrian Kosowski
Date: 2004-06-06
Time limit: 5s
Source limit: 50000B
Languages: All
Resource: ACM Central European Programming Contest, Prague 1999

SPOJ Problem Set (classical)

41. Play on Words

Problem code: WORDS1

Some of the secret doors contain avery interesting word puzzle. The team of archaeologists has to solve it to open that doors. Because there is no other way to open the doors, the puzzle is very important for us.

There is alarge number of magnetic plates on every door. Every plate has one word written on it. The plates must be arranged into a sequence in such a way that every word begins with the same letter as the previous word ends. For example, the word ‘‘acm’’ can be followed by the word ‘‘motorola’’. Your task is to write a computer program that will read the list of words and determine whether it is possible to arrange all of the plates in asequence (according to the given rule) and consequently to open the door.

Input

The input consists of *T* test cases. The number of them (*T*, equal to about 500) is given on the first line of the input file. Each test case begins with a line containing asingle integer number *N* that indicates the number of plates (*1 <= N <= 100000*). Then exactly *N*lines follow, each containing asingle word. Each word contains at least two and at most 1000 lowercase characters, that means only letters ’a’ through ’z’ will appear in the word. The same word may appear several times in the list.

Output

Your program has to determine whether it is possible to arrange all the plates in asequence such that the first letter of each word is equal to the last letter of the previous word. All the plates from the list must be used, each exactly once. The words mentioned several times must be used that number of times.

If there exists such an ordering of plates, your program should print thesentence "`Ordering is possible.`". Otherwise, output thesentence "`The door cannot be opened.`".

Example

```
Sample input:

3
2
acm
ibm
3
acm
malform
mouse
2
ok
ok
```

Sample output:

```
The door cannot be opened.
Ordering is possible.
The door cannot be opened.
```

Warning: large Input/Output data, be careful with certain languages

---

Added by: Adrian Kosowski
Date: 2004-06-06
Time limit: 7s
Source limit: 50000B
Languages: All
Resource: ACM Central European Programming Contest, Prague 1999

SPOJ Problem Set (classical)

42. Adding Reversed Numbers

Problem code: ADDREV

The Antique Comedians of Malidinesia prefer comedies to tragedies. Unfortunately, most of the ancient plays are tragedies. Therefore the dramatic advisor of ACM has decided to transfigure some tragedies into comedies. Obviously, this work is very hard because the basic sense of the play must be kept intact, although all the things change to their opposites. For example the numbers: if any number appears in the tragedy, it must be converted to its reversed form before being accepted into the comedy play.

Reversed number is a number written in arabic numerals but the order of digits is reversed. The first digit becomes last and vice versa. For example, if the main hero had 1245 strawberries in the tragedy, he has 5421 of them now. Note that all the leading zeros are omitted. That means if the number ends with a zero, the zero is lost by reversing (e.g. 1200 gives 21). Also note that the reversed number never has any trailing zeros.

ACM needs to calculate with reversed numbers. Your task is to add two reversed numbers and output their reversed sum. Of course, the result is not unique because any particular number is a reversed form of several numbers (e.g. 21 could be 12, 120 or 1200 before reversing). Thus we must assume that no zeros were lost by reversing (e.g. assume that the original number was 12).

Input

The input consists of *N* cases (equal to about 10000). The first line of the input contains only positive integer *N*. Then follow the cases. Each case consists of exactly one line with two positive integers separated by space. These are the reversed numbers you are to add.

Output

For each case, print exactly one line containing only one integer - the reversed sum of two reversed numbers. Omit any leading zeros in the output.

Example

```
Sample input:

3
24 1
4358 754
305 794

Sample output:

34
1998
1
```

Added by:	Adrian Kosowski
Date:	2004-06-06
Time limit:	5s
Source limit:	50000B
Languages:	All
Resource:	ACM Central European Programming Contest, Prague 1998

SPOJ Problem Set (classical)

43. Copying Books

Problem code: BOOKS1

Before the invention of book-printing, it was very hard to make a copy of a book. All the contents had to be re-written by hand by so called *scribers*. The scriber had been given a book and after several months he finished its copy. One of the most famous scribers lived in the 15th century and his name was Xaverius Endricus Remius Ontius Xendrianus (*Xerox*). Anyway, the work was very annoying and boring. And the only way to speed it up was to hire more scribers.

Once upon a time, there was a theater ensemble that wanted to play famous Antique Tragedies. The scripts of these plays were divided into many books and actors needed more copies of them, of course. So they hired many scribers to make copies of these books. Imagine you have m books (numbered $1, 2 \ldots m$) that may have different number of pages ($p_1, p_2 \ldots p_m$) and you want to make one copy of each of them. Your task is to divide these books among k scribes, $k <= m$. Each book can be assigned to a single scriber only, and every scriber must get a continuous sequence of books. That means, there exists an increasing succession of numbers $0 = b_0 < b_1 < b_2, \ldots < b_{k-1} <= b_k = m$ such that i-th scriber gets a sequence of books with numbers between $b_{i-1}+1$ and b_i. The time needed to make a copy of all the books is determined by the scriber who was assigned the most work. Therefore, our goal is to minimize the maximum number of pages assigned to a single scriber. Your task is to find the optimal assignment.

Input

The input consists of N cases (equal to about 200). The first line of the input contains only positive integer N. Then follow the cases. Each case consists of exactly two lines. At the first line, there are two integers m and k, $1 <= k <= m <= 500$. At the second line, there are integers $p_1, p_2, \ldots p_m$ separated by spaces. All these values are positive and less than 10000000.

Output

For each case, print exactly one line. The line must contain the input succession $p_1, p_2, \ldots p_m$ divided into exactly k parts such that the maximum sum of a single part should be as small as possible. Use the slash character ('/') to separate the parts. There must be exactly one space character between any two successive numbers and between the number and the slash.

If there is more than one solution, print the one that minimizes the work assigned to the first scriber, then to the second scriber etc. But each scriber must be assigned at least one book.

Example

```
Sample input:

2
9 3
100 200 300 400 500 600 700 800 900
5 4
```

```
100 100 100 100 100

Sample output:

100 200 300 400 500 / 600 700 / 800 900
100 / 100 / 100 / 100 100
```

Added by: Adrian Kosowski
Date: 2004-06-06
Time limit: 5s
Source limit: 50000B
Languages: All
Resource: ACM Central European Programming Contest, Prague 1998

SPOJ Problem Set (classical)

44. Substitution Cipher

Problem code: SCYPHER

Antique Comedians of Malidinesia would like to play a new discovered comedy of Aristofanes. Putting it on a stage should be a big surprise for the audience so all the preparations must be kept absolutely secret. The ACM director suspects one of his competitors of reading his correspondece. To prevent other companies from revealing his secret, he decided to use a substitution cipher in all the letters mentioning the new play.

Substitution cipher is defined by a substitution table assigning each character of the substitution alphabet another character of the same alphabet. The assignment is a bijection (to each character exactly one character is assigned -- not neccessary different). The director is afraid of disclosing the substitution table and therefore he changes it frequently. After each change he chooses a few words from a dictionary by random, encrypts them and sends them together with an encrypted message. The plain (i.e. non-encrypted) words are sent by a secure channel, not by mail. The recipient of the message can then compare plain and encrypted words and create a new substitution table.

Unfortunately, one of the ACM cipher specialists have found that this system is sometimes insecure. Some messages can be decrypted by the rival company even without knowing the plain words. The reason is that when the director chooses the words from the dictionary and encrypts them, he never changes their order (the words in the dictionary are lexicographically sorted). String $a_1 a_2 \ldots a_p$ is lexicografically smaller than $b_1 b_2 \ldots b_q$ if there exists an integer i, $i <= p$, $i <= q$, such that $a_j = b_j$ for each j, $1 <= j < i$ and $a_i < b_i$.

The director is interested in which of his messages could be read by the rival company. You are to write a program to determine that.

Input

The input consists of *N* cases (equal to about 1000). The first line of the input contains only positive integer *N*. Then follow the cases. The first line of each case contains only two positive integers *A*, *1 <= A <= 26*, and *K*, separated by space. *A* determines the size of the substitution alphabet (the substitution alphabet consists of the first *A* lowercase letters of the english alphabet (a--z) and *K* is the number of encrypted words. The plain words contain only the letters of the substitution alphabet. The plain message can contain any symbol, but only the letters of the substitution alphabet are encrypted. Then follow *K* lines, each containing exactly one encrypted word. At the next line is encrypted message.

Output

For each case, print exactly one line. If it is possible to decrypt the message uniquely, print the decrypted message. Otherwise, print the sentence '`Message cannot be decrypted.`'.

Example

```
Sample input:

2
5 6
cebdbac
cac
ecd
dca
aba
bac
cedab
4 4
cca
cad
aac
bca
bdac

Sample output:

abcde
Message cannot be decrypted.
```

Warning: large Input/Output data, be careful with certain languages

---

Added by: Adrian Kosowski
Date: 2004-06-06
Time limit: 5s
Source limit: 50000B
Languages: All
Resource: ACM Central European Programming Contest, Prague 1998

SPOJ Problem Set (classical)

45. Commedia dell Arte

Problem code: COMMEDIA

So called *commedia dell' arte* is a theater genre first played in Italy at the beginning of the sixteenth century. It was inspired with the Roman Theater. The play had no fixed script and the actors (also called *performers*) had to improvise a lot. There were only a simple directions by the author like "enter the stage and make something funny" or "everyone comes on stage and everything is resolved happily". You can see it might be very interesting to play the commedia dell' arte. Therefore the ACM want to put a new play on a stage, which was completely unknown before. The main hero has a puzzle that takes a very important role in the play and gives an opportunity of many improvisations. The puzzle is the worldwide known *Lloyd's Fifteen Puzzle*. ACM wants to make the play more interesting so they want to replace the "standard" puzzle with a three-dimensional one. The puzzle consists of a cube containing M^3 slots. Each slot except one contains a cubic tile (one position is free). The tiles are numbered from *1* to M^3-1. The goal of the puzzle is to get the original ordering of the tiles after they have been randomly reshuffled. The only allowed moves are sliding a neighbouring tile into the free position along one of the three principal directions. Original configuration is when slot with coordinates *(x,y,z)* from $\{0,...,M-1\}^3$ contains tile number $z.M^2+y.M+x+1$ and slot *(M-1,M-1,M-1)* is free.

You are to write a program to determine whether it is possible to solve the puzzle or not.

Input

The input consists of *N* cases. The first line of the input contains only positive integer *N*. Then follow the cases. The first line of each case contains only one integer *M*, $1 <= M <= 100$. It is the size of 3D puzzle cube. Then follow *M* lines, each contains exactly M^2 numbers on the tiles for one layer. First is the layer on the top of the cube and the last one on the bottom. In each layer numbers are arranged from the left top corner linewise to the right bottom corner of the layer. In other words, slot with coordinates *(x,y,z)* is described by the *(x+M.y+1)*-th number on the *(z+1)*-th line. Numbers are separated by space. Number *0* means free position.

Output

For each case, print exactly one line. If the original configuration can be reached by sliding the tiles, print the sentence '`Puzzle can be solved.`'. Otherwise, print the sentence '`Puzzle is unsolvable.`'.

Example

```
Sample input:

2
2
1 2 3 4
5 7 6 0
2
2 1 3 5
```

```
4 6 0 7

Sample output:

Puzzle is unsolvable.
Puzzle can be solved.
```

Warning: large Input/Output data, be careful with certain languages

---

Added by: Adrian Kosowski
Date: 2004-06-06
Time limit: 5s
Source limit: 50000B
Languages: All
Resource: ACM Central European Programming Contest, Prague 1998

SPOJ Problem Set (classical)

47. Skyscraper Floors

Problem code: SCRAPER

What a great idea it is to build skyscrapers! Using not too large area of land, which is very expensive in many cities today, the skyscrapers offer an extremely large utility area for flats or offices. The only disadvantage is that it takes too long to get to the upper floors. Of course these skyscrapers have to be equiped not only with a stairway but also with several elevators. But even using ordinary elevators is very slow. Just imagine you want to get from the very top floor to the base floor and many other people on other floors want the same. As a result the elevator stops on almost every floor and since its capacity is limited and the elevator is already full from the upper floors, most stops are useless and just cause a delay. If there are more elevators in the skyscrapers, this problem is a little bit eliminated but still not completely. Most people just press all the buttons of all the elevators and then take the first one so that all elevators will stop on the floor anyway.

However, the solution exists as we shall see. The Antique Comedians of Midilesia headquarters reside in a skyscraper with a very special elevator system. The elevators do not stop on every floor but only on every *X*-th floor. Moreover each elevator can go just to a certain floor *Y* (called starting floor) and cannot go any lower. There is one high-capacity elevator which can stop on every elevator's starting floor.

The ACM has a big problem. The headquarters should be moved to another office this week, possibly on a different floor. Unfortunately, the high-capacity elevator is out of order right now so it is not always possible to go to the base floor. One piece of furniture cannot be moved using the stairway because it is too large to pass through the stairway door. You are to write a program that decides whether it is possible to move a piece of furniture from the original office to the other.

Input

The input consists of *N* cases (equal to about 2000). The first line contains only one positive integer *N*. Then follow the cases. Each case starts with a line containing four integers *F*, *E*, *A*, *B*, where *F*, *1 <= F < 50000000* determines the number of floors in the skyscraper (this means that there are floors *0* to *F-1*), *E*, *0 < E < 100* is the number of elevators and *A*, *B*, *0 <= A,B < F* are numbers of the two floors between which the piece of furniture should be moved. Then follow *E* lines. Each of them contains description of one elevator. There are exactly two integers *X* and *Y*, *X > 0*, *Y >= 0* at each line. *Y* determines, that the elevator starts on the *Y*-th floor and *X* determines, that it stops on every *X*-th floor, eg. for *X = 3*, *Y = 7* the elevator stops on floors 7, 10, 13, 16, etc.).

Output

For each case, print exactly one line. If floor *B* is reachable from floor *A* not using the stairway, print the sentence '`It is possible to move the furniture.`', otherwise print '`The furniture cannot be moved.`'.

Example

```
Sample input:

2
22 4 0 6
3 2
4 7
13 6
10 0
1000 2 500 777
2 0
2 1

Sample output:

It is possible to move the furniture.
The furniture cannot be moved.
```

Warning: large Input/Output data, be careful with certain languages

---

Added by: Adrian Kosowski
Date: 2004-06-06
Time limit: 13s
Source limit: 50000B
Languages: All
Resource: ACM Central European Programming Contest, Prague 1998

SPOJ Problem Set (classical)

48. Glass Beads

Problem code: BEADS

Once upon a time there was a famous actress. As you may expect, she played mostly Antique Comedies most of all. All the people loved her. But she was not interested in the crowds. Her big hobby were beads of any kind. Many bead makers were working for her and they manufactured new necklaces and bracelets every day. One day she called her main *Inspector of Bead Makers* (*IBM*) and told him she wanted a very long and special necklace.

The necklace should be made of glass beads of different sizes connected to each other but without any thread running through the beads, so that means the beads can be disconnected at any point. The actress chose the succession of beads she wants to have and the IBM promised to make the necklace. But then he realized a problem. The joint between two neighbouring beads is not very robust so it is possible that the necklace will get torn by its own weight. The situation becomes even worse when the necklace is disjoined. Moreover, the point of disconnection is very important. If there are small beads at the beginning, the possibility of tearing is much higher than if there were large beads. IBM wants to test the robustness of a necklace so he needs a program that will be able to determine the worst possible point of disjoining the beads.

The description of the necklace is a string $A = a_1 a_2 \ldots a_m$ specifying sizes of the particular beads, where the last character a_m is considered to precede character a_1 in circular fashion.

The disjoint point i is said to be worse than the disjoint point j if and only if the string $a_i a_{i+1} \ldots a_n a_1 \ldots a_{i-1}$ is lexicografically smaller than the string $a_j a_{j+1} \ldots a_n a_1 \ldots a_{j-1}$. String $a_1 a_2 \ldots a_n$ is lexicografically smaller than the string $b_1 b_2 \ldots b_n$ if and only if there exists an integer i, $i <= n$, so that $a_j = b_j$, for each j, $1 <= j < i$ and $a_i < b_i$.

Input

The input consists of N cases. The first line of the input contains only positive integer N. Then follow the cases. Each case consists of exactly one line containing necklace description. Maximal length of each description is 10000 characters. Each bead is represented by a lower-case character of the english alphabet (a--z), where $a < b \ldots z$.

Output

For each case, print exactly one line containing only one integer -- number of the bead which is the first at the worst possible disjoining, i.e. such i, that the string $A[i]$ is lexicographically smallest among all the n possible disjoinings of a necklace. If there are more than one solution, print the one with the lowest i.

Example

```
Sample input:

4
helloworld
amandamanda
dontcallmebfu
aaabaaa

Sample output:

10
11
6
5
```

Added by:	Adrian Kosowski
Date:	2004-06-06
Time limit:	5s
Source limit:	50000B
Languages:	All
Resource:	ACM Central European Programming Contest, Prague 1998

SPOJ Problem Set (classical)

49. Hares and Foxes

Problem code: HAREFOX

The Antique Comedians of Malidinesia play an interesting comedy where many animals occur. Because they want their plays to be as true as possible, a specialist studies the behaviour of various animals. Recently, he is interested in a binary dynamic ecological system hares-foxes (SHF). As a part of this project, you are asked to design and implement intelligent automatic target evaluation simulator (IATES) for this system. The behaviour of the SHF follows so called *standard model*, described by the following set of difference equations.

$$h_{y+1} = a.h_y - b.f_y$$
$$f_{y+1} = c.f_y + d.h_y$$

where h_y resp. f_y represent the difference of the number of hares resp. foxes in year y and the reference count determined at the beginning of the experiment. The units of h_y and f_y are unknown. Therefore, h_y and f_y are to be treated as real numbers. Your task is to write a program to determine the long term evolution of SHF.

Input

The input consists of *N* cases (equal to about 5000). The first line of the input contains only positive integer *N*. Then follow the cases. Each case consists of six real numbers *a, b, c, d,* h_{1998} and f_{1998}, written in this order on three lines, two numbers per line, separated by one or more spaces. The numbers are given in the classical format, i.e. optional sign, sequence of digits, optional dot and optional sequence of digits. The text form of a number does not exceed 10 characters. Each case is followed by one empty line.

Output

For each case, print one of the following sentences:

- '`Ecological balance will develop.`' - if after sufficiently long time the population of both hares and foxes approaches the reference count with an arbitrary a priori given precision, i.e. $lim\ h_y = 0$ and $lim\ f_y = 0$.
- '`Hares will die out while foxes will overgrow.`' - if after sufficiently long time the population of hares resp. foxes falls under resp. exceeds any a priori given threshold, i.e. $lim\ h_y = -infinity$ and $lim\ f_y = +infinity$.
- '`Hares will overgrow while foxes will die out.`' - if after sufficiently long time the population of foxes resp. hares falls under resp. exceeds any a priori given threshold, i.e. $lim\ h_y = +infinity$ and $lim\ f_y = -infinity$.
- '`Both hares and foxes will die out.`' - if after sufficiently long time the population of both hares and foxes falls under any a priori given threshold, i.e. $lim\ h_y = -infinity$ and $lim\ f_y = -infinity$.

- ’`Both hares and foxes will overgrow.`’ - if after sufficiently long time the population of both hares and foxes exceeds any a priori given threshold, i.e. $lim\ h_y = +infinity$ and $lim\ f_y = +infinity$.
- ’`Chaos will develop.`’ - if none of the above mentioned description fits.

Example

```
Sample input:

2
2 0.5
0.5 0.6
2 3

0.1 1
2 0.1
1 1

Sample output:

Both hares and foxes will overgrow.
Hares will die out while foxes will overgrow.
```

Added by: Adrian Kosowski
Date: 2004-06-06
Time limit: 5s
Source limit: 50000B
Languages: All
Resource: ACM Central European Programming Contest, Prague 1998

SPOJ Problem Set (classical)

50. Invitation Cards

Problem code: INCARDS

In the age of television, not many people attend theater performances. Antique Comedians of Malidinesia are aware of this fact. They want to propagate theater and, most of all, Antique Comedies. They have printed invitation cards with all the necessary information and with the programme. A lot of students were hired to distribute these invitations among the people. Each student volunteer has assigned exactly one bus stop and he or she stays there the whole day and gives invitation to people travelling by bus. A special course was taken where students learned how to influence people and what is the difference between influencing and robbery.

The transport system is very special: all lines are unidirectional and connect exactly two stops. Buses leave the originating stop with passangers each half an hour. After reaching the destination stop they return empty to the originating stop, where they wait until the next full half an hour, e.g. X:00 or X:30, where 'X' denotes the hour. The fee for transport between two stops is given by special tables and is payable on the spot. The lines are planned in such a way, that each round trip (i.e. a journey starting and finishing at the same stop) passes through a *Central Checkpoint Stop* (CCS) where each passenger has to pass a thorough check including body scan.

All the ACM student members leave the CCS each morning. Each volunteer is to move to one predetermined stop to invite passengers. There are as many volunteers as stops. At the end of the day, all students travel back to CCS. You are to write a computer program that helps ACM to minimize the amount of money to pay every day for the transport of their employees.

Input

The input consists of *N* cases. The first line of the input contains only positive integer *N*. Then follow the cases. Each case begins with a line containing exactly two integers *P* and *Q*, *1 <= P,Q <= 1000000*. *P* is the number of stops including CCS and *Q* the number of bus lines. Then there are *Q* lines, each describing one bus line. Each of the lines contains exactly three numbers - the originating stop, the destination stop and the price. The CCS is designated by number *1*. Prices are positive integers the sum of which is smaller than *1000000000*. You can also assume it is always possible to get from any stop to any other stop.

Output

For each case, print one line containing the minimum amount of money to be paid each day by ACM for the travel costs of its volunteers.

Example

```
Sample input:

2
2 2
1 2 13
```

```
2 1 33
4 6
1 2 10
2 1 60
1 3 20
3 4 10
2 4 5
4 1 50

Sample output:

46
210
```

Warning: large Input/Output data, be careful with certain languages

---

Added by: Adrian Kosowski
Date: 2004-06-06
Time limit: 5s
Source limit: 50000B
Languages: All
Resource: ACM Central European Programming Contest, Prague 1998

SPOJ Problem Set (classical)

51. Fake tournament

Problem code: TOUR

We consider only special type of tournaments. Each tournament consists of a series of matches. We have *n* competitors at the beginning of a competition and after each match the loser is moved out of the competition and the winner stays in (there are no draws). The tournament ends when there is only one participant left - the winner. It is a task of National Sports Federation to schedule the matches. Members of this committee can pick the contestants for the first match. Then, after they know the result, they say which of the remaining contestants meet in the second match, and so on until there is only one participant left.
It is easy to see that not only skill and training decides about the win, but also "luck" - i.e. the schedule. The members of NSF know it as well.
The committee used the training time to look carefully on the performance of each probable contestant. It is clear now, at the start of the season, that some of the results between the competitors are 100% predictable. Having this information NSF considers if it is possible to schedule the matches in such a way that the given contestant **x** wins. That is to plan the matches for **x** only with those who will lose with him (then he wins the whole tournament of course). If it is possible then w say that **the tournament can be set for x**.

Task

Your task is to write a program which determines the number of contestants of a given tournament for which it is possible to set it.

Input

t [number of tests to solve].
In the first line of each test: n (1<=n<=1000) - the number of participants of the tournament. We number the participants with numbers 1,2, ... ,n. The following line contains a list of participants who will inevitably win with participant 1. This list begins with a number *m* (the number of contestants "better" than 1) and numbers $n_1, n_2, \ldots, n_m$ delimited by single spaces.
Next *n-1* lines contain analogous lists for participants 2, 3, ..., n.
Remark 1. The fact that participant **a** would lose with **b** and **b** would lose with **c** doesn't necessarily mean that **a** would lose with **c** in a direct match.
Remark 2. It is not possible that **a** is on the list of contestants better than **b** and **b** is on the list of **a** at the same time.

Output

For each test your program should output a single integer - the number of participants, for which it is possible to set the tournament.

Example

```
Input:
1
3
2 3 2
1 3
0
Output:
1
```

Added by:	Adam Dzedzej
Date:	2004-06-08
Time limit:	1s
Source limit:	50000B
Languages:	All
Resource:	Internet Contest **Pogromcy Algorytmów** (Algorithm Tamers) Round IV, 2001

SPOJ Problem Set (classical)

54. Julka

Problem code: JULKA

Julka surprised her teacher at preschool by solving the following riddle:

Klaudia and Natalia have 10 apples together, but Klaudia has two apples more than Natalia. How many apples does each of he girls have?

Julka said without thinking: Klaudia has 6 apples and Natalia 4 apples. The teacher tried to check if Julka's answer wasn't accidental and repeated the riddle every time increasing the numbers. Every time Julka answered correctly. The surprised teacher wanted to continue questioning Julka, but with big numbers she could't solve the riddle fast enough herself. Help the teacher and write a program which will give her the right answers.

Task

Write a program which

- reads from standard input the number of apples the girls have together and how many more apples Klaudia has,
- counts the number of apples belonging to Klaudia and the number of apples belonging to Natalia,
- writes the outcome to standard output

Input

Ten test cases (given one under another, you have to process all!). Every test case consists of two lines. The first line says how many apples both girls have together. The second line says how many more apples Klaudia has. Both numbers are positive integers. It is known that both girls have no more than 10^{100} (1 and 100 zeros) apples together. As you can see apples can be very small.

Output

For every test case your program should output two lines. The first line should contain the number of apples belonging to Klaudia. The second line should contain the number of apples belonging to Natalia.

Example

```
Input:
10
2
[and 9 test cases more]
Output:
6
4
[and 9 test cases more]
```

Added by:	Adam Dzedzej
Date:	2004-06-08
Time limit:	2s
Source limit:	50000B
Languages:	All
Resource:	Internet Contest **Pogromcy Algorytmow** (Algorithm Tamers) Round II, 2003

SPOJ Problem Set (classical)

55. Jasiek

Problem code: JASIEK

Jasiek is only 6 years old, but he already has many skills. He likes drawing and asking riddles very much. This morning he got a sheet of grid paper and a pencil from his mother and he started drawing. All his drawings have some common properties:

- Jasiek colors full grid squares;
- if some coloured grid squares touch each other, it means they have a common edge or a corner;
- all grid squares are connected, which means between every two coloured grid squares there is a sequence of coloured grid squares, which have a common edge;
- there are no white holes, that is from every white grid box it is possible to draw a line to the boundary of the sheet which never touches any coloured grid square.

At noon mom phoned and asked what Jasiek's today's picture was. The boy didn't answer directly, but described the picture by a sequence of moves needed to walk around the centres of the coloured squares on its boundary, ie. those squares which have at least one common corner with a white square. Jasiek set the starting square and then gave the sequence of moves necessary to walk along the boundary squares anti-clockwise. Mom was very surprised by the complexity of the picture and especcialy by the number of coloured squares. Given Jasiek's description, can you quickly count how many coloured squares there are in the picture?

Task

Write a program which

- reads (from standard input) Jasiek's description of the picture,
- counts the number of coloured squares,
- writes out the outcome (to standard output).

Input

Ten test cases (given one under another, you have to process all!). Each of the test cases is a series of lines. Each line consists of only one character. The letter *P* means the beginning of the description. The letter *K* means the end of the desription (and the test case). All other lines (if any) contain one of the letters N, W, S or E (N meaning North, W - West, S - South and E - East). Every line of the description corresponds to the relative position of the centre of some square on the boundary of the picture. The first and the last line correspond to the same square. A letter in a line other than the first or the last tells you which way you have to go in order to get to the next boundary square when going around the picture anti-clockwise. Jasiek's description finishes after going around the picture once. The length of the description doesn't exceed 20000 letters.

Output

For every testcase your program should write (to the standard output) only one line with one integer, equal to the number of coloured squares in Jasiek's picture.

Example

Example illustration

Input:

```
P
S
S
S
E
N
E
E
S
E
E
N
N
N
N
S
S
S
W
W
N
N
W
W
W
N
S
K
[and 9 test cases more]
```

Output:

```
23
[and 9 test cases more]
```

Added by:	Adam Dzedzej
Date:	2004-06-09
Time limit:	3s
Source limit:	50000B
Languages:	All
Resource:	Internet Contest **Pogromcy Algorytmow** (Algorithm Tamers) Round III, 2003

SPOJ Problem Set (classical)

56. Dyzio

Problem code: DYZIO

Dyzio is Jasiek's friend and he also likes riddles. Here is a riddle he came up with:

Jasiek, here is a piece of string, which has to be cut into smaller pieces. I will not tell you directly how to do it, but look at this sequence of zeros (0) and ones (1). A one at the begining means that the string has to be cut in half. If the first digit was zero, it would be the only digit in the sequence and mean you don't have to cut anything - I want the whole string. If you have to cut the string anyway then after the first 1 I wrote what to do with the left piece (according to the same rules as with the whole string) and then I wrote what to do with the right piece of string (all the time with the same rules of notation). Every time you have to cut the left piece first, only then can you cut the right one. Now start cutting and tell me, how many cuts you have to do until you have cut off the shortest piece.

Unfortunately mom hid the scissors from Jasiek, but luckily a computer was at hand and Jasiek quickly wrote a program simulating the string cutting. Can you write such a program?

Task

Write a program which

- reads (from standard input) description of the way the string is cut,
- counts how many cuts have to be made in order to get the first shortest piece.
- writes out the outcome (to standard output)

Input

Ten test cases (given one under another, you have to process all!). Each test case consists of two lines. In the first line there is a number *n (1<=n<=20000)*. In the second line one zero-one word (a sequence of zeros and ones without spaces between them) of length *n* - the description of the cutting procedure given by Dyzio.

Output

For every testcase your program should write (to the standard output) only one line with one integer equal to the number of cuts which have to be made in order to get the shortest piece.

Example

Input:
```
9
110011000
[and 9 test cases more]
```
Output:
```
4
[and 9 test cases more]
```

Added by:	Adam Dzedzej
Date:	2004-06-10
Time limit:	3s
Source limit:	50000B
Languages:	All
Resource:	Internet Contest **Pogromcy Algorytmow** (Algorithm Tamers) Round III, 2003

SPOJ Problem Set (classical)

57. Supernumbers in a permutation

Problem code: SUPPER

An *n*-element permutation is an *n*-element sequence of distinct numbers from the set *{1, 2, ...,n}*. For example the sequence 2,1,4,5,3 is a 5-element permutation. We are interested in the longest increasing subsequences in a permutation. In this exemplary permutation they are of length 3 and there are exactly 2 such subsequences: 2,4,5 and 1,4,5. We will call a number belonging to any of the longest increasing subsequences a *supernumber*. In the permutation 2,1,4,5,3 the supernumbers are 1,2,4,5 and 3 is not a supernumber. Your task is to find all supernumbers for a given permutation.

Task

Write a program which

- reads a permutation from standard input,
- finds all its supernumbers,
- writes all found numbers to standard output.

Input

Ten test cases (given one under another, you have to process all!). Each test case consists of two lines. In the first line there is a number *n* (1<=n<=100000). In the second line: an *n*-element permutation - *n* numbers separated by single spaces.

Output

For every test case your program should write two lines. In the first line - the number of supernumbers in the input permutation. In the second line the supernumbers separated by single spaces in increasing order.

Example

Input:
```
5
2 1 4 5 3
[and 9 test cases more]
```
Output:
```
4
1 2 4 5
[and 9 test cases more]
```

Warning: large Input/Output data, be careful with certain languages

---

Added by:	Adam Dzedzej
Date:	2004-06-10
Time limit:	9s
Source limit:	50000B
Languages:	All
Resource:	Internet Contest **Pogromcy Algorytmow** (Algorithm Tamers) Round IV, 2003

SPOJ Problem Set (classical)

58. Crime at Piccadily Circus

Problem code: PICAD

Sherlock Holmes is carrying out an investigation into the crime at Piccadily Circus. Holmes is trying to determine the maximal and minimal number of people staying simultaneously at the crime scene at a moment when the crime could have been commited. Scotland Yard has already carried out a thorough investigation already, interrogated everyone seen at the crime scene and determined what time they appeared at the crime scene and what time they left. Doctor Watson offered his help to process the data gathered by Scotland Yard and find the numbers interesting Sherlock Holmes, but he has some difficulties. Help him!

Task

Write a program which

- reads from standard input the time interval during which the crime was commited and the data gathered by Scotland Yard,
- finds the minimal and the maximal number of people present simultaneously in the time interval when the crime could have been commited, (these numbers can be zero, though it would seem strange that noone was present at the crime scene when the crime was commited, but that's the type of crime Holmes and Watson have to deal with)
- writes the outcome to standard output.

Input

Ten test cases (given one under another, you have to process all!). The first line of each test case consists of two integer numbers p and k, $0<=p<=k<=100000000$. These denote the first and the last moment when the crime could have been commited. The second line of each test case contains one integer n, $3<=n<=5000$. This is the number of people interrogated by Scotland Yard. The next n lines consist of two integers - line $i+2$ contains numbers a_i and b_i separated by a single space, $0<=a_i<=b_i<=1000000000$. These are the moments at which the i-th person apperared at and left the crime scene respectively. It means that the i-th person was at the crime scene for the whole time from moment a_i until moment b_i (inclusive).

Output

For every test case your program should write to the standard output only one line with two integers separated by a single space: the minimal and maximal number of people staying simultaneously at the crime scene, in the interval between moment p and k, (inclusive).

Example

Only one test case.

Input:
```
5 10
4
1 8
5 8
7 10
8 9
```
Output:
```
1 4
```

Added by:	Adam Dzedzej
Date:	2004-06-10
Time limit:	13s
Source limit:	50000B
Languages:	All
Resource:	Internet Contest **Pogromcy Algorytmow** (Algorithm Tamers) Round IV, 2003

SPOJ Problem Set (classical)

59. Bytelandian Information Agency

Problem code: BIA

Bytelandian Information Agency (BIA) uses a net of *n* computers. The computers are numbered from *1* to *n*, and the computer number *1* is a server. The computers are connected by one-way information channels. Every channel connects a pair of computers. The whole network is organised in such a way that one can send information from the server to any other computer either directly or indirectly.

When BIA acquires new information, the information is put on the server and propagated in the net. The chief of BIA considers what would happen if one computer stopped working (was blown away by terrorists for example). It could happen that some other computers would stop receiving information from the server, because the broken computer was a necessary transmitter. We will call such computers *critical*. For example in the situation in the picture below the critical computers are *1* and *2*. *1* is the server and all information sent from the server to *3* has to go through *2*.

BIA computer net

Task

Write a program which

- reads a description of the net from standard input,
- finds all critical computers.
- writes the numbers of critical computers to standard output.

Input

Ten test cases (given one under another, you have to process all!). Each test case consists of several lines. In the first line there are numbers *n* and *m*. *n* denotes the number of computers in the net,*(2<=n<=5000)*. *m* denotes the number of information channels, *n-1<=m<=200000*. The following *m* lines describes a single information channel and consist of two integer numbers *a* and *b* separated by a space. It means the computer *a* sends information to computer *b* by that channel. You may assume there are no two channels which start and end at the same points *a, b*.

Output

For every testcase your program should write two lines. In the first line *k* - the number of critical computers in the net. In the second line *k* numbers separated by single spaces - the numbers of critical computers in increasing order.

Example

Input:

```
4 5
1 2
1 4
```

```
2 3
3 4
4 2
[and 9 test cases more]
```

Output:

```
2
1 2
[and 9 test cases more]
```

Warning: large Input/Output data, be careful with certain languages

---

Added by:	Adam Dzedzej
Date:	2004-06-14
Time limit:	7s
Source limit:	50000B
Languages:	All
Resource:	Internet Contest **Pogromcy Algorytmow** (Algorithm Tamers) Round IV, 2003

SPOJ Problem Set (classical)

60. The Gordian Dance

Problem code: DANCE

The Gordian Dance is a traditional Bytelandian dance performed by two pairs of dancers. At the beginning the dancers are standing in the corners of the square *ABCD*, forming two pairs: *A*-*B* and *C*-*D*. Every pair is holding an outstretched string. So in the starting position both strings are stretched horizontally and parallel.

The starting position of dancers.

The dance consists of a series of moves. There are two kinds of moves:

- (*S*) The dancers standing at points *B* and *C* swap positions (without releasing their strings) in such a way that the dancer standing at *B* raises the hand in which he is holding the string and, when going to point *C*, lets the dancer going from *C* to *B* pass in front of him, under his arm.
- (*R*) All dancers make a turn by 90 degrees clockwise without releasing their strings. This means that the dancer from *A* goes to *B*, the dancer from *B* goes to *C*, the dancer from *C* goes to *D*, and the dancer from *D* goes to *A*.

During the dance the strings tangle with each other, but in the end they should be untangled and stretched horizontally and parallel. The dancers do not have to occupy the same spots as in the begining. The dance requires a lot of experience, because the strings can be extremely tangled during the dance. The sequence of moves after which they are no longer tangled and are stretched horizontally and parallel can be difficult to guess.

Your program should help beginner dancers end a dance. You are to determine the minimal number of mover required to end the dance given a sequence of moves already performed.

Illustration

For example after the sequence *SS* we get the following configuration.

The configuration after SS

The shortest sequence of moves required to end the dance is of length 5: *RSRSS*.

Task

Write a program which

- reads from standard input the moves made in a dance,
- finds the minimal number of moves required to untangle the strings and stretch them horizontally and parallel (the dancers don't have to be in their starting spots).
- writes the outcome to standard output.

Input

Ten test cases (given one under another, you have to process all!). The first line of each test case consists of one integer *n* equal to the nmber of moves already made, *0<=n<=1000000*. The second line of each test case consists of one word of length *n*, made up of letters *S* and/or *R*.

Output

For every testcase your program should write to standard output only one line with one integer number: the minimal number of moves required to untangle the strings and stretch them horizontally and parallel.

Example

Input:

```
2
SS
[and 9 test cases more]
```

Output:

```
5
[and 9 test cases more]
```

Warning: large Input/Output data, be careful with certain languages

---

Added by: Adam Dzedzej
Date: 2004-06-15
Time limit: 3s
Source limit:50000B
Languages: All
Resource: Internet Contest **Pogromcy Algorytmow**(Algorithm Tamers) 2003 Round V

SPOJ Problem Set (classical)

61. Brackets

Problem code: BRCKTS

We will call a **bracket word** any word constructed out of two sorts of characters: the opening bracket "(" and the closing bracket ")". Among these words we will distinguish **correct bracket expressions**. These are such bracket words in which the brackets can be matched into pairs such that

- every pair consists of an opening bracket and a closing bracket appearing further in the bracket word
- for every pair the part of the word between the brackets of this pair has equal number of opening and closing brackets

On a bracket word one can do the following operations:

- **replacement** -- changes the i-th bracket into the opposite one
- **check** -- if the word is a correct bracket expression

Task

Write a program which

- reads (from standard input) the bracket word and the sequence of operations performed,
- for every check operation determines if the current bracket word is a correct bracket expression,
- writes out the outcome (to standard output).

Input

Ten test cases (given one under another, you have to process all!). Each of the test cases is a series of lines. The first line of a test consists of a single number *n (1<=n<=30000)* denoting the length of the bracket word. The second line consists of *n* brackets, not separated by any spaces. The third line consists of a single number *m* -- the number of operations. Each of the following *m* lines carries a number *k* denoting the operation performed. *k=0* denotes the check operation, *k>0* denotes replacement of *k*-th bracket by the opposite.

Output

For every test case your program should print a line:
Test i:
where i is replaced by the number of the test and in the following lines, for every check operation in the i-th test your program should print a line with the word YES, if the current bracket word is a correct bracket expression, and a line with a word NO otherwise. (There should be as many lines as check operations in the test.)

Example

```
Input:
4
()((
4
4
0
2
0
[and 9 test cases more]
Output:
Test 1:
YES
NO
[and 9 test cases more]
```

Warning: large Input/Output data, be careful with certain languages

---

Added by: Adam Dzedzej
Date: 2004-06-15
Time limit: 11s
Source limit:50000B
Languages: All
Resource: Internet Contest **Pogromcy Algorytmow**(Algorithm Tamers) 2003 Round IV

SPOJ Problem Set (classical)

62. The Imp

Problem code: IMP

An Imp jumps on an infinite chessboard. Moves possible for the Imp are described by two pairs of integers: (a,b) and (c,d) - from square (x,y) the Imp can move to one of the squares: (x+a,y+b), (x-a,y-b), (x+c,y+d), (x-c,y-d). We want to know for which square different from (0,0) to which the Imp can jump from (0,0) (possibly in many moves) the value |x|+|y| is the lowest.

Task

Write a program which

- reads from standard input two pairs (a,b) and (c,d) of integers, different from (0,0), describing moves of the Imp,
- determines a pair of integers (x,y) different from (0,0), for which the Imp can jump (possibly in many moves) from square (0,0) to square (x,y) and for which the value |x|+|y| is the lowest.
- writes out to standard output the value |x|+|y|.

Input

Ten test cases. Each test consists of four numbers a,b,c,d in one line, separated by spaces.
-100000 <= a, b, c, d <= 100000

Output

For every test case your program should write a single line with a number equal the lowest possible value |x|+|y|.

Example

Input:
```
13 4 17 5
[and 9 test cases more]
```
Output:
```
2
[and 9 answers more]
```

Added by: Adam Dzedzej
Date: 2004-06-15
Time limit: 3s
Source limit:50000B
Languages: All
Resource: Internet Contest **Pogromcy Algorytmow**(Algorithm Tamers) 2003 Round V

SPOJ Problem Set (classical)

63. Square Brackets

Problem code: SQRBR

You are given:

- a positive integer n,
- an integer k, 1<=k<=n,
- an increasing sequence of k integers $0 < s_1 < s_2 < ... < s_k <= 2n$.

What is the number of proper bracket expressions of length 2n with opening brackets appearing in positions $s_1, s_2,...,s_k$?

Illustration

Several proper bracket expressions:

```
[[][[[]][]]
[[[][]]][[]]
```

An improper bracket expression:

```
[[[][]]][][[]]
```

There is exactly one proper expression of length 8 with opening brackets in positions 2, 5 and 7.

Task

Write a program which for each data set from a sequence of several data sets:

- reads integers n, k and an increasing sequence of k integers from input,
- computes the number of proper bracket expressions of length 2n with opening brackets appearing at positions $s_1,s_2,...,s_k$,
- writes the result to output.

Input

The first line of the input file contains one integer d, 1 <= d <= 10, which is the number of data sets. The data sets follow. Each data set occupies two lines of the input file. The first line contains two integers n and k separated by single space, 1 <= n <= 19, 1 <= k <= n. The second line contains an increasing sequence of k integers from the interval [1;2n] separated by single spaces.

Output

The i-th line of output should contain one integer - the number of proper bracket expressions of length 2n with opening brackets appearing at positions $s_1, s_2,...,s_k$.

Example

Sample input:

```
5
1 1
1
1 1
2
2 1
1
3 1
2
4 2
5 7
```

Sample output:

```
1
0
2
3
2
```

Added by: Adrian Kosowski
Date: 2004-06-22
Time limit: 3s
Source limit: 50000B
Languages: All
Resource: III Polish Collegiate Team Programming Contest (AMPPZ), 1998

SPOJ Problem Set (classical)

64. Permutations

Problem code: PERMUT1

Let A = [a_1,a_2,...,a_n] be a permutation of integers 1,2,...,n. A pair of indices (i,j), 1<=i<=j<=n, is an *inversion* of the permutation A if a_i>a_j. We are given integers n>0 and k>=0. What is the number of n-element permutations containing exactly k inversions?

For instance, the number of 4-element permutations with exactly 1 inversion equals 3.

Task

Write a program which for each data set from a sequence of several data sets:

- reads integers n and k from input,
- computes the number of n-element permutations with exactly k inversions,
- writes the result to output.

Input

The first line of the input file contains one integer d, 1<=d<=10, which is the number of data sets. The data sets follow. Each data set occupies one line of the input file and contains two integers n (1<=n<=12) and k (0<=k<=98) separated by a single space.

Output

The i-th line of the output file should contain one integer - the number of n-element permutations with exactly k inversions.

Example

Sample input:
```
1
4 1
```

Sample output:
```
3
```

Added by: Adrian Kosowski
Date: 2004-06-22
Time limit: 3s
Source limit:50000B
Languages: All
Resource: III Polish Collegiate Team Programming Contest (AMPPZ), 1998

SPOJ Problem Set (classical)

65. Ball

Problem code: BALL1

On the rectangular chessboard of n x m square fields we choose one field adjacent to the edge of the chessboard, called the starting field. Then we put a ball in the center of this field and push it to roll through the chessboard. The diameter of the ball equals the width (and height) of chessboard field. The angle between the direction of ball movement and the edge of the chessboard equals 45 degrees. The ball bounces off the edges of the chessboard: if the ball touches the edge of the chessboard then each composite of its velocity perpendicular to the edge touched is reversed. At the start the ball is pushed toward increasing coordinates (when the starting field is a field of the highest coordinate, the ball bounces momentarily).

We assign a point to a field of the chessboard each time the point of adjacency between the ball and the chessboard enters the interior of the field. The game is over when a point is assigned to the starting field. What is the number of fields to which an odd number of points is assigned? The following figures illustrate the problem. The route of the ball is marked with a dashed line. Fields with the odd number of points are shadowed.

[IMAGE]

Task

Write a program which for each data set from a sequence of several data sets:

- reads the dimensions of the chessboard and the coordinates of starting field from input,
- computes the number of fields with the odd number of points,
- writes the result to output.

Input

The first line of the input file contains one integer d, 1 <= d <= 10, which is the number of data sets. The data sets follow. Each data set occupies one line of the input file. Such a line consists of four integers x, y, a, b separated with single spaces. These integers are the x- and y-dimensions of the chessboard and x- and y-coordinates of the starting field, respectively. Integers x and y are greater than two, the number of fields of the chessboard does not exceed 10^9,the starting field is adjacent to the edge of the chessboard.

Output

The i-th line of output should contain one integer which is equal to the number of fields of the chessboard with the odd number of points.

Example

```
Sample input:

2
13 6 1 5
10 7 1 5

Sample output:

2
22
```

Added by: Adrian Kosowski
Date: 2004-06-06
Time limit: 5s
Source limit: 50000B
Languages: All
Resource: III Polish Collegiate Team Programming Contest (AMPPZ), 1998

SPOJ Problem Set (classical)

66. Cross-country

Problem code: CRSCNTRY

Agness, a student of computer science, is very keen on cross-country running, and she participates in races organised every Saturday in a big park. Each of the participants obtains a route card, which specifies a sequence of checkpoints, which they need to visit in the given order. Agness is a very atractive girl, and a number of male runners have asked her for a date. She would like to choose one of them during the race. Thus she invited all her admirers to the park on Saturday and let the race decide. The winner would be the one, who scores the maximum number of points. Agnes came up with the following rules:

- a runner scores one point if he meets Agnes at the checkpoint,
- if a runner scored a point at the checkpoint, then he cannot get another point unless he and Agnes move to the next checkpoints specified in their cards.
- route specified by the card may cross the same checkpoint more than once,
- each competitor must strictly follow race instructions written on his card.

Between two consecutive meetings, the girl and the competitors may visit any number of checkpoints. The boys will be really doing their best, so you may assume, that each of them will be able to visit any number of checkpoints whilst Agnes runs between two consecutive ones on her route.

Task

Write a program which for each data set from a sequence of several data sets:

- reads in the contents of Agnes' race card and contents of race cards presented to Tom,
- computes the greatest number of times Tom is able to meet Agnes during the race,
- writes it to output.

Input

There is one integer d in the first line of the input file, 1 <= d <= 10. This is the number of data sets. The data sets follow. Each data set consists of a number of lines, with the first one specifying the route in Agnes' race card. Consecutive lines contain routes on cards presented to Tom. At least one route is presented to Tom. The route is given as a sequence of integers from interval [1, 1000] separated by single spaces. Number 0 stands for the end of the route, though when it is placed at the beginning of the line it means the end of data set. There are at least two and at most 2000 checkpoints in a race card.

Output

The i-th line of the output file should contain one integer. That integer should equal the greatest number of times Tom is able to meet with Agnes for race cards given in the i-th data set.

Example

```
Sample input:

3
1 2 3 4 5 6 7 8 9 0
1 3 8 2 0
2 5 7 8 9 0
1 1 1 1 1 1 2 3 0
1 3 1 3 5 7 8 9 3 4 0
1 2 35 0
0
1 3 5 7 0
3 7 5 1 0
0
1 2 1 1 0
1 1 1 0
0

Sample output:

6
2
3
```

Added by: Adrian Kosowski
Date: 2004-06-08
Time limit: 5s
Source limit: 50000B
Languages: All
Resource: III Polish Collegiate Team Programming Contest (AMPPZ), 1998

SPOJ Problem Set (classical)

67. Cutting out

Problem code: CUTOUT

One has to cut out a number of rectangles from a paper square. The sides of each rectangle are to be parallel to the sides of the square. Some rectangles can be already cut out. What is the largest area of a rectangle which can be cut out from the remaining paper?

Illustration

Three rectangles have been cut out from the square 10x10 in the figure shown below. The area of the largest rectangle that can be cut out from the remaining paper is 16. One of such rectangles is shown with a dashed line.

[IMAGE]

Task

Write a program that for each data set from a sequence of several data sets:

- reads descriptions of a square and rectangles from the input,
- computes the area of the largest rectangle which can be cut out from the remaining paper,
- writes the result to output.

Input

The first line of the input file contains one positive integer d not larger than 10. This is the number of data sets. The data sets follow. Each set of data occupies two consecutive lines of the input file. The first line of each data set contains two integers n and r, 1 <= n <= 40000, 0 <= r <= 100. The integer n is the length of the sides of an input square. The integer r is the number of rectangles which have been cut out from the square. The second line of the data set contains a sequence of 4r integers x_1, x_2,...,x_{4r} from the interval [0,n] separated by single spaces. For each i = 1,...,r, integers x_{4i-3}, x_{4i-2}, x_{4i-1}, x_{4i} describe the i-th rectangle: x_{4i-3} is the distance of its left side from the left side of the square, x_{4i-2} is the distance of its right side from the left side of the square, x_{4i-1} is the distance of the bottom side of the rectangle from the bottom side of the square and x_{4i} is the distance of its top side from the bottom side of the square.

Output

For each i = 1,...,d, your program should write only one integer to the i-th line of the output file -- the largest area of a rectangle which can be cut out from the rest of the i-th square.

Example

```
Sample input:

2
6 2
0 3 0 3 3 6 3 6
10 3
0 5 0 5 0 10 5 10 9 10 0 5

Sample output:

9
20
```

Added by: Adrian Kosowski
Date: 2004-06-08
Time limit: 5s
Source limit: 50000B
Languages: All
Resource: III Polish Collegiate Team Programming Contest (AMPPZ), 1998

SPOJ Problem Set (classical)

68. Expression

Problem code: EXPR1

We are given an integer k and an arithmetic expression E with the operations '+', '-', and arguments from the set {0,1,...,9}. Is it possible to put some parentheses in E to get a new expression E' whose value equals k? If the answer is positive what is the minimum number of pairs of parentheses '(', ')' that are necessary?

Illustration

It is sufficient to put one pair of parentheses in the expression 5 - 4 + 5 to get an expression with value -4, namely 5 - (4 + 5) = -4.

Task

Write a program that for each data set from a sequence of several data sets:

- reads an expression E and an integer k from input,
- verifies whether it is possible to put some parentheses in E to get a new expression E' whose value equals k and computes the minimal number of pairs of parentheses '(', ')' necessary, if the answer is positive,
- writes the result to output.

Input

The first line of the input file contains one positive integer d not larger than 10. This is the number of data sets. The data sets follow. Each set of data occupies two consecutive lines of the input file. The first line contains two integers n and k, 2 <= n <= 40, -180 <= k <= 180. The even integer n is the length of E. The second line contains the expression itself written as a string of length n. The string contains operators '+' or '-' in odd positions and numbers from the set {0,1,...,9} in even positions.

Output

For each i = 1,...,d, your program should write to the i-th line of the output file one word 'NO' if the i-th input expression cannot be transformed into any expression of value k, and the smallest number of pairs of parentheses necessary otherwise.

Example

```
Sample input:

5
6 -4
+5-4+5
2 1
+1
```

```
4 1
-1+1
4 0
-1+1
4 -2
-1+1
```

Sample output:

```
1
0
NO
0
1
```

Added by: Adrian Kosowski
Date: 2004-06-08
Time limit: 5s
Source limit: 50000B
Languages: All
Resource: III Polish Collegiate Team Programming Contest (AMPPZ), 1998

SPOJ Problem Set (classical)

69. Moulds

Problem code: MOULDS

In a factory, moulds for casting metal objects are produced by a special cutting device. The device is equipped with cuboid-shaped blade of size 1 mm x 1 mm x 30 mm (its height) which operates with each of its sides thus producing the mould from cuboid of size 250 mm x 250 mm x 30 mm (its height). The end of the blade newer lowers below the bottom surface of the cuboid. In any moment the distance between initial and current position doesn't exceed 1000.

The machine understands special command language which has the following grammar:

```
<command block> ::= [ <command> ; {<command> ; } ]
<command>       ::= <lift> | <shift> | <command block>
<lift>          ::= ^ <distance>
<shift>         ::= @ <direction> <distance>
<direction>     ::= N | S | W | E
<distance>      ::= <sign> <number> | <number>
<number>        ::= <digit> {<digit>}
<sign>          ::= - | +
<digit>         ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
```

where {exp} means zero or more exps.

The command <lift> causes moving the blade downwards when the distance is a positive number and upwards otherwise. The command <shift> moves the blade in the appropriate direction (N--north, S--south, W--west, E--east).

Task

Write a program which for each data set from a sequence of several data sets:

- reads a command block from input,
- computes the volume of hollows made by the machine commanded by a given command block (assuming that before the execution the blade is located 1 mm above the north-west corner of the virgin cuboid),
- writes the result to output.

Input

The first line of the input file contains one integer d, 1 <= d <= 10, which is the number of data sets. The data sets follow. Each data set occupies one line of the input file and is a word derived from <command block> of the above grammar of length not exceeding 10000 characters.

Output

The i-th line of the output file should contain one integer -- the volume (in cubic mm) of the hollows made by the machine controlled by the command block given in the i-th data set.

Example

```
Sample input:

1
[^2;@S2;]

Sample output:

3
```

Added by: Adrian Kosowski
Date: 2004-06-08
Time limit: 5s
Source limit: 50000B
Languages: All
Resource: III Polish Collegiate Team Programming Contest (AMPPZ), 1998

SPOJ Problem Set (classical)

70. Relations

Problem code: RELATS1

You are given a directed graph, whose edges are labeled with relational symbols '<', '>' and '='. For a nonnegative integer k, a k-correct G-labeling is a mapping from vertices of G into integers from interval [0,k] such that numbers at the ends of each edge satisfy the relation described by the label of the edge. We assume that an element on the left side of the relational symbol is a number assigned to the initial vertex. Compute the smallest k for which k-correct G-labeling exists or verify that such labeling doesn't exist for any k.

Illustration

For the graph in the figure the smallest k = 2.

[IMAGE]

Task

Write a program that for each data set from a sequence of several data sets:

- reads a description of a graph G from the input file,
- verifies whether there exist an integer k for which it is possible to label G k-correctly and, if the answer is positive, computes the smallest such k,
- writes the result to the output file.

Input

The first line of the input file contains one positive integer d not larger than 10. This is the number of data sets. The data sets follow. Each data set is described in two consecutive lines of the input file. In the first line there are two integers n and m separated by a single space. The number n is the number of vertices of G and m is the number of edges of G. Numbers n and m satisfy the inequalities: 1 <= n <= 1000, 0 <= m <= 10000. The vertices are numbered with integers from 1 to n and are identified by these numbers. There are no parallel edges and self-loops in the graph. (Two different edges u_1 -> v_1 and u_2 -> v_2 are parallel iff $u_1 = u_2$ and $v_1 = v_2$.) There are 3m integers separated by single spaces in the second line. The numbers at positions 3i-2 and 3i-1, 1 <= i <= m, are the ends of the i-th edge, the beginning and the end, respectively, whereas the number at position 3i is a number from the set {-1,0,1} and it is the label of the i-th edge: -1 represents '<', 0 represents '=' and 1 represents '>'.

Output

For the i-th data set, 1 <= i <= d, your program should write one word NO in the i-th line of the output file if a k-correct labeling doesn't exist for any k, or the smallest integer k for which such a labeling exists.

Example

Sample input:

```
4
4 4
1 2 -1 2 3 0 2 4 -1 3 4 -1
2 2
1 2 -1 2 1 -1
2 2
1 2 -1 2 1 1
3 3
1 2 0 3 2 0 3 1 0
```

Sample output:

```
2
NO
1
0
```

Added by: Adrian Kosowski
Date: 2004-06-08
Time limit: 5s
Source limit: 50000B
Languages: All
Resource: III Polish Collegiate Team Programming Contest (AMPPZ), 1998

SPOJ Problem Set (classical)

71. Tree

Problem code: TREE1

Consider an n-vertex binary search tree T containing n keys 1,2,...,n. A permutation $p = [p_1,...,p_n]$ of the integers 1,2,...,n is said to be *consistent with the tree T* if the tree can be built from the empty one as the result of inserting integers $p_1,p_2,...,p_n$. Find how many permutations are consistent with the tree T.

Illustration

Exactly 2 permutations are consistent with the tree in the figure below.

[IMAGE]

Task

Write a program that for each data set from a sequence of several data sets:

- reads from the input file a description of an input tree T,
- computes the number of permutations consistent with T,
- writes the result to output.

Input

The first line of the input file contains one positive integer d not larger than 10. This is the number of data sets. The data sets follow. Each set of data occupies two consecutive lines of the input file. The first line contains only one integer n, 1 <= n <= 30. This is the number of vertices of the tree. The second line contains a sequence of n integers separated by single spaces. The integers are keys in the input tree given in the prefix order. The first integer in the sequence is the key from the root of the tree. It is followed by the keys from the left subtree written in the prefix order. The sequence ends with the keys from the right subtree, also given in the prefix order.

Output

For each i = 1,...,d, your program should write to the i-th line of output the number of permutations consistent with the tree described in the i-th data set.

Example

```
Sample input:

5
3
2 1 3
3
1 2 3
```

```
1
1
4
2 1 3 4
4
1 4 2 3
```

Sample output:

```
2
1
1
3
1
```

Added by: Adrian Kosowski
Date: 2004-06-08
Time limit: 5s
Source limit: 50000B
Languages: All
Resource: III Polish Collegiate Team Programming Contest (AMPPZ), 1998

SPOJ Problem Set (classical)

73. Bacterial

Problem code: BAC

In the biology laboratory we are observing several bacterial samples, and under the microscope we have them shaded with different colors to see them expanding their territory on the plate.

It is interesting to know that the bacterial are quite 'friendly' that once they meet each other, they do not expand into each other's occupation any more. The bacterial samples are expanding at similar speeds and we take them as the same speed.

Since the experiment is tedious and lengthy (Oh My God! there are several thousand samples at our pick), we are going to run a simulation based on this reality, taking the variable that these samples may be planted in different starting spots.

We are using rectangular plates and bacterial racing is bounded within the plate.

[IMAGE]

Input format

There are multiple test cases (about 20000 of them) each taking the following format:

- one line with two integers between 1 and 1000 inclusive indicating width and height of the plate
- one line with one integer between 1 and 100 inclusive indicating the number of bacterial samples
- for each bacterial sample there is one line with two integers indicating the sample's position:
  x y, where x, y specify a position within or on the bound of the plate.

The plate lies in such a coordinating system that the lower-left corner of it is (0,0) and the upper-right corner is (width,height).

A test with zero plate area marks the end of the tests and this one shall not be processed.

Between each input block there is a blank line.

Output format

Generate a report having the samples sorted on their domination, with each line taking the following format:
<sample id> <area occupation>
where: 'sample id' takes 3 columns right justified, with '0' padded to the left as necessary, and 'area occupation' takes 14 columns with 2 digit precision, right justified.

The sample occupying more area shall be reported prior to those occupying less. The input data will ensure enough difference in areas to avoid ambiguity.

Between each output block there shall be a blank line.

Example

Sample input:

```
10 10
2
5 5
0 0

0 0
```

Sample output:

```
001         87.50
002         12.50
```

Warning: large Input/Output data, be careful with certain languages

---

Added by: Neal Zane
Date: 2004-06-08
Time limit: 9s
Source limit: 50000B
Languages: All
Resource: Neal Zane

SPOJ Problem Set (classical)

74. Divisor Summation

Problem code: DIVSUM

Given a natural number n (1 <= n <= 500000), please output the summation of all its proper divisors.

Definition: A proper divisor of a natural number is the divisor that is strictly less than the number.

e.g. number 20 has 5 proper divisors: 1, 2, 4, 5, 10, and the divisor summation is: 1 + 2 + 4 + 5 + 10 = 22.

Input

An integer stating the number of test cases (equal to about 200000), and that many lines follow, each containing one integer between 1 and 500000 inclusive.

Output

One integer each line: the divisor summation of the integer given respectively.

Example

```
Sample Input:
3
2
10
20

Sample Output:
1
8
22
```

Warning: large Input/Output data, be careful with certain languages

---

Added by: Neal Zane
Date: 2004-06-10
Time limit: 3s
Source limit:5000B
Languages: All
Resource: Neal Zane

SPOJ Problem Set (classical)

75. Editor

Problem code: EDIT1

Have you ever programmed in Brainf**k? If yes, then you know how annoying it is to press the same key several times in a row. So what we all need, is a good editor. Here are the functions that the editor should have:

- '\n': begin a new line. If the last line was empty, stop processing and print out all lines.
- 'd': copy all characters from the current line, and append them after the last character in this line. For example, if current line contains ab, and d is pressed two times, the result will be abababab
- any other character: append it to the current line.

Please note, that the solution may only be submitted in Brainfk or Intercal.**

Input

There is exactly one test case. You can assume, that there is no key press of 'd' when the line is still empty.

Output

Print the output that the editor described above would produce on the given input. You can assume, that no line is created with more than 150 characters.

Example

Input:

```
sample-test-dd-d-ddend signalled by two newlines
```

Output:

```
sample-test----------------------------------------------enen signalleenen signalle by two newlines
```

Added by: Adrian Kuegel
Date: 2004-06-12
Time limit: 3s
Source limit:50000B
Languages: BF ICK
Resource: own problem

SPOJ Problem Set (classical)

76. Editor Inverse

Problem code: EDIT2

You are given a text. Calculate the minimum number of keystrokes needed to produce this text, if the editor described below is used.
If you haven't read the problem "Editor" before, here is a description of the functionality of the editor:

- '\n': begin a new line. If the last line was empty, stop processing and print out all lines.
- 'd': copy all characters from the current line, and append them after the last character in this line. For example, if current line contains ab, and d is pressed two times, the result will be abababab
- any other character: append it to the current line.

Input

The input consists of **exactly ten** test cases. Each test case consists of a line with at most 600 characters. The character 'd' is not used in any of the lines, but all other printable ascii characters may occur.

Output

For each test case, first print a line containing the minimum number of key strokes to produce the given line of text. In the next lines, write the keys that are pressed to produce the text. If there are several possibilites with minimum number of keystrokes, you should also minimise the number of lines, if there is still more than one possibility, minimise number of keystrokes before the first '\n', then second '\n', ...

Since 'd' is a costly operation in the editor, for each output line you should minimise the number of 'd' characters as the 2nd criterion after minimising number of keystrokes in this line.

The original input line should be the same as the output of the editor (processing the output you produce), if '\n' characters are ignored.

Notice that you have to terminate the input for the editor with two '\n'.

Example

Here only two test cases.

Input:

```
000011234444567890000112344444446789
```

Output:

```
18
00d1123444456789

18
00d1123
444d6789
```

Added by: Adrian Kuegel
Date: 2004-06-12
Time limit: 3s
Source limit: 50000B
Languages: All
Resource: own problem

SPOJ Problem Set (classical)

77. New bricks disorder

Problem code: BRICKS

You have n bricks arranged in a line on the table. There is exactly one letter on each of them. Your task is to rearrange those bricks so that letters on them create some specified inscription. While rearanging you can only swap adjacent bricks with specified letters (you are given m pairs (a1,b1),...,(am,bm) and you are only allowed to swap bricks with ai on one of them and bi on the second, for some i=1,..,m). You should check if it is possible to accomplish this - and if it is - calculate minimal needed number of swaps.

Input

There is a single integer c on the first line of input. Then c test cases follow: each of them consists of two lines of small letters (a..z) with lengths not exceeding 100000 (descriptions of starting and ending configurations), one integer m in the next line and then m lines with two letters ai,bi in each of them.

Output

For each test case you should print -1 if it is not possible to rearrange bricks or the minimal number of swaps if it is possible (if so, output this value modulo 2^{32}).

Example

```
Input:
4
ab
ba
0
abc
cba
3
ab
cb
ca
cabbbc
cbabbc
1
ab
abba
baab
1
ab

Output:
-1
3
1
2
```

Warning: large Input/Output data, be careful with certain languages

Added by: Pawel Gawrychowski
Date: 2004-06-17
Time limit: 9s
Source limit: 10000B
Languages: All

SPOJ Problem Set (classical)

78. Marbles

Problem code: MARBLES

Hänschen dreams he is in a shop with an infinite amount of marbles. He is allowed to select n marbles. There are marbles of k different colors. From each color there are also infinitely many marbles. Hänschen wants to have at least one marble of each color, but still there are a lot of possibilities for his selection. In his effort to make a decision he wakes up. Now he asks you how many possibilites for his selection he would have had. Assume that marbles of equal color can't be distinguished, and the order of the marbles is irrelevant.

Input

The first line of input contains a number T <= 100 that indicates the number of test cases to follow. Each test case consists of one line containing n and k, where n is the number of marbles Hänschen selects and k is the number of different colors of the marbles. You can assume that 1<=k<=n<=1000000.

Output

For each test case print the number of possibilities that Hänschen would have had. You can assume that this number fits into a signed 64 bit integer.

Example

Input:
```
2
10 10
30 7
```

Output:
```
1
475020
```

Added by: Adrian Kuegel
Date: 2004-06-19
Time limit: 1s
Source limit: 10000B
Languages: All
Resource: own problem

SPOJ Problem Set (classical)

82. Easy Problem

Problem code: EASYPIE

Last year there were a lot of complaints concerning the set of problems. Most contestants considered our problems to be too hard to solve. One reason for this is that the team members responsible for the problems are not able to evaluate properly whether a particular problem is easy or hard to solve. (We have created until now so many problems, that all seems quite easy.) Because we want our future contests to be better we would like to be able to evaluate the hardness of our problems after the contest using a history of submissions.

There are a few statistics that we can use for evaluating the hardness of a particular problem: the number of accepted solutions of the problem, the average number of submissions of the problem and the average time consumed to solve it (as "General rules" of the contest state "the time consumed for a solved problem is the time elapsed from the beginning of the contest to the submittal of the accepted run"). For the latter two statistics we consider only the teams which solved this particular problem. Needless to say we ask you to write a program that computes aforementioned statistics for all problems.

Task

Write a program that:

- reads a history of submissions during an ACM contest,
- computes for each problem the number of accepted solutions of the problem, the average number of submissions and the average time consumed to solve it,
- writes the result.

Input

The input begins with the integer t, the number of test cases. Then t test cases follow.

For each test case, the first line of the input contains one integer n (1 <= n <= 2000) being the number of submissions during the contest. Each of the next n lines describes one submission and contains a submission time (measured in seconds from the beginning of the contest), a team identifier, a problem identifier and a result of evaluating the submission separated by single spaces. The submission time is a positive integer not greater then 18000. The team identifier is a non-empty string consisting of at most five small letters or digits. The problem identifier is a capital letter A, B, ..., or I. The result is a capital letter A (the submission is accepted) or R (the submission is rejected).

Submissions are given in nondecreasing order according to submission times and there are 62 teams competing.

Please note that if a problem is accepted all further submission of this problem by the same team are possible but they should not be taken to the statistics.

Output

For each test case the output consists of nine lines. The first line corresponds to problem A, the second line to problem B, and so on. Each line should contain the problem identifier, the number of accepted solutions of the problem, the average number of submissions done by teams that solved that problem and the average time consumed to solve it separated by single spaces. The latter two statistics should be printed only if there was at least one accepted solution of the given problem and should be rounded to two fractional digits (in particular 1.235 should be rounded to 1.24).

Example

Sample input:

```
1
12
10 wawu1 B R
100 chau1 A A
2000 uwr2 B A
2010 wawu1 A R
2020 wawu1 A A
2020 wawu1 B A
4000 wawu2 C R
6000 chau1 A R
7000 chau1 A A
8000 pp1 A A
8000 zil2 B R
9000 zil2 B A
```

Sample output:

```
A 3 1.33 3373.33
B 3 1.67 4340.00
C 0
D 0
E 0
F 0
G 0
H 0
I 0
```

Added by: Adrian Kosowski
Date: 2004-06-26
Time limit: 3s
Source limit: 50000B
Languages: All
Resource: ACM Central European Programming Contest, Warsaw 2003

SPOJ Problem Set (classical)

83. Bundling

Problem code: BUNDLE

Outel, a famous semiconductor company, recently released a new model of microprocessor called Platinium. Like many modern processors, Platinium can execute many instructions in one clock step providing that there are no dependencies between them (instruction I_2 is dependent on instruction I_1 if for example I_2 reads a register that I_1 writes to). Some processors are so clever that they calculate on the fly which instructions can be safely executed in parallel. Platinium however expects this information to be explicitly specified. A special marker, called simply a stop, inserted between two instructions indicates that some instructions after the stop are possibly dependent on some instructions before the stop. In other words instructions between two successive stops can be executed in parallel and there should not be dependencies between them.

Another interesting feature of Platinium is that an instruction sequence must be split into groups of one, two or three successive instructions. Each group has to be packed into a container called a bundle. Each bundle has 3 slots and a single instruction can be put into each slot, however some slots may stay empty. Each instruction is categorized into one of 10 instruction types denoted by consecutive capital letters from A to J (instructions of the same type have similar functionality, for example type A groups integer arithmetic instructions and type F groups instructions). Only instructions of certain types are allowed to be packed into one bundle. A template specifies one permissible combination of instruction types within a bundle. A template can also specify a position of a stop in the middle of a bundle (there is at most one such stop allowed). In addition, stops are allowed between any two adjoining bundles. A set of templates is called a bundling profile. When packing instructions into bundles, one has to use templates from bundling profile only.

Although Platinium is equipped with an instruction cache it was found that for maximal performance it is most crucial to pack instructions as densely as possible. Second important thing is to use a small number of stops.

Your task is to write a program for bundling Platinium instructions. For the sake of simplicity we assume that the instructions cannot be reordered.

Task

Write a program that:

- reads a bundling profile and a sequence of instructions,
- computes the minimal number of bundles into which the sequence can be packed without breaking the dependencies and the minimal number of all stops that are required for the minimal number of bundles,
- writes the result.

Input

The input begins with the integer z, the number of test cases. Then z test cases follow.

The first line of each test case descripition contains two integers t and n separated by a single space. Integer t (1 <= t <= 1500) is the number of templates in the bundling profile. Integer n (1 <= n <= 100000) is the number of instructions to be bundled.

Each of the next t lines specifies one template and contains 3 capital letters t_1, t_2, t_3 with no spaces in between followed by a space and an integer p. Letter t_i (A < = t_i <= J) is an instruction type allowed in the i-th slot. Integer p (0 <= p <= 2) is the index of the slot after which the stop is positioned (0 means no stop within the bundle).

Each of the next n lines specifies one instruction. The i-th line of these n lines contains one capital letter c_i and an integer d_i, separated by a single space. Letter c_i (A <= c_i <=J) is the type of the i-th instruction. Integer d_i (0 < = d_i < i) is the index of the last instruction (among the previous ones) that the i-th instruction is dependent on (0 means that the instruction is not dependent on any former instruction).

You can assume that for each instruction type c describing an instruction in the instruction sequence there is at least one template containing c.

Output

For each test case, the first and only line of the output contains two integers b and s. Integer b is the minimal number of bundles in a valid packing. Integer s is the minimal number of all stops that are required for the minimal number of bundles.

Example

Sample input:

```
1
4 9
ABB 0
BAD 1
AAB 0
ABB 2
B 0
B 1
A 1
A 1
B 4
D 0
A 0
B 3
B 0
```

Sample output:

```
4 3
```

Warning: large Input/Output data, be careful with certain languages

---

Added by: Adrian Kosowski
Date: 2004-06-26
Time limit: 7s
Source limit: 50000B
Languages: All
Resource: ACM Central European Programming Contest, Warsaw 2003

SPOJ Problem Set (classical)

84. Shortcut

Problem code: SHORTCUT

Mirek has a favourite way from home to the university that he traverses every working day. The route consists of sections and each section is a straight segment 10 meters long. Each section is either a straight ahead extension of the previous section or it is perpendicular to the previous section. After traversing each section Mirek takes a small break to admire the beauty of the nature. During his walk he never visits the same place twice.

A sample map

Yesterday Mirek stayed up long in the night at the party and today he got up late from bed. He knows that he will miss the first lecture unless he changes his usual route. He plans to make one shortcut but he wants the shortcut to be as short as possible (well, we can tell you in secret that he doesn't want to be on time, he just wants to calm his conscience). The shortcut must be either a horizontal or vertical segment connecting two break points of Mirek's route.

Please help Mirek find the shortest shortcut.

Task

Write a program that:

- reads Mirek's route,
- computes the shortest shortcut on the route,
- writes the result.

Input

The input begins with the integer t, the number of test cases. Then t test cases follow.

For each test case, the first line of the input contains one integer n (3 <= n <= 250 000) being the number of sections of the route. The second line of the input contains a sequence of n characters N, E, S or W with no spaces in between. Each character is a description of one section of the route. Character N, E, S or W means that Mirek walks 10 meters north, east, south or west respectively. You may assume that at least one shortcut exists for the given route.

Output

The first and only line of the output contains integers l, b, e and character d separated by single spaces. Integer l is the length of the shortest shortcut (measured in 10 m segments). Integers b and e are the numbers of break points where the shortcut begins and ends respectively (we number break points with consecutive integers from 0 for Mirek's home to n for the university). Character d is the direction of the shortcut. If more than one shortcut of the minimal length exists you should output the one that begins earliest on the route. If more than one shortcut of the minimal length begins at the same break point you should output the one that ends furthest on the route.

Example

Sample input:

```
1
12
NNNENNWWWSSW
```

Sample output:

```
2 3 11 W
```

Warning: large Input/Output data, be careful with certain languages

---

Added by: Adrian Kosowski
Date: 2004-06-26
Time limit: 13s
Source limit: 50000B
Languages: All
Resource: ACM Central European Programming Contest, Warsaw 2003

SPOJ Problem Set (classical)

85. Dice Contest

Problem code: DICE1

Everyone loves gambling in the Dicent City. Every Saturday the whole community meets to attend a dice contest. They started a few years ago with a classic six-sided die with 1 to 6 dots displayed on the sides and had a lot of fun.

A die

However they soon got bored and that's why more sophisticated dice are in use nowadays. They put a sticker on each side and write a positive integer on each sticker.

The contest is run on a strip divided into squares in a chessboard-like manner. The strip is 4 squares wide and infinite to the left and to the right (is anyone going to say it can't exist in the real world, huh?). The rows of the strip are numbered from 1 to 4 from the bottom to the top and the columns are numbered by consecutive integers from the left to the right. Each square is identified by a pair (x,y) where x is a column number and y is a row number.

The game begins with a die placed on a square chosen be a contest committee with one-dot side on the top and two-dots side facing the player. To move the die the player must roll the die over an edge to an adjacent (either horizontally or vertically) square. The number displayed on the top of the die after a roll is the cost of the move. The goal of the game is to roll the die from the starting square to the selected target square so that the sum of costs of all moves is minimal.

Task

Write a program that:

- reads the description of a die, a starting square and a target square,
- computes the minimal cost of rolling the die from the starting square to the target square,
- writes the result.

Note: all teams participating in the contest received dice from the organisers.

Input

The input begins with the integer t, the number of test cases. Then t test cases follow.

For each test case the first line of the input contains six integers $l_1, l_2, l_3, l_4, l_5, l_6$ ($1 <= l_i <= 50$) separated by single spaces. Integer l_i is the number written on a side having originally i dots. The second line of the input contains four integers x_1, y_1, x_2, y_2 ($-10^9 <= x_1, x_2 <= 10^9$, $1 <= y_1, y_2 <= 4$) separated by single spaces. Integers x_1, y_1 are the column and the row number of the starting square respectively. Integers x_2, y_2 are the column and the row number of the target square respectively.

Output

For each test case the first and the only line of the output should contain the minimal cost of rolling the die from the starting square to the target square.

Example

Sample input:

```
1
1 2 8 3 1 4
-1 1 0 2
```

Sample output:

```
7
```

Added by: Adrian Kosowski
Date: 2004-06-26
Time limit: 13s
Source limit: 50000B
Languages: All
Resource: ACM Central European Programming Contest, Warsaw 2003

SPOJ Problem Set (classical)

86. November Rain

Problem code: RAIN1

Contemporary buildings can have very complicated roofs. If we take a vertical section of such a roof it results in a number of sloping segments. When it is raining the drops are falling down on the roof straight from the sky above. Some segments are completely exposed to the rain but there may be some segments partially or even completely shielded by other segments. All the water falling onto a segment as a stream straight down from the lower end of the segment on the ground or possibly onto some other segment. In particular, if a stream of water is falling on an end of a segment then we consider it to be collected by this segment.

Rooftops

For the purpose of designing a piping system it is desired to compute how much water is down from each segment of the roof. To be prepared for a heavy November rain you should count one liter of rain water falling on a meter of the horizontal plane during one second.

Task

Write a program that:

- reads the description of a roof,
- computes the amount of water down in one second from each segment of the roof,
- writes the results.

Input

The input begins with the integer t, the number of test cases. Then t test cases follow.

For each test case the first line of the input contains one integer n (1 <= n < = 40000) being the number of segments of the roof. Each of the next n lines describes one segment of the roof and contains four integers x_1, y_1, x_2, y_2 (0 <= x_1, y_1, x_2, y_2 < = 1000000, x_1 < x_2, y_1<>y_2) separated by single spaces. Integers x_1, y_1 are respectively the horizontal position and the height of the left end of the segment. Integers x_2, y_2 are respectively the horizontal position and the height of the right end of the segment. The segments don't have common points and there are no horizontal segments. You can also assume that there are at most 25 segments placed above any point on the ground level.

Output

For each test case the output consists of n lines. The i-th line should contain the amount of water (in liters) down from the i-th segment of the roof in one second.

Example

Sample input:

```
1
6
13 7 15 6
3 8 7 7
1 7 5 6
5 5 9 3
6 3 8 2
9 6 12 8
```

Sample output:

```
2
4
2
11
0
3
```

Warning: large Input/Output data, be careful with certain languages

---

Added by: Adrian Kosowski
Date: 2004-06-26
Time limit: 13s
Source limit: 50000B
Languages: All
Resource: ACM Central European Programming Contest, Warsaw 2003

SPOJ Problem Set (classical)

87. Football

Problem code: FOOTBALL

Eric has a classic football that is made of 32 pieces of leather: 12 black pentagons and 20 white hexagons. Each pentagon adjoins 5 hexagons and each hexagon adjoins 3 pentagons and 3 hexagons. Eric drew a polygon (i.e. a closed line without intersections) along the edges of the pieces. The polygon divided the ball into two parts and Eric painted one of them green.

Eric's football

He is curious if given a description of the polygon you are able to compute the number of black, white and green pieces?

Task

Write a program that:

- reads the description of a polygon,
- computes the number of black, white and green pieces,
- writes the result.

Contest note: the first accepted solution will be awarded with the original football used for preparing the problem, signed by Eric, the author of the problem!

SPOJ note: the first accepted solution will be awarded some other sphere, without anybody's signatures, sent in PNG format to the author's email address [the offer is invalid, the sphere has already been presented to Robin Nittka, University of Ulm, Germany].

Input

The input begins with the integer t, the number of test cases. Then t test cases follow.

For each test case, the first line of the input contains one integer n being the number of vertices of the polygon. The second line of the input contains n integers a_1, a_2,..., a_n separated by single spaces. Integer a_i (equal 1 or 2) is the number of green pieces adjoining the i-th vertex of the polygon. The side of the polygon connecting the n-th and the first vertex always lies between two hexagons.

Output

For each test case the first and only line of the output contains three integers b, w and g - the numbers of black, white and green pieces respectively.

Example

Sample input:

```
1
21
1 2 1 2 1 2 1 1 1 2 2 1 1 1 1 2 2 2 1 1 1
```

Sample output:

```
11 15 6
```

Added by: Adrian Kosowski
Date: 2004-06-26
Time limit: 2s
Source limit: 50000B
Languages: All
Resource: ACM Central European Programming Contest, Warsaw 2003 (E. Kopczynski)

SPOJ Problem Set (classical)

88. Which is Next

Problem code: TREE2

Every computer science student knows binary trees. Here is one of many possible definitions of binary trees. Binary trees are defined inductively. A binary tree t is either an external node (leaf) o or an ordered pair $t = (t_1, t_2)$ representing an internal node * with two subtrees attached, left subtree t_1 and right subtree t_2. Under this definition the number of nodes in any binary tree is odd. Given an odd integer n let B(n) denote the set of all binary trees with n nodes, both internal and external. For instance B(1) consists of only one tree o , B(3) = {(o , o)} and B(5) = {(o , (o , o)), ((o , o), o)}. The trees of B(5) are depicted in the figure below.

The trees B(5)

Denote by |t| the number of nodes in a tree t. Given a tree t we define its unique integer identifier N (t) as follows:

- $N(o) = 0$
- $N(t_1, t_2) = 2^{|t_1|+|t_2|} + 2^{|t_2|} * N(t_1) + N(t_2)$

For instance, $N(o, o) = 2^2 + 2^1 * 0 + 0 = 4$, $N(o, (o, o)) = 2^4 + 2^3 * 0 + 4 = 20$,
$N((o, o), o) = 2^4 + 2^1 * 4 + 0 = 24$.

Consider the following linear order on all binary trees:

1) $o <= t$
2) $(t_1, t_2) <= (u_1, u_2)$ when $t_1 < u_1$, or $t_1 = u_1$ and $t_2 <= u_2$

In this order a single leaf o is the smallest tree and given two nonleaf trees, the smaller one is that with the smaller left tree, if the left subtrees are different, and that with the smaller right subtree, otherwise. Hence for instance (o , (o , o)) < ((o , o), o), since we have o < (o , o). Assume now that the trees in B(n) were sorted using the relation <=. Then, for each tree t in B(n) we define the successor of t as the tree that immediately follows t in B(n). If t is the largest one in B(n) then the successor of t is the smallest tree in set B(n). For instance, the successor of (o , o) in B(3) is the same tree (o , o) and the successor of (o , (o , o)) in B(5) is ((o , o), o). Given the integer identifier of some tree t can you give the identifier of the successor of t in B(|t|)?

Task

Write a program that:

- reads the identifier of some binary tree t,
- computes the identifier of the successor of t in B(|t|),
- writes the result.

Input

The input begins with the integer t, the number of test cases. Then t test cases follow.

For each test case the first and only line of the input contains one integer n ($0 <= n < = 2^{30}$) - the identifier of some binary tree t.

Output

For each test case the first and only line of the output should contain one integer s - the identifier of the successor of t in B(|t|).

Example

Sample input:

```
1
20
```

Sample output:

```
24
```

Added by: Adrian Kosowski
Date: 2004-06-26
Time limit: 2s
Source limit: 50000B
Languages: All
Resource: ACM Central European Programming Contest, Warsaw 2003

SPOJ Problem Set (classical)

89. Hang or not to hang

Problem code: HANGLET

Little Tom is learning how to program. He has just written some programs but is afraid to run them, because he does not know if they will ever stop. Please write a program to help him. This task is not as easy as it may seem, because Tom's programs are possibly not deterministic. Given a program written by Tom, your program should tell him whether his program can stop and if so, what is the shortest possible time before it stops.

Tom's computer consists of 32 1-bit registers and the program consists of n instructions. The registers are numbered from 0 to 31 and the instructions are numbered from 0 to n-1.

Below, MEM[a] stands for the contents of the a-th register, 0 <= a, b < 32, 0 <= x < n, 0 <= c <= 1.

The instruction set is as follows:

Instruction	Semantics
AND a b	MEM[a] := MEM[a] and MEM[b]
OR a b	MEM[a] := MEM[a] or MEM[b]
XOR a b	MEM[a] := MEM[a] xor MEM[b]
NOT a	MEM[a] := not MEM[a]
MOV a b	MEM[a] := MEM[b]
SET a c	MEM[a] := c
RANDOM a	MEM[a] := random value (0 or 1)
JMP x	jump to the instruction with the number x
JZ x a	jump to the instruction with the number x if MEM[a] = 0
STOP	stop the program

The last instruction of a program is always STOP (although there can be more than one STOP instruction). Every program starts with the instruction number 0. Before the start, the contents of the registers can be arbitrary values. Each instruction (including STOP) takes 1 processor cycle to execute.

Task

Write a program that:

- reads the program,
- computes the shortest possible running time of the program,
- writes the result.

Input

The input begins with the integer t, the number of test cases. Then t test cases follow.

For each test case the first line of the input contains an integer n (1 <= n <= 16) being the number of instructions of the program. Each of the next n lines contains one instruction of the program in the format given above. You may assume that the only white characters in the program are single spaces between successive tokens of each instruction.

Output

For each test case the first and only line of the output should contain the shortest possible running time of the program, measured in processor cycles. If the program cannot stop, output should contain the word HANGS.

Example

Sample input:

```
2
5
SET 0 1
JZ 4 0
RANDOM 0
JMP 1
STOP
5
MOV 3 5
NOT 3
AND 3 5
JZ 0 3
STOP
```

Sample output:

```
6
HANGS
```

Added by: Adrian Kosowski
Date: 2004-06-26
Time limit: 3s
Source limit: 50000B
Languages: All
Resource: ACM Central European Programming Contest, Warsaw 2003

SPOJ Problem Set (classical)

90. Minimizing maximizer

Problem code: MINIMAX

The company Chris Ltd. is preparing a new sorting hardware called Maximizer. Maximizer has n inputs numbered from 1 to n. Each input represents one integer. Maximizer has one output which represents the maximum value present on Maximizer's inputs.

Maximizer is implemented as a pipeline of sorters Sorter(i_1, j_1), ... , Sorter(i_k, j_k). Each sorter has n inputs and n outputs. Sorter(i, j) sorts values on inputs i, i+1,... , j in non-decreasing order and lets the other inputs pass through unchanged. The n-th output of the last sorter is the output of the Maximizer.

An intern (a former ACM contestant) observed that some sorters could be excluded from the pipeline and Maximizer would still produce the correct result. What is the length of the shortest subsequence of the given sequence of sorters in the pipeline still producing correct results for all possible combinations of input values?

Task

Write a program that:

- reads a description of a Maximizer, i.e. the initial sequence of sorters in the pipeline,
- computes the length of the shortest subsequence of the initial sequence of sorters still producing correct results for all possible input data,
- writes the result.

Input

The input begins with the integer t, the number of test cases. Then t test cases follow.

For each test case the first line of the input contains two integers n and m (2 <= n <= 50000, 1 <= m <= 500000) separated by a single space. Integer n is the number of inputs and integer m is the number of sorters in the pipeline. The initial sequence of sorters is described in the next m lines. The k-th of these lines contains the parameters of the k-th sorter: two integers i_k and j_k (1 <= i_k < j_k <= n) separated by a single space.

Output

For each test case the output consists of only one line containing an integer equal to the length of the shortest subsequence of the initial sequence of sorters still producing correct results for all possible data.

Example

Sample input:

```
1
40 6
20 30
1 10
10 20
20 30
15 25
30 40
```

Sample output:

```
4
```

Warning: enormous Input/Output data, be careful with certain languages

---

Added by: Adrian Kosowski
Date: 2004-06-26
Time limit: 13s
Source limit: 50000B
Languages: All
Resource: ACM Central European Programming Contest, Warsaw 2003

SPOJ Problem Set (classical)

91. Two squares or not two squares

Problem code: TWOSQRS

Given integer n decide if it is possible to represent it as a sum of two squares of integers.

Input

First line of input contains one integer c<=100 - number of test cases. Then c lines follow, each of them consisting of exactly one integer 0<=n<=10^12.

Output

For each test case output Yes if it is possible to represent given number as a sum of two squares and No if it is not possible.

Example

```
Input:
10
1
2
7
14
49
9
17
76
2888
27

Output:
Yes
Yes
No
No
Yes
Yes
Yes
No
Yes
No
```

Added by: Pawel Gawrychowski
Date: 2004-06-29
Time limit: 2s
Source limit:50000B
Languages: All

SPOJ Problem Set (classical)

92. Cutting off Squares

Problem code: CUTSQRS

Two players take it in turns to cut off squares from a rectangle. If the lengths of the sides of the rectangle are *a* and *b* (a<=b) at the beginning of a player's turn, he may cut off as many squares with a side of length *a* as he likes (but at least 1 square), provided the square he is cutting off has at least three of its sides lying on the sides of the rectangle he is trimming. After every cut, the cut off square is removed from the rectangle. When the last part of the rectangle is removed, the game ends and the person who cut it off wins.

Michael, a friend of the players', is taking down a log of the games they are playing in the form of a sequence of consecutive numbers, each number denoting how many squares a player cut off in his turn. Since the game is rather slow, Michael is getting a little bored and he has started writing a detailed analysis of the game in his notebook. For given starting dimensions a and b, he always writes down:

- the number of different possible game sequences,
- the number of different possible game sequences in which the starting player wins,
- the word 'first' if the starting player can win (provided he does not make any mistakes) regardless of what the other player does, and the word 'second' in all other cases.

After writing for several hours Michael began to worry whether he had enough room left in his notebook for all the information he wanted to write down. Please help him answer this question.

Input

An integer t denoting the number of test cases, (t<=10000) followed by t pairs of integers a, b, ($1<=a<=b<=10^9$) given in separate lines.

Output

For each test case, output the number of characters Michael has to write down (excluding spaces).

Example

Sample input:

```
2
1 1
2 3
```

Sample output:

```
7
8
```

(In the first case Michael has to write '1 1 first', in the second case '2 1 second'.)

Added by: Adrian Kosowski
Date: 2004-06-22
Time limit: 3s
Source limit: 50000B
Languages: All
Resource: DASM Programming League 2004 (problemset 1)

SPOJ Problem Set (classical)

94. Numeral System of the Maya

Problem code: MAYA

The Maya lived in Central America during the first millennium. In many regards, they consituted one of the most developed and most fascinating cultures of this epoch. Even though draught animals and the wheel were unknown to the Mayas, they excelled in the fields of weaving, architecture and pottery. But truely breath-taking were their achievements in the fields of astronomy and mathematics. Whilst Europe was trudging through the dark Middle Ages, the Maya determined the solar year to 365.242 days (modern-day measurement: 365.242198) and the lunar cycle to 29.5302 days (modern-day measurement: 29.53059). Such astonishingly precise findings were hardly possible without a powerful numeral system. In this task we will explore the Maya's numeral system.

Maya priests and astronomers used a numerical system to the base of 20. Unusual to their time, their system also included the concepts of digits and of the zero. Both concepts were completely unknown to the Europeans at this time. The first nineteen numbers of the vigesimal system were represented by dots and dashes according to the following table:

[IMAGE]

The zero was written down as a symbol resembling a shell. Multi-digit numbers (i.e. the numbers bigger than 19) were written in vertical arrangement, with the highest-value digit on top. For example, the number 79 was written as

[IMAGE]

As can be seen, the second digit possesses a value of 20.

Due to an interference of the two calendar systems of the Maya, the third digit did not hold the value 400 (20x20), as would be expected, but 360. All the following digits were again treated regularly, i.e. the fourth digit counted 7200 (360x20), the fifth 144000 (7200x20), and so on.

Hence, the number 13495 (=1x7200+17x360+8x20+15) was written as follows:

[IMAGE]

Write a program to convert Maya numbers to decimal numbers!

Input

The input file contains a list of numbers written down in Maya fashion. Of course, dots are represented as points (.), and dashes are represented as hyphens (-). The zero digit, the shell symbol, is written as a capital letter S (S). Description of a Maya number starts with *n* - the number of the Maya digits. The following *n* lines contain one digit each. One digit is written from top to bottom using spaces as vertical separators.

One number will not have more than seven digits. Each two numbers are separated by a blank line. Input terminates with $n = 0$

Output

Your program has to output the value of the number in the input file in the nowadays more common decimal system. One number per line.

Example

Sample input:

```
1
..

5
... -
. - -
S
S
S

0
```

Sample output:

```
2
1231200
```

Added by: Michał Czuczman
Date: 2004-07-11
Time limit: 3s
Source limit: 50000B
Languages: All
Resource: Swiss Olympiad in Informatics 2004

SPOJ Problem Set (classical)

95. Street Parade

Problem code: STPAR

For sure, the love mobiles will roll again on this summer's street parade. Each year, the organisers decide on a fixed order for the decorated trucks. Experience taught them to keep free a side street to be able to bring the trucks into order.
The side street is so narrow that no two cars can pass each other. Thus, the love mobile that enters the side street last must necessarily leave the side street first. Because the trucks and the ravers move up closely, a truck cannot drive back and re-enter the side street or the approach street.
You are given the order in which the love mobiles arrive. Write a program that decides if the love mobiles can be brought into the order that the organisers want them to be.

Input

There are several test cases. The first line of each test case contains a single number n, the number of love mobiles. The second line contains the numbers 1 to n in an arbitrary order. All the numbers are separated by single spaces. These numbers indicate the order in which the trucks arrive in the approach street. No more than 1000 love mobiles participate in the street parade. Input ends with number 0.

Output

For each test case your program has to output a line containing a single word `yes` if the love mobiles can be re-ordered with the help of the side street, and a single word `no` in the opposite case.

Example

Sample input:

```
5
5 1 2 4 3
0
```

Sample output:

```
yes
```

Illustration

The sample input reflects the following situation:
[IMAGE]

The five trucks can be re-ordered in the following way:
[IMAGE] [IMAGE] [IMAGE] [IMAGE] [IMAGE]

Added by:	Patryk Pomykalski
Date:	2004-07-01
Time limit:	2s
Source limit:	50000B
Languages:	All
Resource:	Swiss Olympiad in Informatics 2004

SPOJ Problem Set (classical)

96. Shopping

Problem code: SHOP

Crowd in the supermarket The old tube screen to your computer turned out to be the cause of your chronic headaches. You therefore decide to buy one of these new flat TFT monitors. At the entrance of the computer shop you see that it is quite full with customers.

In fact, the shop is rather packed with customers and moving inside involves a certain amount of elbowing. Since you want to return home quickly to complete your half finished SPOJ tasks, you want to sidestep the crowd as much as possible. You examine the situation somewhat closer and realise that the crowding is less in some parts of the shop. Thus, there is reason for hope that you can reach your goal in due time, provided that you take the shortest way. But which way is the shortest way?

You sketch the situation on a piece of paper but even so, it is still a tricky affair. You take out your notebook from your pocket and start to write a program which will find the shortest way for you.

Input

The first line of the input specifies the width w and height h of the shop. Neither dimension exceeds 25.

The following h lines contain w characters each. A letter X symbolises a shelf, the letter S marks your starting position, and the letter D marks the destination (i.e. the square in front of the monitors). All free squares are marked with a digit from 1 to 9, meaning the number of seconds needed to pass this square.

There are many test cases separated by an empty line. Input terminates with width and height equal 0 0.

Output

Your program is to output the minimum number of seconds needed to reach to destination square. Each test case in a separate line. Movements can only be vertical and horizontal. Of course, all movements must take place inside the grid. There will always be a way to reach the destination.

Example

Sample input:

```
4 3
X1S3
42X4
X1D2

5 5
S5213
2X2X5
51248
4X4X2
```

```
1445D

0 0
```

Sample output:

```
4
23
```

Added by:	Michał Czuczman
Date:	2004-07-01
Time limit:	3s
Source limit:	50000B
Languages:	All
Resource:	Swiss Olympiad in Informatics 2004

SPOJ Problem Set (classical)

97. Party Schedule

Problem code: PARTY

You just received another bill which you cannot pay because you lack the money.
Unfortunately, this is not the first time to happen, and now you decide to investigate the cause of your constant monetary shortness. The reason is quite obvious: the lion's share of your money routinely disappears at the entrance of party localities.
You make up your mind to solve the problem where it arises, namely at the parties themselves. You introduce a limit for your party budget and try to have the most possible fun with regard to this limit.
You inquire beforehand about the entrance fee to each party and estimate how much fun you might have there. The list is readily compiled, but how do you actually pick the parties that give you the most fun and do not exceed your budget?
Write a program which finds this optimal set of parties that offer the most fun. Keep in mind that your budget need not necessarily be reached exactly. Achieve the highest possible fun level, and do not spend more money than is absolutely necessary.

Input

The first line of the input specifies your party budget and the number n of parties.
The following n lines contain two numbers each. The first number indicates the entrance fee of each party. Parties cost between 5 and 25 francs. The second number indicates the amount of fun of each party, given as an integer number ranging from 0 to 10.
The budget will not exceed 500 and there will be at most 100 parties. All numbers are separated by a single space.
There are many test cases. Input ends with 0 0.

Output

For each test case your program must output the sum of the entrance fees and the sum of all fun values of an optimal solution. Both numbers must be separated by a single space.

Example

Sample input:

```
50 10
12 3
15 8
16 9
16 6
10 2
21 9
18 4
12 4
17 8
18 9
```

```
50 10
13 8
19 10
16 8
12 9
10 2
12 8
13 5
15 5
11 7
16 2

0 0
```

Sample output:

```
49 26
48 32
```

Added by: Patryk Pomykalski
Date: 2004-07-01
Time limit: 3s
Source limit: 50000B
Languages: All
Resource: Swiss Olympiad in Informatics 2004

SPOJ Problem Set (classical)

98. Dance Floor

Problem code: DFLOOR

You recently watched a video clip in which a singer danced on a grid of colourful tiles enlightened from below. Each step on a tile flipped the tile's state, i.e. light on or off. In addition to that, all the neighbouring tiles flipped their states, too.
In this task, you are supposed to come up with a short program that decides if it is possible for the singer to switch on the lights of all the tiles, provided that he dances on the appropriate tiles.
The dance floor has rectangular shape. At the beginning, some of the tiles are already alight. Your program may temporarily switch off some tiles, if it deems that necessary to reach its goal. Stepping on a tile toggles its own state as well as the states of the four neighbouring tiles directly above, below, to the left and to the right. Of course, in the case of a peripheral tile, there will be only three or two neighbouring tiles.
Here comes an example:
[IMAGE]
If the dancer steps on the tile indicated by the brown shoe, all the tiles within the white area change their states. The resulting dance floor is depicted on the right.
You may assume that the singer is fit enough to jump from any tile to any other tile, even if the destination tile lies on the opposite side of the dance floor.

Input

There are several test cases. The first line of each case contains two integer numbers x and y, indicating the width and the height of the dance floor grid. The numbers are separated by a single space and satisfy $3 <= x,y <= 15$.
The following y lines containing $x$characters each describe the initial on/off states of the tiles. A zero means "the tile is switched off", a one digit means "the tile is alight".
Input ends with 0 0.

Output

For each test case your program should output the number of steps needed to switch all the lights on, followed by exactly that many lines with two space-separated numbers i and j. Each individual line commands the singer to step on the i-th tile of the j-th row. Starting with the situation of the input file and executing all the commands in the output file, all the tiles must be switched on.
If more than one solution exist, your program should output an arbitrary one of them. If, on the other hand, no solution exists, your program should write the number "-1".

Example

Sample input

```
4 3
0111
1010
1000

0 0
```

Sample output

```
3
1 2
1 3
4 3
```

Added by:	Michał Czuczman
Date:	2004-07-01
Time limit:	3s
Source limit:	50000B
Languages:	All
Resource:	Swiss Olympiad in Informatics 2004

SPOJ Problem Set (classical)

99. Bus

Problem code: BUS

[IMAGE] The city Buscelona (as the name suggests) has a great bus transport system. All buses have circular lines. The bus drivers in Buscelona like to chat. Fortunately most bus lines have some stops in common. If a bus driver meets a colleague on a bus stop they chat a bit and exchange all news they know.

The operation of buses is highly synchronized. The time necessary to get from one stop to the next stop is always exactly 1 minute.

Each morning each bus driver has some important news that only he knows. When a busdriver meets a colleague he will tell him all news he knows. If two bus drivers share the same start station, they will exchange their news there already (before they start working). Note that exchanging news and stopping does not take any time.

Input

The first line of a test case contains the number of bus lines n ($0 < n < 50$). The following n lines start with a number s ($0 < s < 50$) indicating the stops of a busline. On the same line follow s numbers representing a bus station each. A bus starts at the first station. When a bus reaches the last station, the bus will drive to the first station again.

There are many test cases separated by an empty line. Input data terminates with $n = 0$.

Output

For each test case you should output the time in minutes which it takes until all bus drivers know all news. If that never happens, your program should write the word "NEVER" (without quotes).

Example

Sample input:

```
3
3 1 2 3
3 2 3 1
4 2 3 4 5

2
2 1 2
2 5 8

0
```

Sample output:

```
12
NEVER
```

Added by: Michał Czuczman
Date: 2004-07-03
Time limit: 7s
Source limit: 50000B
Languages: All
Resource: Swiss Olympiad in Informatics 2004

SPOJ Problem Set (classical)

100. Tower of Babylon

Problem code: BABTWR

[IMAGE] Apart from the Hanging Gardens the Babylonians (around 3000-539 b.c.) built the Tower of Babylon as well. The tower was meant to reach the sky, but the project failed because of a confusion of language imposed from much higher above.

For the 2638th anniversary a model of the tower will be rebuilt. n different types of blocks are available. Each one of them may be duplicated as many times as you like. Each type has a height y, a width x and a depth z. The blocks are to be stacked one upon eachother so that the resulting tower is as high as possible. Of course the blocks can be rotated as desired before stacking. However for reasons of stability a block can only be stacked upon another if *both* of its baselines are shorter.

Input

The number of types of blocks n is located in the first line of each test case. On the subsequent n lines the height y_i, the width x_i and the depth z_i of each type of blocks are given. There are never more than 30 different types available.

There are many test cases, which come one by one. Input terminates with n = 0.

Output

For each test case your program should output one line with the height of the highest possible tower.

Example

Sample input:

```
5
31 41 59
26 53 58
97 93 23
84 62 64
33 83 27
1
1 1 1
0
```

Sample output:

```
342
1
```

Added by:	Michał Czuczman
Date:	2004-07-06
Time limit:	3s
Source limit:	50000B
Languages:	All
Resource:	Swiss Olympiad in Informatics 2004

SPOJ Problem Set (classical)

101. Fishmonger

Problem code: FISHER

A Fishmonger A fishmonger wants to bring his goods from the port to the market. On his route he has to traverse an area with many tiny city states. Of course he has to pay a toll at each border.

Because he is a good business man, he wants to choose the route in such a way that he has to pay as little money for tolls as possible. On the other hand, he has to be at the market within a certain time, otherwise his fish start to smell.

Input

The first line contains the number of states *n* and available time *t*. The first state is the port, the last state is the market. After this line there are *n* lines with *n* numbers each, specifying for each state the travel time to the *i*-th state. This table is terminated with an empty line. The table of the tolls follows in the same format.

n is at least 3 and at most 50. The time available is less than 1000. All numbers are integers.

There are many test cases separated by an empty line. Input terminates with number of states and time equal 0 0.

Output

For each test case your program should print on one line the total amount of tolls followed by the actual travelling time.

Example

Sample input:

```
4 7
0 5 2 3
5 0 2 3
3 1 0 2
3 3 2 0

0 2 2 7
2 0 1 2
2 2 0 5
7 2 5 0

0 0
```

Sample output:

```
6 6
```

This corresponds to the following situation, the connections are labeled with (time, toll):

[IMAGE]

Added by:	Michał Czuczman
Date:	2004-07-07
Time limit:	3s
Source limit:	50000B
Languages:	All
Resource:	Swiss Olympiad in Informatics 2004

SPOJ Problem Set (classical)

102. GX Light Pipeline Inc

Problem code: LITEPIPE

The GX Light Pipeline Inc. started to prepare bent pipes for the new transgalactic light pipeline. However during the design of the pipeline they ran into the problem of determing how far the light can reach inside the pipe. In order to improve your scarce budget you decided to fill a summer job at the GX Light Pipeline Inc. Now it's your task to create a program which computes how far the light reaches in the pipeline.

The pipeline consists of seamlessly welded together segments made of non-reflecting opaque materials. The upper points of the pipe contour are described by a sequence of points $[x_1, y_1]$, $[x_2, y_2]$, $[x_3, y_3]$, ..., $[x_n, y_n]$, where $x_k < x_{k+1}$. The bottom points of the pipe contour are the same points with y-coordinate decreased by 1.

The company wants to find the points with maximal x-coordinate that the light will reach. The light is emitted by a segment source with endpoints $[x_1, y_1]$ and $[x_1, y_1\text{-}1]$ (endpoints are emitting light too). Assume that the light is not bent at the pipe bent points and the bent points do not stop the light beam.

[IMAGE]

Input

Each test case starts with the number of bent points n. Each of the next n lines contains a pair of real values x_i, y_i separated by space.

The number of bent points never excedes 200.

There are many test cases. Input terminates with $n = 0$.

Output

For each test case your program should output on a single line the maximal x-coordinate of the point where the light can reach from the source segment, written with precision of two decimal places. If the light goes trough all the pipe, your program should output x_n.

Example

Sample input:

```
4
0.00 1.00
2.00 2.00
4.00 1.00
6.00 4.00
0
```

Sample output:

```
4.67
```

Added by: Michał Czuczman
Date: 2004-07-11
Time limit: 3s
Source limit: 50000B
Languages: All
Resource: Swiss Olympiad in Informatics 2004

SPOJ Problem Set (classical)

104. Highways

Problem code: HIGH

In some countries building highways takes a lot of time... Maybe that's because there are many possiblities to construct a network of highways and engineers can't make up their minds which one to choose. Suppose we have a list of cities that can be connected directly. Your task is to count how many ways there are to build such a network that between every two cities there exists exactly one path. Two networks differ if there are two cities that are connected directly in the first case and aren't in the second case. At most one highway connects two cities. No highway connects a city to itself. Highways are two-way.

Input

The input begins with the integer t, the number of test cases (equal to about 1000). Then t test cases follow. The first line of each test case contains two integers, the number of cities (1<=n<=12) and the number of direct connections between them. Each next line contains two integers a and b, which are numbers of cities that can be connected. Cities are numbered from 1 to n. Consecutive test cases are separated with one blank line.

Output

The number of ways to build the network, for every test case in a separate line. Assume that when there is only one city, the answer should be 1. The answer will fit in a signed 64-bit integer.

Example

Sample input:

```
4
4 5
3 4
4 2
2 3
1 2
1 3

2 1
2 1

1 0

3 3
1 2
2 3
3 1
```

Sample output:

```
8
1
1
3
```

Added by: Piotr Łowiec
Date: 2004-07-02
Time limit: 7s
Source limit:50000B
Languages: All

SPOJ Problem Set (classical)

105. Alice and Bob

Problem code: ALICEBOB

This is a puzzle for two persons, let's say Alice and Bob. Alice draws an n-vertex convex polygon and numbers its vertices with integers 1, 2, ... , n in an arbitrary way. Then she draws a number of noncrossing diagonals (the vertices of the polygon are not considered to be crossing points). She informs Bob about the sides and the diagonals of the polygon but not telling him which are which. Each side and diagonal is specified by its ends. Bob has to guess the order of the vertices on the border of the polygon. Help him solve the puzzle.

If n = 4 and (1,3), (4,2), (1,2), (4,1), (2,3) are the ends of four sides and one diagonal then the order of the vertices on the border of this polygon is 1, 3, 2, 4 (with the accuracy to shifting and reversing).

Task

Write a program that:

- reads the description of sides and diagonals given to Bob by Alice,
- computes the order of the vertices on the border of the polygon,
- writes the result.

Input

The first line of the input contains exactly one positive integer d equal to the number of data sets, 1 <= d <= 20. The data sets follow.

Each data set consists of exactly two consecutive lines.

The first of those lines contains exactly two integers n and m separated by a single space, 3 <= n <= 10 000, 0 <= m <= n-3. Integer n is the number of vertices of a polygon and integer m is the number of its diagonals, respectively.

The second of those lines contains exactly 2(m+n) integers separated by single spaces. Those are ends of all sides and some diagonals of the polygon. Integers a_j, b_j on positions 2j-1 and 2j, 1 <= j <= m+n, 1 <= a_j <= n, 1 <= b_j <= n, a_j <> b_j, specify ends of a side or a diagonal. The sides and the diagonals can be given in an arbitrary order. There are no duplicates. Alice does not cheat, i.e. the puzzle always has a solution.

Output

Line i, 1 <= i <= d, should contain a sequence of n integers separated by single spaces - a permutation of 1, 2, ... , n, i.e. the numbers of subsequent vertices on the border of the polygon from the i-th data set, the sequence should always start from 1 and its second element should be the smaller vertex of the two border neighbours of vertex 1.

Example

Sample input:

```
1
4 1
1 3 4 2 1 2
4 1 2 3
```

Sample output:

```
1 3 2 4
```

Warning: large Input/Output data, be careful with certain languages

---

Added by: Adrian Kosowski
Date: 2004-07-02
Time limit: 7s
Source limit: 50000B
Languages: All
Resource: ACM Central European Programming Contest, Warsaw 2001

SPOJ Problem Set (classical)

106. Binary Stirling Numbers

Problem code: BINSTIRL

The Stirling number of the second kind S(n, m) stands for the number of ways to partition a set of n things into m nonempty subsets. For example, there are seven ways to split a four-element set into two parts: {1, 2, 3} u {4}, {1, 2, 4} u {3}, {1, 3, 4} u {2}, {2, 3, 4} u {1}, {1, 2} u {3, 4}, {1, 3} u {2, 4}, {1, 4} u {2, 3}.

There is a recurrence which allows you to compute S(n, m) for all m and n.
S(0, 0) = 1,
S(n, 0) = 0, for n > 0,
S(0, m) = 0, for m > 0,
S(n, m) = m*S(n-1, m) + S(n-1, m-1), for n, m > 0.

Your task is much "easier". Given integers n and m satisfying 1 <= m <= n, compute the parity of S(n, m), i.e. S(n, m) mod 2.

For instance, S(4, 2) mod 2 = 1.

Task

Write a program that:

- reads two positive integers n and m,
- computes S(n, m) mod 2,
- writes the result.

Input

The first line of the input contains exactly one positive integer d equal to the number of data sets, 1 <= d <= 200. The data sets follow.

Line i + 1 contains the i-th data set - exactly two integers n_i and m_i separated by a single space, $1 <= m_i <= n_i <= 10^9$.

Output

The output should consist of exactly d lines, one line for each data set. Line i, 1 <= i <= d, should contain 0 or 1, the value of $S(n_i, m_i)$ mod 2.

Example

Sample input:

```
1
4 2
```

Sample output:

```
1
```

Added by: Adrian Kosowski
Date: 2004-07-02
Time limit: 3s
Source limit: 50000B
Languages: All
Resource: ACM Central European Programming Contest, Warsaw 2001

SPOJ Problem Set (classical)

107. Calendar of the Maya

Problem code: MAYACAL

The Classical Maya civilization developed in what is today southern Mexico, Guatemala, Belize and northern Honduras. During its height they developed a sophisticated system for time keeping which they used both to record history and for divinatory rituals. Their calendar consisted of 3 components. the Tzolkin, the Haab and the Long Count.

For divinatory purposes the Maya used the Tzolkin which was composed of 20 day names to which numeric coefficients from 1 to 13 were attached giving a total of 260 distinct combinations. This is the size of the Tzolkin, or ritual, year. From Spanish colonial sources, we know the names of the days: `Imix, Ik, Akbal, Kan, Chikchan, Kimi, Manik, Lamat, Muluk, Ok, Chuen, Eb, Ben, Ix, Men, Kib, Kaban, Etznab, Kawak, Ajaw`. The sequence of days developed as follows (starting for example at 9 Imix):

```
9 Imix, 10 Ik, 11 Akbal, 12 Kan, 13 Chikchan, 1 Kimi, 2 Manik, ...
```

The Haab calendar was an astronomical one. It had 365 days divided into 19 months each with 20 days, except the last one which had only 5 days. In a manner similar to the Tzolkin each month name had a number from 1 to 20 indicating the day number within the month. Again, from Spanish colonial sources, we know the names of the months: `Pohp, Wo, Sip, Zotz, Sek, Xul, Yaxkin, Mol, Chen, Yax, Sak, Keh, Mak, Kankin, Muan, Pax, Kayab, Kumku, Wayeb`. The month Wayeb had just 5 days and was considered an unlucky time of the year.

The Tzolkin and Haab were combined in the inscriptions to create the Calendar Round, combining the 260 day cycle of the Tzolkin and the 365 day cycle of the Haab. A typical Calendar Round date in the inscriptions might be. 3 Lamat 6 Pax. Note that not all of the combination of days, months and coefficients are possible.

A typical sequence of days in the Calendar Round (starting for example at 3 Lamat 6 Pax):

```
3 Lamat 6 Pax, 4 Muluk 7 Pax, 5 Ok 8 Pax, 6 Chuen 9 Pax, 7 Eb 10 Pax,
8 Ben 11 Pax, 9 Ix 12 Pax, 10 Men 13 Pax, 11 Kib 14 Pax, 12 Kaban 15 Pax,
13 Etznab 16 Pax, 1 Kawak 17 Pax, 2 Ajaw 18 Pax, 3 Imix 19 Pax, 4 Ik 20 Pax,
5 Akbal 1 Kayab, 6 Kan 2 Kayab, ...
```

Finally, at the beginning of the Classic Period (AD 200 - 900), the Maya developed an absolute calendar called Long Count which counted the days from a fixed date in the past (the date when the current world was created according to Maya belief). Dates in the Long Count are given (for simplicity) in 5-tuples of the form. 9.2.3.4.5. Such a date one reads "9 baktuns 2 katuns 3 tuns 4 winals 5 kins since the zero date". A "kin" is just one day. A winal is a group of 20 days. A tun is a group of 18 winals (thus a tun has 20*18 = 360 days, 5 days short of a year). From here on all units come in multiples of 20. Thus a katun is equal to 20 tuns (almost 20 years) and a baktun means 20 katuns (almost 400 years). Thus 9.2.3.4.5 really means "9*144000+2*7200+3*360+4*20+5 days since the zero date". Note that for every Long Count date b.k.t.w.i we have 0 <= k < 20; 0 <= t < 20; 0 <= w < 18; 0 <= i < 20. Given the periodicity of the Calendar Round, a legal date such as 3 Lamat 6 Pax has multiple occurrences in the Long Count. Thus, one difficulty in reading inscriptions is in establishing a

date for the inscription when the date is given only in terms of a Calendar Round (very common). In this case one must compute "all" the possible Long Count dates associated with the particular Calendar Round and based in some other context information deduce (for example, the text mentions a king for which other dates are known) which one applies.

We limit our interest to the Long Count dates in the baktuns 8 and 9 (they cover all the Classic Period). We know that the Long Count date 8.0.0.0.0 fell on the Calendar Round 9 Ajaw 3 Sip.

Task

Write a program that:

- reads a Calendar Round date,
- computes all Long Count dates in the baktuns 8 and 9 for the given Calendar Round date if this date is legal,
- writes the result.

Input

The first line of the input contains exactly one positive integer d equal to the number of data sets, 1 <= d <= 30. The data sets follow.

Each data set consists of exactly one line that contains exactly one Calendar Round date (maybe illegal). Tzolkin day number, Tzolkin day name, Haab day number and Haab month name separated by single spaces.

Output

For every data set your program must output an ascending sequence of Long Count dates computed for a given Calendar Round date. The first line of the output for the given input set should contain exactly one integer n equal to the length of the sequence (0, if the input date is illegal).

Each of the next n lines should contain exactly one Long Count date specified by exactly 5 integers (meaning the numbers of baktuns, katuns, tuns, winals and kins respectively) separated by single dots.

Example

Sample input:

```
2
3 Lamat 6 Pax
1 Ajaw 9 Chen
```

Sample output:

```
15
8.0.17.17.8
8.3.10.12.8
8.6.3.7.8
8.8.16.2.8
8.11.8.15.8
8.14.1.10.8
8.16.14.5.8
8.19.7.0.8
9.1.19.13.8
9.4.12.8.8
```

```
9.7.5.3.8
9.9.17.16.8
9.12.10.11.8
9.15.3.6.8
9.17.16.1.8
0
```

Added by: Adrian Kosowski
Date: 2004-07-02
Time limit: 3s
Source limit: 50000B
Languages: All
Resource: ACM Central European Programming Contest, Warsaw 2001

SPOJ Problem Set (classical)

108. Decoding Morse Sequences

Problem code: MORSE

Before the digital age, the most common "binary" code for radio communication was the Morse code. In Morse code, symbols are encoded as sequences of short and long pulses (called dots and dashes respectively). The following table reproduces the Morse code for the alphabet, where dots and dashes are represented as ASCII characters "." and "-":

```
A .-   B -... C -.-. D -..
E .    F ..-. G --.  H ....
I ..   J .--- K -.-  L .-..
M --   N -.   O ---  P .--.
Q --.- R .-.  S ...  T -
U ..-  V ...- W .--  X -..-
Y -.-- Z --..
```

Notice that in the absence of pauses between letters there might be multiple interpretations of a Morse sequence. For example, the sequence -.-..-- could be decoded both as CAT or NXT (among others). A human Morse operator would use other context information (such as a language dictionary) to decide the appropriate decoding. But even provided with such dictionary one can obtain multiple phrases from a single Morse sequence.

Task

Write a program that:

- reads a Morse sequence and a list of words (a dictionary),
- computes the number of distinct phrases that can be obtained from the given Morse sequence using words from the dictionary,
- writes the result.

Notice that we are interested in full matches, i.e. the complete Morse sequence must be matched to words in the dictionary.

Input

The first line of the input contains exactly one positive integer d equal to the number of data sets, 1 <= d <= 20. The data sets follow.

The first line of each data set contains a Morse sequence - a nonempty sequence of at most 10000 characters "." and "-" with no spaces in between.

The second line contains exactly one integer n, 1 <= n <= 10000, equal to the number of words in a dictionary. Each of the following n lines contains one dictionary word - a nonempty sequence of at most 20 capital letters from "A" to "Z". No word occurs in the dictionary more than once.

Output

The output should consist of exactly d lines, one line for each data set. Line i should contain one integer equal to the number of distinct phrases into which the Morse sequence from the i-th data set can be parsed. You may assume that this number is at most $2*10^9$ for every single data set.

Example

Sample input:

```
1
.---.--.-.-.-.---...-.---.
6
AT
TACK
TICK
ATTACK
DAWN
DUSK
```

Sample output:

```
2
```

Added by: Adrian Kosowski
Date: 2004-07-02
Time limit: 7s
Source limit: 50000B
Languages: All
Resource: ACM Central European Programming Contest, Warsaw 2001

SPOJ Problem Set (classical)

109. Exchanges

Problem code: EXCHNG

Given n integer registers $r_1, r_2, \ldots, r_n$ we define a Compare-Exchange Instruction CE(a,b), where a, b are register indices (1 <= a < b <= n):

```
CE(a, b)::
  if content(r_a) >  content(r_b) then
     exchange the contents of registers r_a  and r_b;
```

A Compare-Exchange program (shortly CE-program) is any finite sequence of Compare-Exchange instructions. A CE-program is called a Minimum-Finding program if after its execution the register r_1 always contains the smallest value among all values in the registers. Such a program is called reliable if it remains a Minimum-Finding program after removing any single Compare-Exchange instruction. Given a CE-program P, what is the smallest number of instructions that should be added at the end of program P in order to get a reliable Minimum-Finding program?

For instance, consider the following CE-program for 3 registers: CE(1, 2), CE(2, 3), CE(1, 2). In order to make this program a reliable Minimum-Finding program it is sufficient to add only two instructions: CE(1, 3) and CE(1, 2).

Task

Write a program that:

- reads the description of a CE-program,
- computes the smallest number of CE-instructions that should be added to make this program a reliable Minimum-Finding program,
- writes the result.

Input

The first line of the input contains exactly one positive integer d equal to the number of data sets, 1 <= d <= 10. The data sets follow.

Each data set consists of exactly two consecutive lines. The first of those lines contains exactly two integers n and m separated by a single space, 2 <= n <= 10000, 0 <= m <= 25000. Integer n is the number of registers and integer m is the number of program instructions.

The second of those lines contains exactly 2m integers separated by single spaces - the program itself. Integers a_j, b_j on positions 2j-1 and 2j, 1 <= j < = m, 1 < = a_j < b_j <= n, are parameters of the j-th instruction in the program.

Output

The output should consist of exactly d lines, one line for each data set. Line i, 1 <= i <= d, should contain only one integer - the smallest number of instructions that should be added at the end of the i-th input program in order to make this program a reliable Minimum-Finding program.

Example

Sample input:

```
1
3 3
1 2 2 3 1 2
```

Sample output:

```
2
```

Added by: Adrian Kosowski
Date: 2004-07-02
Time limit: 5s
Source limit: 50000B
Languages: All
Resource: ACM Central European Programming Contest, Warsaw 2001

SPOJ Problem Set (classical)

110. Fill the Cisterns

Problem code: CISTFILL

During the next century certain regions on earth will experience severe water shortages. The old town of Uqbar has already started to prepare itself for the worst. Recently they created a network of pipes connecting the cisterns that distribute water in each neighbourhood, making it easier to fill them at once from a single source of water. But in case of water shortage the cisterns above a certain level will be empty since the water will flow to the cisterns below.

Example of cistern arrangement

You have been asked to write a program to compute the level to which cisterns will be filled with a certain volume of water, given the dimensions and position of each cistern. To simplify we will neglect the volume of water in the pipes.

Task

Write a program that:

- reads the description of cisterns and the volume of water,
- computes the level to which the cisterns will be filled with the given amount of water,
- writes the result.

Input

The first line of the input contains the number of data sets k, 1 <= k <= 30. The data sets follow.

The first line of each data set contains one integer n, the number of cisterns, 1 <= n <= 50000. Each of the following n lines consists of 4 nonnegative integers, separated by single spaces: b, h, w, d - the base level of the cistern, its height, width and depth in meters, respectively. The integers satisfy 0 <= b <= 10^6 and 1 <= h*w*d <= 40000. The last line of the data set contains an integer V - the volume of water in cubic meters to be injected into the network. Integer V satisfies 1 <= V <= $2*10^9$.

Output

The output should consist of exactly d lines, one line for each data set. Line i, 1 <= i <= d, should contain the level that the water will reach, in meters, rounded up to two fractional digits, or the word ‘OVERFLOW’, if the volume of water exceeds the total capacity of the cisterns.

Example

Sample input:

```
3
2
0 1 1 1
2 1 1 1
1
```

```
4
11 7 5 1
15 6 2 2
5 8 5 1
19 4 8 1
132
4
11 7 5 1
15 6 2 2
5 8 5 1
19 4 8 1
78
```

Sample output:

```
1.00
OVERFLOW
17.00
```

Warning: enormous Input/Output data, be careful with certain languages

---

Added by: Adrian Kosowski
Date: 2004-07-02
Time limit: 13s
Source limit: 50000B
Languages: All
Resource: ACM Central European Programming Contest, Warsaw 2001

SPOJ Problem Set (classical)

112. Horizontally Visible Segments

Problem code: SEGVIS

There is a number of disjoint vertical line segments in the plane. We say that two segments are horizontally visible if they can be connected by a horizontal line segment that does not have any common points with other vertical segments. Three different vertical segments are said to form a triangle of segments if each two of them are horizontally visible. How many triangles can be found in a given set of vertical segments?

Task

Write a program that:

- reads the description of a set of vertical segments,
- computes the number of triangles in this set,
- writes the result.

Input

The first line of the input contains exactly one positive integer d equal to the number of data sets, 1 <= d <= 20. The data sets follow.

The first line of each data set contains exactly one integer n, 1 <= n < = 8000, equal to the number of vertical line segments.

Each of the following n lines consists of exactly 3 nonnegative integers separated by single spaces: y'_i, y''_i, x_i (that is the y-coordinate of the beginning of a segment, y-coordinate of its end and its x-coordinate, respectively). The coordinates satisfy: $0 <= y'_i < y''_i <= 8000$, $0 <= x_i <= 8000$. The segments are disjoint.

Output

The output should consist of exactly d lines, one line for each data set. Line i should contain exactly one integer equal to the number of triangles in the i-th data set.

Example

Sample input:

```
1
5
0 4 4
0 3 1
3 4 2
0 2 2
```

```
0 2 3
```

Sample output:

```
1
```

Added by: Adrian Kosowski
Date: 2004-07-02
Time limit: 13s
Source limit: 50000B
Languages: All
Resource: ACM Central European Programming Contest, Warsaw 2001

SPOJ Problem Set (classical)

115. Family

Problem code: FAMILY

We want to find out how much related are the members of a family of monsters. Each monster has the same number of genes but the genes themselves may differ from monster to monster. It would be nice to know how many genes any two given monsters have in common. This is impossible, however, since the number of genes is very large. Still, we do know the family tree (well, not actually a tree, but you cannot really blame them, these are monsters, right?) and we do know how the genes are inherited so we can estimate the number of common genes quite well.

The inheritance rule is very simple: if a monster C is a child of monsters A and B then each gene of C is identical to the corresponding gene of either A or B, each with probability 50%. Every gene of every monster is inherited independently.

Let us define the degree of relationship of monsters X and Y as the expected number of common genes. For example consider a family consisting of two completely unrelated (i.e. having no common genes) monsters A and B and their two children C and D. How much are C and D related? Well, each of C's genes comes either from A or from B, both with probability 50%. The same is true for D. Thus, the probability of a given gene of C being the same as the corresponding gene of D is 50%. Therefore the degree of relationship of C and D (the expected number of common genes) is equal to 50% of all the genes. Note that the answer would be different if A and B were related. For if A and B had common genes, these would be necessarily inherited by both C and D.

Your task is to write a program that, given a family graph and a list of pairs of monsters, computes the degree of relationship for each of these pairs.

Task

Write a program that:

- reads the description of a family and a list of pairs of its members from the standard input,
- computes the degree of relationship (in percentages) for each pair on the list,
- writes the result to the standard output.

Input

The input begins with the integer t, the number of test cases. Then t test cases follow.

For each test case the first line of the input contains two integers n and k separated by a single space. Integer n (2 <= n <= 300) is the number of members in a family. Family members are numbered arbitrarily from 1 to n. Integer k (0 <= k <= n - 2) is the number of monsters that do have parents (all the other monsters were created by gods and are completely unrelated to each other).

Each of the next k lines contains three different integers a, b, c separated by single spaces. The triple a, b, c means that the monster a is a child of monsters b and c.

The next input line contains an integer m ($1 <= m < = n^2$) - the number of pairs of monsters on the list. Each of the next m lines contains two integers separated by a single space - these are the numbers of two monsters.

You may assume that no monster is its own ancestor. You should not make any additional assumptions on the input data. In particular, you should not assume that there exists any valid sex assignment.

Output

For each test case the output consists of m lines. The i-th line corresponds to the i-th pair on the list and should contain single number followed by the percentage sign. The number should be the exact degree of relationship (in percentages) of the monsters in the i-th pair. Unsignificant zeroes are not allowed in the output (please note however that there must be at least one digit before the period sign so for example the leading zero in number 0.1 is significant and you cannot print it as .1). Confront the example output for the details of the output format.

Example

Sample input:

```
1
7 4
4 1 2
5 2 3
6 4 5
7 5 6
4
1 2
2 6
7 5
3 3
```

Sample output:

```
0%
50%
81.25%
100%
```

Warning: large Input/Output data, be careful with certain languages

---

Added by: Adrian Kosowski
Date: 2004-07-07
Time limit: 15s
Source limit:50000B
Languages: All
Resource: ACM Central European Programming Contest, Warsaw 2002

SPOJ Problem Set (classical)

116. Intervals

Problem code: INTERVAL

You are given n closed integer intervals $[a_i, b_i]$ and n integers $c_1, ..., c_n$.

Task

Write a program that:

- reads the number of intervals, their endpoints and integers $c_1, ..., c_n$ from the standard input,
- computes the minimal size of a set Z of integers which has at least c_i common elements with interval $[a_i, b_i]$, for each i = 1, 2, ..., n,
- writes the answer to the standard output.

Input

The input begins with the integer t, the number of test cases. Then t test cases follow.

For each test case the first line of the input contains an integer n (1 <= n <= 50000) - the number of intervals. The following n lines describe the intervals. Line (i+1) of the input contains three integers a_i, b_i and c_i separated by single spaces and such that $0 <= a_i <= b_i <= 50000$ and $1 <= c_i <= b_i - a_i + 1$.

Output

For each test case the output contains exactly one integer equal to the minimal size of set Z sharing at least c_i elements with interval $[a_i, b_i]$, for each i= 1, 2, ..., n.

Example

Sample input:

```
1
5
3 7 3
8 10 3
6 8 1
1 3 1
10 11 1
```

Sample output:

```
6
```

Warning: enormous Input/Output data, be careful with certain languages

---

Added by: Adrian Kosowski
Date: 2004-07-07
Time limit: 7s
Source limit: 50000B
Languages: All
Resource: ACM Central European Programming Contest, Warsaw 2002

SPOJ Problem Set (classical)

118. Rhombs

Problem code: RHOMBS

An unbounded triangular grid is a plane covered by equilateral triangles:

rhombs

Two neighboring triangles in the grid form a rhomb. There are 3 types of such rhombs:

rhombs

A grid polygon is a simple polygon which sides consist entirely of sides of triangles in the grid. We say that a grid polygon is rhombastic if it can be partitioned into internally disjoint rhombs of types A, B and C.

As an example let's consider the following grid hexagon:

rhombs

This hexagon can be partitioned into 4 rhombs of type A, 4 rhombs of type B and 4 rhombs of type C:

rhombs

For a given rhombastic grid polygon P compute the numbers of rhombs of types A, B and C in some correct partition.

Task

Write a program that:

- reads a description of a rhombastic grid polygon from the standard input,
- computes the numbers of rhombs of types A, B and C in some correct partition of the polygon,
- writes the results to the standard output.

Input

The input begins with the integer t, the number of test cases. Then t test cases follow.

For each test case the first line of the input contains an integer n (3 <= n <= 50000) - the number of sides of a rhombastic grid polygon. Each of the next n lines contains a description of one side of the polygon. The sides are given one by one in the clockwise order. No two consecutive sides of the polygon lie on the same straight line. The description of a side consists of two integers d and k. Integer d says what is the direction of the side according to the following figure:

rhombs

Integer k is the length of the polygon side measured in the number of sides of grid triangles. Sum of all numbers k is not larger than 100000.

Output

For each test case the first and only line of the output contains three integers separated by single spaces denoting the number of rhombs of type A, B and C respectively, in some partition of the input polygon.

Example

Sample input:

```
1
6
1 2
2 2
3 2
4 2
5 2
6 2
```

Sample output:

```
4 4 4
```

Added by: Adrian Kosowski
Date: 2004-07-07
Time limit: 7s
Source limit: 50000B
Languages: All
Resource: ACM Central European Programming Contest, Warsaw 2002

SPOJ Problem Set (classical)

119. Servers

Problem code: SERVERS

The Kingdom of Byteland decided to develop a large computer network of servers offering various services.

The network is built of n servers connected by bidirectional wires. Two servers can be directly connected by at most one wire. Each server can be directly connected to at most 10 other servers and every two servers are connected with some path in the network. Each wire has a fixed positive data transmission time measured in milliseconds. The distance (in milliseconds) D(V, W) between two servers V and W is defined as the length of the shortest (transmission time-wise) path connecting V and W in the network. For convenience we let D(V, V) = 0 for all V.

Some servers offer more services than others. Therefore each server V is marked with a natural number r(V), called a rank. The bigger the rank the more powerful a server is.

At each server, data about nearby servers should be stored. However, not all servers are interesting. The data about distant servers with low ranks do not have to be stored. More specifically, a server W is interesting for a server V if for every server U such that D(V, U) <= D(V, W) we have r(U) <= r(W).

For example, all servers of the maximal rank are interesting to all servers. If a server V has the maximal rank, then exactly the servers of the maximal rank are interesting for V . Let B(V) denote the set of servers interesting for a server V.

We want to compute the total amount of data about servers that need to be stored in the network being the total sum of sizes of all sets B(V). The Kingdom of Byteland wanted the data to be quite small so it built the network in such a way that this sum does not exceed 30*n.

Task

Write a program that:

- reads the description of a server network from the standard input,
- computes the total amount of data about servers that need to be stored in the network,
- writes the result to the standard output.

Input

The input begins with the integer z, the number of test cases. Then z test cases follow.

For each test case, in the first line there are two natural numbers n, m, where n is the number of servers in the network (1 <= n <= 30000) and m is the number of wires (1 <= m <= 5n). The numbers are separated by single space.

In the next n lines the ranks of the servers are given. Line i contains one integer r_i (1 <= r_i <= 10) - the rank of i-th server.

In the following m lines the wires are described. Each wire is described by three numbers a, b, t (1 <= t <= 1000, 1 <= a, b <= n, a<> b), where a and b are numbers of the servers connected by the wire and t is the transmission time of the wire in milliseconds.

Output

For each test case the output consists of a single integer equal to the total amount of data about servers that need to be stored in the network.

Example

Sample input:

```
1
4 3
2
3
1
1
1 4 30
2 3 20
3 4 20
```

Sample output:

```
9
```

(because B(1) = {1, 2}, B(2) = {2}, B(3) = {2, 3}, B(4) = {1, 2, 3, 4})

Warning: large Input/Output data, be careful with certain languages

---

Added by: Adrian Kosowski
Date: 2004-07-07
Time limit: 12s
Source limit:50000B
Languages: All
Resource: ACM Central European Programming Contest, Warsaw 2002

SPOJ Problem Set (classical)

120. Solitaire

Problem code: SOLIT

Solitaire is a game played on an 8x8 chessboard. The rows and columns of the chessboard are numbered from 1 to 8, from the top to the bottom and from left to right respectively.

There are four identical pieces on the board. In one move it is allowed to:

- move a piece to an empty neighboring field (up, down, left or right),
- jump over one neighboring piece to an empty field (up, down, left or right).

possible moves in solitaire

There are 4 moves allowed for each piece in the configuration shown above. As an example let's consider a piece placed in the row 4, column 4. It can be moved one row up, two rows down, one column left or two columns right.

Task

Write a program that:

- reads two chessboard configurations from the standard input,
- verifies whether the second one is reachable from the first one in at most 8 moves,
- writes the result to the standard output.

Input

The input begins with the integer t, the number of test cases. Then t test cases follow.

For each test case, each of two input lines contains 8 integers a_1, a_2, ..., a_8 separated by single spaces and describes one configuration of pieces on the chessboard. Integers a_{2j-1} and a_{2j} (1 <= j <= 4) describe the position of one piece - the row number and the column number respectively.

Output

For each test case the output should contain one word for each test case - 'YES' if a configuration described in the second input line is reachable from the configuration described in the first input line in at most 8 moves, or one word 'NO' otherwise.

Example

Sample input:

```
1
4 4 4 5 5 4 6 5
```

```
2 4 3 3 3 6 4 6
```

Sample output:

```
YES
```

Added by: Adrian Kosowski
Date: 2004-07-07
Time limit: 7s
Source limit: 50000B
Languages: All
Resource: ACM Central European Programming Contest, Warsaw 2002

SPOJ Problem Set (classical)

121. Timetable

Problem code: TTABLE

You are the owner of a railway system between n cities, numbered by integers from 1 to n. Each train travels from the start station to the end station according to a very specific timetable (always on time), not stopping anywhere between. On each station a departure timetable is available. Unfortunately each timetable contains only direct connections. A passenger that wants to travel from city p to city q is not limited to direct connections however - he or she can change trains. Each change takes zero time, but a passenger cannot change from one train to the other if it departs before the first one arrives. People would like to have a timetable of all optimal connections. A connection departing from city p at A o'clock and arriving in city q at B o'clock is called optimal if there is no connection that begins in p not sooner than at A, ends in q not later than at B, and has strictly shorter travel time than the considered connection. We are only interested in connections that can be completed during same day.

Task

Write a program that:

- reads the number n and departure timetable for each of n cities from the standard input,
- creates a timetable of optimal connections from city 1 to city n,
- writes the answer to the standard output.

Input

The input begins with the integer t, the number of test cases. Then t test cases follow.

For each test case the first line of the input contains an integer n (2 <= n <= 100000). The following lines contain n timetables for cities 1, 2, ..., n respectively.

The first line of the timetable description contains only one integer m. Each of the following m lines corresponds to one position in the timetable and contains: departure time A, arrival time B (A < B) and destination city number t (1 <= t <= n) separated by single spaces. Departure time A and arrival time B are written in format hh : mm, where hh are two digits representing full hours (00 <= hh <= 23) and mm are two digits representing minutes (00 <= mm <= 59). Positions in the timetable are given in non-decreasing order according to the departure times. The number of all positions in all timetables does not exceed 1000000.

Output

For each test case the first line of the output contains an integer r - the number of positions in the timetable being the solution. Each of the following r lines contains a departure time A and an arrival time B separated by single space. The time format should be like in the input and positions in the timetable should be ordered increasingly according to the departure times. If there is more then one optimal connection with the same departure and arrival time, your program should output just one.

Example

Sample input:

```
1
3
3
09:00 15:00 3
10:00 12:00 2
11:00 20:00 3
2
11:30 13:00 3
12:30 14:00 3
0
```

Sample output:

```
2
10:00 14:00
11:00 20:00
```

Warning: enormous Input/Output data, be careful with certain languages

---

Added by: Adrian Kosowski
Date: 2004-07-07
Time limit: 9s
Source limit: 50000B
Languages: All
Resource: ACM Central European Programming Contest, Warsaw 2002

SPOJ Problem Set (classical)

122. Voracious Steve

Problem code: STEVE

Steve and Digit bought a box containing a number of donuts. In order to divide them between themselves they play a special game that they created. The players alternately take a certain, positive number of donuts from the box, but no more than some fixed integer. Each player's donuts are gathered on the player's side. The player that empties the box eats his donuts while the other one puts his donuts back into the box and the game continues with the "loser" player starting. The game goes on until all the donuts are eaten. The goal of the game is to eat the most donuts. How many donuts can Steve, who starts the game, count on, assuming the best strategy for both players?

Task

Write a program that:

- reads the parameters of the game from the standard input,
- computes the number of donuts Steve can count on,
- writes the result to the standard output.

Input

The input begins with the integer t, the number of test cases. Then t test cases follow.

For each test case the first and only line of the input contains exactly two integers n and m separated by a single space, 1 <= m <= n <= 100 - the parameters of the game, where n is the number of donuts in the box at the beginning of the game and m is the upper limit on the number of donuts to be taken by one player in one move.

Output

For each test case the output contains exactly one integer equal to the number of donuts Steve can count on.

Example

Sample input:

```
1
5 2
```

Sample output:

```
3
```

Added by: Adrian Kosowski
Date: 2004-07-07
Time limit: 7s
Source limit: 50000B
Languages: All
Resource: ACM Central European Programming Contest, Warsaw 2002

SPOJ Problem Set (classical)

123. Paying in Byteland

Problem code: PAYING

There are infinitely many coin denominations in the Byteland. They have values of 2^i for i=0,1,2,... . We will say that set of coins c1,c2,...,ck is perfect when it is possible to pay every amount of money between 0 and c1+...+ck using some of them (so {4,2,2,1} is perfect while {8,1} is not). The question is - is it always possible to change given sum n into a perfect set of coins? Of course it is possible ;). Your task will be more complicated: for a sum n you should find minimal number of coins in its perfect representation.

Input

First line of input contains one integer c<=50 - number of test cases. Then c lines follow, each of them consisting of exactly one integer n<=10^1000.

Output

For each test case output minimal number of coins.

Example

```
Input:
5
507
29
8574
233
149

Output:
14
7
21
11
10
```

Added by: Pawel Gawrychowski
Date: 2004-07-07
Time limit: 7s
Source limit:50000B
Languages: All

SPOJ Problem Set (classical)

130. Rent your airplane and make money

Problem code: RENT

"ABEAS Corp." is a very small company that owns a single airplane. The customers of ABEAS Corp are large airline companies which rent the airplane to accommodate occasional overcapacity.

Customers send renting orders that consist of a time interval and a price that the customer is ready to pay for renting the airplane during the given time period. Orders of all the customers are known in advance. Of course, not all orders can be accommodated and some orders have to be declined. Eugene LAWLER, the Chief Scientific Officer of ABEAS Corp would like to maximize the profit of the company.

You are requested to compute an optimal solution.

Small Example

Consider for instance the case where the company has 4 orders:

- Order 1 (start time 0, duration 5, price 10)
- Order 2 (start time 3, duration 7, price 8)
- Order 3 (start time 5, duration 9, price 7)
- Order 4 (start time 6, duration 9, price 8)

The optimal solution consists in declining Order 2 and 3 and the gain is 10+8 = 18.
Note that the solution made of Order 1 and 3 is feasible (the airplane is rented with no interruption from time 0 to time 14) but non-optimal.

Input

The first line of the input contains a number T <= 30 that indicates the number of test cases to follow. The first line of each test case contains the number of orders n (n <= 10000). In the following n lines the orders are given. Each order is described by 3 integer values: The start time of the order st (0 <= st < 1000000), the duration d of the order (0 < d < 1000000), and the price p (0 < p < 100000) the customer is ready to pay for this order.

Output

You are required to compute an optimal solution. For each test case your program has to write the total price paid by the airlines.

Example

Input:

```
1
4
0 5 10
3 7 14
```

```
5 9 7
6 9 8
```
Output:
```
18
```

Warning: large Input/Output data, be careful with certain languages

---

Added by: Adrian Kuegel
Date: 2004-07-13
Time limit: 3s
Source limit: 50000B
Languages: All
Resource: ACM Southwestern European Regional Contest, Paris 2003

SPOJ Problem Set (classical)

131. Square dance

Problem code: SQDANCE

You are hired by french NSA to break the RSA code used on the Pink Card. The easiest way to do that is to factor the public modulus and you have found the fastest algorithm to do that, except that you have to solve a subproblem that can be modeled in the following way.
Let $ cal P$ $ = {p_1,p_2,...,p_n}$ be a set of prime numbers. If $ S = {s_1,s_2,...,s_u}$ and $ T = {t_1,...,t_v}$ are formed with elements of $ cal P$, then S*T will denote the quantity

$displaystyle s_1*s_2*cdot cdot cdot *s_u*t_1*t_2*cdot cdot cdot *t_v.$

We call relation a set of two primes p,q, where p and q are distinct elements of $ cal P$. You dispose of a collection of R relations $ S_i = {p_i,q_i}$ and you are interested in finding sequences of these, $ S_{i_1}, S_{i_2}, ..., S_{i_k}$ such that

$displaystyle S_{i_1}*S_{i_2}*cdot cdot cdot *S_{i_k}$

is a perfect square.

The way you look for these squares is the following. The ultimate goal is to count squares that appear in the process. Relations arrive one at a time. You maintain a collection $ cal C$ of relations that do not contain any square subproduct. This is easy: at first, $ cal C$ is empty. Then a relation arrives and $ cal C$ begins to grow. Suppose a new relation $ {p,q}$ arrives. If no square appears when adding $ {p,q}$ to $ cal C$, then $ {p,q}$ is added to the collection. Otherwise, a square is about to appear, we increase the number of squares, but we do not store this relation, hence $ cal C$ keeps the desired property.
Let us consider an example. First arrives $ S_1 = {2,3}$ and we put it in $ cal C$; then arrives $ S_2 = {5,11},S_3 = {3,7}$ and they are stored in $ cal C$. Now enters the relation $ S_4 = {2,7}$. This relation could be used to form the square:

$displaystyle S_1*S_3*S_4 = (2*3)*(3*7)*(2*7) = (2*3*7)^2.$

So we count 1 and do not store $ S_4$ in $ cal C$. Now we consider $ S_5 = {5,11}$ that could make a square with $ S_2$, so we count 1 square more. Then $ S_6 = {2,13}$ is put into $ cal C$. Now $ S_7 = {7,13}$ could make the square $ S_1*S_3*S_6*S_7$. Eventually, we get 3 squares.

Input

The first line of the input contains a number T <= 30 that indicates the number of test cases to follow. Each test case begins with a line containing two integers P and R: $ Ple 10^5$ is the number of primes occurring in the test case; R ($ le 10^5$) is the number of sets of primes that arrive. The subsequent R lines each contain two integers i and j making a set $ {p_i,q_i}(1le i,jle P)$. Note that we actually do not deal with the primes, they are irrelevant to the solution.

Output

For each test case, output the number of squares that can be formed using the preceding rules.

Example

Input:

```
2
6 7
1 2
3 5
2 4
1 4
3 5
1 6
4 6
2 3
1 2
1 2
1 2
```

Output:

```
3
2
```

Warning: large Input/Output data, be careful with certain languages

---

Added by: Adrian Kuegel
Date: 2004-07-13
Time limit: 5s
Source limit: 50000B
Languages: All
Resource: ACM Southwestern European Regional Contest, Paris 2003

SPOJ Problem Set (classical)

132. Help R2-D2!

Problem code: HELPR2D2

In Episode III of Star Wars (whose alleged title is "How I became Vader"), R2-D2 (Artoo-Detoo) is again confronted to a tedious work. He is responsible for the loading of the republic transport starships in the fastest way. Imagine a huge space area where n starships are parked. Each starship has a capacity of K cubic femtoparsec. Containers C_i arrive one at a time with some volume v_i (expressed in cubic femtoparsec). R2-D2 wants to minimize the number of starships used for a given sequence of containers.
Smart as he is, R2-D2 knows for sure that the problem is a hard one, even with the force being around. Here is the heuristics he selected to solve his problem. Start with all starships ready to load, and numbered S_0,S_1,etc. When a container C_j arrives, select the starship of minimal index i that can contain C_j and put it in S_i. In some sense, this heuristic minimizes the move of the container arriving before its loading.
At the end of the n arrivals, R2-D2 counts the number s of starships used and he measures the total waste w of the sequence. For i=0..s-1, the waste in starship i is given by the unused volume.
Your task is to simulate the algorithm of R2-D2.

Input

The first line of the input contains a number T <= 10 that indicates the number of test cases to follow. Each test case begins with K on a line (K <= 1000), followed by the number of containers in the sequence, n, on the second line (1 <= n <= 1000000). There are two possible formats for the remaining lines. If it contains one integer, then this is the next v_i. If it begins with the character b (for block), it is followed by 2 integers r and v. This means that the r next containers arriving have volume v.

Output

Your program must output the number s of starships used, followed by a blank, followed by the total waste w.
You can assume, that at most 100000 starships are needed, and R2-D2 has to change the starships in which the next container is loaded at most 100000 times.

Example

Input:

```
2
100
3
50
25
70
100
4
50
b 2 40
```

```
20
```
Output:
```
2 55
2 50
```

Added by: Adrian Kuegel
Date: 2004-07-14
Time limit: 7s
Source limit: 50000B
Languages: All
Resource: ACM Southwestern European Regional Contest, Paris 2003

SPOJ Problem Set (classical)

134. Phony Primes

Problem code: PHONY

You are chief debugger for Poorly Guarded Privacy, Inc. One of the top selling product, ReallySecureAgent(c), seems to have a problem with its prime number generator. It produces from time to time bogus primes N.
After a while, you realize that the problem is due to the way primes are recognized.
Every phony prime N you discover can be characterized as follows. It is odd and has distinct prime factors, say $ N = p_1 * p_2 * ... * p_k$ with $ p_ine p_j$, where the number k of factors is at least 3. Moreover, for all i=1..k, $ p_i-1$ divides N-1. For instance, 561 = 3*11*17 is a phony prime.
Intrigued by this phenomenon, you decide to write a program that enumerates all such N's in a given interval $ [N_{min},N_{max}[$ with $ 1 le N_{min} < N_{max} < 2^31, N_{max}-N_{min} < 10^6$.
Please note, that the source code limit for this problem is 2000 Bytes to avoid precalculated tables.

Input

Each test case contains one line. On this line are written two integers $ N_{min}$ and $ N_{max}$ separated by a blank. The end of the input is signalled by a line containing two zeros. The number of test cases is approximately 2000.

Output

For each test case, output the list of phony primes in increasing order, one per line. If there are no phony primes in the interval, then simply output none on a line.

Example

Input:

```
10 2000
20000 21000
0 0
```

Output:

```
561
1105
1729
none
```

Added by: Adrian Kuegel
Date: 2004-07-15
Time limit: 13s
Source limit: 2000B
Languages: All
Resource: ACM Southwestern European Regional Contest, Paris 2003

SPOJ Problem Set (classical)

135. Men at work

Problem code: MAWORK

Every morning you have to drive to your workplace. Unfortunately, roads are under constant repair. Fortunately, administration is aware that this may cause trouble and they enforce a strict rule on roadblocks: roads must be blocked only half of the time. However, contractors are free to schedule their working hours, still they must follow regulations:

- Working periods (when the road is blocked) and rest periods (when the road is open) must alternate and be of fixed length.
- The beginning of the day (time zero) must coincide with the beginning of a period.

Write a program that, given a description of the road network and of contractors schedules outputs the minimal time needed to drive from home to work.

Input

The first line of the input contains a number T <= 10 that indicates the number of test cases to follow. The road network is represented on a N x N grid and the first line of each test case consists in the number N, 2 <= N <= 25.

Then follows N lines of N characters that represent the road network at time zero. Those lines are made of "." (standing for open road) and "*" (standing for roadblock) and they encode the rows of the grid in increasing order, while columns are also presented in increasing order. Conventionally, your home is at the position first row, first column, while your workplace is at the position last row, last column. Furthermore, you leave home at time t=0, that is, your starting position is first row, first column at time zero.

At a given time t, your car must be on some "open road" cell. It takes one time unit to drive to any of the four adjacent cells heading toward north, south, west or east, and you may also choose to stay on the same cell for one time unit. Of course, those five moves are valid if and only if the target cell exists and is free at time t+1.

Finally comes N lines of N characters that represent the contractors schedules. Those lines match the ones of the grid description and are made of N characters 0,1,...,9 that specify the duration of the working (and rest) period for a given cell. Observe that 0 is a bit special, since it means that the corresponding cell status does not change.

Output

The output consist in a single line for each test case, holding either the requested time, or NO, if driving from home to work is not possible.

Example

Input:

```
2
10
.*********
........**
*.******.*
*.******.*
*.******.*
*........*
*.******.*
*.******.*
*........*
********..
0000000000
0000000000
0000000000
0000000000
0000000000
0123456780
0000000000
0000000000
0123456780
0000000000
3
...
**.
**.
021
002
000
```

Output:

```
34
NO
```

Added by: Adrian Kuegel
Date: 2004-07-16
Time limit: 9s
Source limit: 50000B
Languages: All
Resource: ACM Southwestern European Regional Contest, Paris 2003

SPOJ Problem Set (classical)

136. Transformation

Problem code: TRANS

You are given two short sequences of numbers, X and Y. Try to determine the minimum number of steps of transformation required to convert sequence X into sequence Y, or determine that such a conversion is impossible.

In every step of transformation of a sequence, you are allowed to replace exactly one occerunce of one of its elements by a sequence of 2 or 3 numbers inserted in its place, according to a rule specified in the input file.

Input

The input begins with the integer t, the number of test cases. Then t test cases follow.

For each test case, the first line of input contains four integers - N, M, U, V (1<=N,M<=50). The next two lines of input contain sequences X and Y, consisting of N and M integers respectively. The next U lines contain three integers: *a b c* each, signifying that integer *a* can be converted to the sequence *b c* in one step of transformation. The next V-U lines contain four integers: *a b c d* each, signifying that integer *a* can be converted to the sequence *b c d* in one step of transformation. With the exception of N and M, all integers provided at input are positive and do not exceed 30.

The format of one set of input data is illustrated below.
[IMAGE]

Output

For each test case output -1 if it is impossible to convert sequence X into sequence Y, or the minimum number of steps required to achieve this conversion otherwise.

Example

Sample input:

```
1
3 10 2 3
2 3 1
2 1 1 2 2 1 2 1 2 1
3 1 2
3 3 3
3 1 3 2
```

Sample output:

```
6
```

Added by:	Adrian Kosowski
Date:	2004-07-18
Time limit:	7s
Source limit:	50000B
Languages:	All
Resource:	based on a problem from the VI Polish Collegiate Team Programming Contest (AMPPZ), 2001

SPOJ Problem Set (classical)

137. Partition

Problem code: PARTIT

A *partition* of positive integer m into n components is any sequence $a_1,...,a_n$ of positive integers such that $a_1+...+a_n=m$ and $a_1<=a_2<=...<=a_n$. Your task is to determine the partition, which occupies the k-th position in the lexicographic order of all partitions of m into n components.

The lexicographic order is defined as follows: sequence $a_1,...,a_n$ comes before $b_1,...,b_n$ iff there exists such an integer i,1<=i<=n, that $a_j=b_j$ for all j, 1<= j< i, and $a_i < b_i$.

Input

The input begins with the integer t, the number of test cases. Then t test cases follow.

For each test case the input consists of three lines, containing the positive integers m, n and k respectively (1<=n<= 10, 1<= m<=220, k is not larger than the number of partitions of m into n components).

Output

For each test case output the ordered elements of the sought partition, separated by spaces.

Example

Sample input:
```
1
9
4
3
```

Sample output:
```
1 1 3 4
```

Added by: Adrian Kosowski
Date: 2004-07-19
Time limit: 7s
Source limit:50000B
Languages: All
Resource: VI Polish Collegiate Team Programming Contest (AMPPZ), 2001

SPOJ Problem Set (classical)

138. Election Posters

Problem code: POSTERS

A parliamentary election was being held in Byteland. Its enterprising and orderly citizens decided to limit the entire election campaign to a single dedicated wall, so as not to ruin the panorama with countless posters and billboards. Every politician was allowed to hang exactly one poster on the wall. All posters extend from top to bottom, but are hung at different points of the wall, and may be of different width. The wall is divided horizontally into sections, and a poster completely occupies two or more adjacent sections.

With time, some of the posters were covered (partially or completely) by those of other politicians. Knowing the location of all the posters and the order in which they were hung, determine how many posters have at least one visible section in the end.

Input

The input begins with the integer t, the number of test cases. Then t test cases follow.

Each test case begins with a line containing integer n - the number of posters (1<=n<=40000). Then n lines follow, the i-th (1<=i<=n) containing exactly two integers l_i r_i, denoting the numbers of the leftmost and rightmost sections covered by the i-th poster ($1<=l_i< r_i<= 10^7$). The input order corresponds to the order of hanging posters.

Output

For each test case output a line containing one integer- the number of posters with visible sections.

Example

Sample input:

```
1
5
1 4
2 6
8 10
3 4
7 10
```

Sample output:

```
4
```

An illustration of the sample input is given below.
The wall with posters

Added by: Adrian Kosowski
Date: 2004-07-19
Time limit: 7s
Source limit: 50000B
Languages: All
Resource: VI Polish Collegiate Team Programming Contest (AMPPZ), 2001

SPOJ Problem Set (classical)

139. The Long and Narrow Maze

Problem code: MAZE

Consider a maze consisting of 3 rows of n square blocks each. The passageways in every block match one of three possible patterns, numbered 0 (empty), 1 (straight) and 2 (bent), as depicted below.

Illustration of possible patterns

Your task is to determine whether it is possible to create a passage in a given maze, with an entrance at the left end and an outlet at the right end of the maze, only by rotating some of the squares of the maze by a multiple of 90 degrees.

Input

The input begins with the integer t, the number of test cases. Then t test cases follow.

Each test case begins with a line containing a single integer n - the number of squares in one row of the maze (1<= n <= 200000). The next n lines contain three integers each, denoting the types of blocks in consecutive columns of the maze. A column description is of the form *a b c* (0<=*a,b,c*<=2), where *a* represents the type of the block in the first row, *b* - in the second row and *c* - in the third row.

Output

For each test case output the word `yes` if it is possible to rotate the squares so as to form a connection between the left and right edge, and the word `no` in the opposite case.

Example

Sample input:

```
1
6
1 0 1
1 2 2
2 2 1
2 2 1
2 2 1
1 2 2
```

Sample output:

```
yes
```

Indeed, the sample input corresponds to the following maze:
Input illustration
for which there exists a correct solution to the problem:
Illustration of the solution

Warning: large Input/Output data, be careful with certain languages

---

Added by: Adrian Kosowski
Date: 2004-07-19
Time limit: 10s
Source limit: 50000B
Languages: All
Resource: VI Polish Collegiate Team Programming Contest (AMPPZ), 2001

SPOJ Problem Set (classical)

140. The Loner

Problem code: LONER

The loner is a one-dimensional board game for a single player. The board is composed of squares arranged in a single line, some of which initially have pawns on them. The player makes a move by jumping with a pawn over a pawn on an adjacent field, to an empty square two fields to the right or left of its initial position. The pawn that was jumped over is removed directly after the move, as illustrated below.

The two acceptable types of moves

The game is considered won if exactly one pawn remains on the gaming board, and is lost if the player cannot make a move.

Given the initial state of the gaming board, your task is to determine whether it is possible for the player to win the game.

Input

The input begins with the integer t, the number of test cases. Then t test cases follow.

Each test cases begins with the positive integer n <= 32000, denoting the size of the gaming board. The second and last line of the test case description contains a sequence of n characters 0 or 1, without any white spaces. The i-th square of the board is occupied by a pawn at the start of the game iff the i-th character of this sequence is 1.

Output

For each test case output the word `yes` if it is possible for the player to win the game for the presented starting configuration, or the word `no` in the opposite case.

Example

Sample input:

```
2
7
0110011
6
111001
```

Sample output:

```
yes
no
```

Added by:	Adrian Kosowski
Date:	2004-07-21
Time limit:	7s
Source limit:	50000B
Languages:	All
Resource:	VI Polish Collegiate Team Programming Contest (AMPPZ), 2001

SPOJ Problem Set (classical)

142. Johnny and the Glue

Problem code: GLUE

Little Johnny decided he needed to stick an open metal box to the floor in the hall of his parents' house, so that all guests coming in would trip on it. He knew that as soon as his parents saw what he had done, they would try to remove it, and he wasn't going to stand for this. So, he chose the strongest glue in his possession and left lots of dabs of it on the floor (from our point of view, these can be regarded as points). Now, the only question that remained was how to stick the box onto the floor. Johnny is very particular about the way he does this: the box is always stuck face down, so that it only touches the floor on the four edges of the rectangle that forms its base. He would like each of these edges to make contact with at least two dabs of glue. Furthermore, he doesn't want any of the dabs to stay outside the box, since this would ruin the fun (there is no way you can trip someone up, if you've glued them to the floor, is there?).

Obviously, Johnny can sometimes reach his objective in more than one way (especially since he has prepared boxes of all possible dimensions for his act of mischief). Depending on how he does this, a different section of floor will be covered by the box. Determine in how many ways Johnny can choose the section of floor to be covered by the box when gluing.

Input

The input begins with the integer t, the number of test cases. Then t test cases follow.

The first line of each test case contains positive integer n<=10000 - the number of dabs of glue on the floor. The next n lines contain two integers, x y (-15000<=x,y<=15000), representing the x and y coordinates of the dabs (given in the order in which they were placed by Johnny ;).

Output

For each test case output the number of different sections of floor Johnny may choose to cover (possibly 0).

Example

Sample input:

```
1
8
1 0
1 4
0 3
5 4
5 0
6 1
6 3
0 1
```

Sample output:

```
2
```

Added by:	Adrian Kosowski
Date:	2004-07-22
Time limit:	7s
Source limit:	50000B
Languages:	All
Resource:	based on a problem from the VI Polish Collegiate Team Programming Contest (AMPPZ), 2001

SPOJ Problem Set (classical)

145. Aliens

Problem code: ALIENS

Aliens visited our planet with an obvious intention to find some new species for their space zoo. After entering Earth's orbit, they positioned themselves over the town of Belgrade, having detected some life-form activity on the ground. As they approached the surface, they saw a group of half-intelligent beings. Those creatures were actually competitors of the Balkan Olympiad in Informatics who were enjoying the excursion after intense contest. Aliens want to abduct all n (2<=n<=100000) competitors since they are very compassionate, and don't want their creatures to feel lonely in the space zoo.
Aliens use tractor beam to take their prey. Tractor beam works in the following way: it projects a circle-shaped beam from the spacecraft to the ground vertically beneath it, and all beings that are found in that circle or on its boundary are taken. Projecting the tractor beam needs a certain amount of energy to be spent. As the radius of the tractor beam (radius of the circle on the ground) increases, more and more energy is required. Although extremely intelligent, aliens are much more advanced in social sciences than in programming. That's why they are asking you to help them find the position of their spacecraft so that the energy required to take all of the n competitors is minimal.
Help our alien brothers! Write a program that will find the required minimal radius of tractor beam that contains all n competitors and the optimal spacecraft location - which is the same as the center of the circle on the ground.

Input

First line of input contains one integer c<=20 - number of test cases. Each test case begins with number n (2<=n<=100000). Then n lines follow and i-th of them contains two real numbers xi and yi (-10000.0<=xi,yi<=10000.0) representing coordinates of the i-th competitor.

Output

For each test case output radius of the tractor beam and coordinates of the spacecraft. Numbers should be rounded to two decimal places.

Example

```
Input:
1
6
8.0 9.0
4.0 7.5
1.0 2.0
5.1 8.7
9.0 2.0
4.5 1.0

Output:
5.00
5.00 5.00
```

Warning: large Input/Output data, be careful with certain languages

Added by: Pawel Gawrychowski
Date: 2004-07-21
Time limit: 5s
Source limit: 50000B
Languages: All
Resource: Balkan Olympiad in Informatics 2002

SPOJ Problem Set (classical)

146. Fast Multiplication Again

Problem code: MULTIPLY

After trying to solve Problem Number 31 (Fast Multiplication) with some script languages that support arbitrary large integers and timing out, you wonder what would be the best language to do fast multiplication of integers. And naturally it comes to your mind: Of course it is brainf**k, because there are only very cheap operations in that language.

Input

Two positive integers, ended with a line feed (ASCII 10) each.

Output

The product of the two integers, terminated by a line feed. You may assume that this number will be less than 10000.

Example

```
Input:
1
2

Output:
2
```

Added by: Robin Nittka
Date: 2004-07-21
Time limit: 2s
Source limit:5000B
Languages: BF

SPOJ Problem Set (srednie)

147. Tautology

Problem code: TAUT

Write a program that checks if the given logical expression is a tautology. The logical expression is a tautology if it is always true, regardless of logical value of its variables.

Input

On the first line there is the number of expressions to check (at most 35). The expression is in a prefix notation, that means that operator precedes its arguments. The following logical operators will be used:

```
C - and
D - or
I - implies
E - if, and only if
N - not
```

The variables will be lowercase letters (a-z). There will be no more than 16 different letters in the expression. The length of the expression will not exceed 111 characters.

Output

For each expression write one word: YES if it is a tautology, NO in other case.

Example

Sample input:

```
7
IIpqDpNp
NCNpp
Iaz
NNNNNNNp
IIqrIIpqIpr
Ipp
Ezz
```

Sample output:

```
YES
YES
NO
NO
YES
YES
YES
```

Added by: Piotr Łowiec
Date: 2004-07-25
Time limit: 7s
Source limit: 50000B
Languages: C C99 strict C++ JAVA NEM PERL PYTH RUBY ICON TEXT

SPOJ Problem Set (classical)

148. Land for Motorways

Problem code: MLAND

With every year, the plans for the construction of motorways in Poland are more and more advanced. For some time, it seemed as if the building was actually going to start, so the question of purchasing the land under the roads was of some importance. Only certain cities can be connected by a road directly, provided the farmer owning the land under it agrees to sell out. As a result of the constant swing of moods, the price demanded for the land by each farmer changes in a linear fashion, with possibly different coefficients for every road. It may either increase or decrease (and sometimes even be negative, if the owner anticipates future profit from the proximity of a motorway).

It has been decided that the purchase of land will be made at some moment in between two fixed dates. At that moment, the current prices of land will be frozen, and the least costly configuration of bidirectional roads connecting all cities (directly or indirectly) will be chosen. All the land under the selected roads will subsequently be bought at the frozen price. Since business in the proximity of a motorway does have its advantages, some land owners might actually want their land to be bought and they may offer money into the bargain, consequently making the price of purchase negative.

You act as an intermediary for the purchase and charge a steady commission, proportional to the total sum of purchase. Oddly enough, when signing the contract you missed the clause about the possibility of the price being negative and now you begin to wonder whether you won't end up being charged for your own hard work. Since it is one of your tasks to select the moment of purchase, do so in such a way as to maximise your profit (if this is impossible, at least cut your losses as much as possible).

Input

The input begins with the integer t, the number of test cases. Then t test cases follow.

For each test case the first line contains two integers n m, denoting the number of cities to be connected and the number of available potential roads,respectively(1<=n<= 120,1<=m<=820). The next line contains two integers t_1 t_2, which stand for the earliest possible and latest possible moments of purchase (-10000<=t_1<=t_2<=10000). Each of the following m lines contains four integers, the i-th being: u_i v_i a_i b_i, which means that the i-th road connects city u_i with city v_i, and the purchase of the land under it costs b_i+j*a_i units of currency at moment j (e.g. at moment 0 the land costs b_i units). Please note that these integers are chosen from the following ranges: 0<=u_i,v_i<=n-1, -32000<=a_i,b_i<=32000.

Output

For each test case output a line with two floating point numbers, accurate to three digits after the decimal point. The first represents the moment of transaction you ought to choose, the second - the total value of the transaction at that moment. If more than one moment fulfills the conditions of the problem, choose the earliest.

Example

Sample input:

```
2
5 6
0 5
1 0 -6 -4
2 0 3 -3
3 0 1 5
3 1 -2 -3
4 1 -3 -2
4 3 -2 -3
5 7
-20 20
1 0 1 2
2 1 -7 4
3 1 -9 0
3 2 4 9
4 1 0 -2
4 2 2 3
4 3 6 -5
```

Sample output:

```
0.000 -13.000
0.111 -1.000
```

Added by: Adrian Kosowski
Date: 2004-07-24
Time limit: 7s
Source limit: 50000B
Languages: All
Resource: VII Polish Collegiate Team Programming Contest (AMPPZ), 2002

SPOJ Problem Set (classical)

149. Fencing in the Sheep

Problem code: FSHEEP

A shepherd is having some trouble penning in his flock of sheep. After several hours of ineffectual efforts he gives up, with some of the sheep within their polygon-shaped pen and some outside. Exhausted, he moves to a place within the pen from which he can see the whole interior of the pen (without any fence getting in the way) and begins to count the sheep which are within it. Please assist him in his task.

Input

The input begins with the integer t, the number of test cases. Then t test cases follow.

The first line of each test case contains two integers n m, denoting the number of vertices of the polygon forming the fence, and the number of sheep in the whole herd (3<=n<=100000, 0<=m<=100000). The next n lines contain two integers each, the i-th being x_i y_i - the x and y coordinates of the i-th vertex of the fence (given in anti-clockwise order, -32000<=x_i,y_i<=32000). The next m lines contain two integers each, the j-th being x_j y_j - the x and y coordinates of the j-th sheep (arranged in decreasing order of seniority, -32000<=x_j,y_j<=32000). The shepherd's observation point is within the pen and has coordinates (0,0).

Output

For each test case output a line with a single integer - the number of sheep within the pen. The sheep which are sitting back on the fence and enjoying a cigarette should be included in the count.

Example

Sample input:

```
1
6 5
2 2
4 4
6 6
-3 1
-1 -1
5 1
2 1
3 2
6 6
3 3
-3 0
```

Sample output:

```
3
```

Illustration of the sample test data:
The sheep with their shepherd

Warning: large Input/Output data, be careful with certain languages

---

Added by:	Adrian Kosowski
Date:	2004-07-24
Time limit:	7s
Source limit:	50000B
Languages:	All
Resource:	based on a problem from the VII Polish Collegiate Team Programming Contest (AMPPZ), 2002

SPOJ Problem Set (classical)

150. Where to Drink the Plonk?

Problem code: PLONK

Consider a city bounded by a square, whose n horizontal and n vertical streets divide it into $(n+1)^2$ square blocks. However, in tribute to the ancient traditions of the first dwellers (who tended to overindulge in alcohol), all the inhabitants live at crossroads. A group of friends would like to meet for an evening of merriment at the place of residence of one of them. With a good deal of foresight, anticipating the difficulties they might have getting back to their respective homes, they would like to meet in the house of the friend which minimises the total walking distance for all of them. Assume that everybody walks along the streets, turning only at crossroads, and the distance betweenany pair ofadjacent crossroads is 1.

Input

The input begins with the integer t, the number of test cases. Then t test cases follow.

For each test case the first line of input contains the integer n - the number of friends who want to meet ($1<=n<=10000$). The next n lines contain two integers each, the i-th being x_i y_i, standing for the x and y coordinates of the crossroads at which the i-th friend lives ($0<=x_i,y_i<=100000$).

Output

For each test case output the total distance covered by all friends when walking to the meeting place.

Example

Sample input:

```
1
7
1 3
3 2
3 5
6 9
10 1
12 4
5 7
```

Sample output:

```
39
```

Warning: large Input/Output data, be careful with certain languages

---

Added by:	Adrian Kosowski
Date:	2004-07-28
Time limit:	6s
Source limit:	50000B
Languages:	All
Resource:	based on a problem from the VII Polish Collegiate Team Programming Contest (AMPPZ), 2002

SPOJ Problem Set (classical)

151. The Courier

Problem code: COURIER

Byteland is a scarcely populated country, and residents of different cities seldom communicate with each other. There is no regular postal service and throughout most of the year a one-man courier establishment suffices to transport all freight. However, on Christmas Day there is somewhat more work for the courier than usual, and since he can only transport one parcel at a time on his bicycle, he finds himself riding back and forth among the cities of Byteland.

The courier needs to schedule a route which would allow him to leave his home city, perform the individual orders in arbitrary order (i.e. travel to the city of the sender and transport the parcel to the city of the recipient, carrying no more than one parcel at a time), and finally return home. All roads are bi-directional, but not all cities are connected by roads directly; some pairs of cities may be connected by more than one road. Knowing the lengths of all the roads and the errands to be performed, determine the length of the shortest possible cycling route for the courier.

Input

The input begins with the integer t, the number of test cases. Then t test cases follow.

Each test case begins with a line containing three integers: n m b, denoting the number of cities in Byteland, the number of roads, and the number of the courier's home city, respectively (1<=n<=100,1<=b<=m<=10000). The next m lines contain three integers each, the i-th being u_i v_i d_i, which means that cities u_i and v_i are connected by a road of length d_i (1<=u_i,v_i<=100, 1<=d_i<=10000). The following line contains integer z - the number of transport requests the courier has received (1<=z<=5). After that, z lines with the description of the orders follow. Each consists of three integers, the j-th being u_j v_j b_j, which signifies that b_j parcels should be transported (individually) from city u_j to city v_j. The sum of all b_j does not exceed 12.

Output

For each test case output a line with a single integer - the length of the shortest possible bicycle route for the courier.

Example

Sample input:

```
1
5 7 2
1 2 7
1 3 5
1 5 2
2 4 10
2 5 1
3 4 3
3 5 4
3
```

```
1 4 2
5 3 1
5 1 1
```

Sample output:

```
43
```

Added by:	Adrian Kosowski
Date:	2004-07-28
Time limit:	7s
Source limit:	50000B
Languages:	All
Resource:	based on a problem from the VII Polish Collegiate Team Programming Contest (AMPPZ), 2002

SPOJ Problem Set (classical)

153. Balancing the Stone

Problem code: SCALES

You are given scales for weighing loads. On the left side lies a single stone of known weight $W<2^N$. You own a set of N different weights, weighing 1, 2, 4, ..., 2^{N-1} units of mass respectively. Determine how many possible ways there are of placing some weights on the sides of the scales, so as to balance them (put them in a state of equilibrium). Output this value modulo a small integer D.

Input

The input begins with the integer t, the number of test cases. Then t test cases follow.

For each test case, the first line contains three integers: N L D, where N denotes the number of weights at your disposal, L is the length of the binary representation of number W, and D is the modulus (1<= L<= N<= 1000000, 2<= D<=100). The second line contains the value of W, encoded in the binary system as a sequence of exactly L characters 0 or 1 without separating spaces.

Output

For each test case, output a single line containing one integer - the calculated number of possible weight placements, modulo D.

Example

Sample input:

```
2
6 4 6
1000
6 6 100
100110
```

Sample output:

```
3
5
```

Warning: large Input/Output data, be careful with certain languages

---

Added by:	Adrian Kosowski
Date:	2004-07-31
Time limit:	7s
Source limit:	50000B
Languages:	All
Resource:	based on a problem from the VII Polish Collegiate Team Programming Contest (AMPPZ), 2002

SPOJ Problem Set (classical)

154. Sweet and Sour Rock

Problem code: ROCK

A manufacturer of sweets has started production of a new type of sweet called *rock*. Rock comes in sticks composed of one-centimetre-long segments, some of which are sweet, and the rest are sour. Before sale, the rock is broken up into smaller pieces by splitting it at the connections of some segments.

Today's children are very particular about what they eat, and they will only buy a piece of rock if it contains more sweet segments than sour ones. Try to determine the total length of rock which can be sold after breaking up the rock in the best possible way.

Input

The input begins with the integer t, the number of test cases. Then t test cases follow.

For each test case, the first line of input contains one integer N - the length of the stick in centimetres (1<=N<=200). The next line is a sequence of N characters '0' or '1', describing the segments of the stick from the left end to the right end ('0' denotes a sour segment, '1' - a sweet one).

Output

For each test case output a line with a single integer: the total length of rock that can be sold after breaking up the rock in the best possible way.

Example

Sample input:

```
2
15
100110001010001
16
0010111101100000
```

Sample output:

```
9
13
```

Added by:	Adrian Kosowski
Date:	2004-08-03
Time limit:	7s
Source limit:	50000B
Languages:	All
Resource:	based on a problem from the VII Polish Collegiate Team Programming Contest (AMPPZ), 2002

SPOJ Problem Set (classical)

160. Choosing a Palindromic Sequence

Problem code: PALSEC

Given two sequences of words: $X=(x_1,...,x_n)$ and $Y=(y_1,...,y_n)$, determine how many binary sequences $P=(p_1,...,p_n)$ exist, such that the word concatenation $z_1 z_2 ... z_n$, where $z_i=x_i$ iff $p_i=1$ and $z_i=y_i$ iff $p_i=0$, is a palindrome (a word which is the same when read from left to right and from right to left).

Input

The input begins with the integer t, the number of test cases. Then t test cases follow.

For each test case the first line contains the positive integer n - the number of words in a sequence (1<=n<=30). The following n lines contain consecutive words of the sequence X, one word per line. The next n lines contain consecutive words of the sequence Y, one word per line. Words consist of lower case letters of the alphabet ('a' to 'z'), are non-empty, and not longer than 400 characters.

Output

For each test case output one line containing a single integer - the number of different possible sequences P.

Example

Sample input:

```
1
5
ab
a
a
ab
a
a
baaaa
a
a
ba
```

Sample output:

```
12
```

Added by: Adrian Kosowski
Date: 2004-08-07
Time limit: 7s
Source limit: 50000B
Languages: All
Resource: IV Polish Olympiad in Informatics (Wojciech Rytter)

SPOJ Problem Set (classical)

174. Paint templates

Problem code: PAINTTMP

The Painter's Studio is preparing mass production of paintings. Paintings are going to be made with aid of square matrices of various sizes. A matrix of size i consists of 2^i rows and 2^i columns. There are holes on intersections of some rows and columns. Matrix of size 0 has one hole. For $i > 0$, matrix of size i is built of four squares of size $2^{(i-1)} * 2^{(i-1)}$. Look at the following figure:

[IMAGE]

Both squares on the right side and the bottom-left square are matrices of size i-1. Top-left square has no holes. Pictures are constructed in the following way. First, we fix three non-negative integers n, x, y. Next, we take two matrices of size n, place one of them onto the other and shift the upper one x columns right and y rows up. We place such a pattern on a white canvas and cover the common part of matrices with the yellow paint. In this way we get yellow stains on the canvas in the places where the holes in both matrices agree.

Example

Consider two matrices of size 2.

[IMAGE]

The upper matrix was shifted 2 columns right and 2 rows up. There are three places where holes agree.

Task

Write a program that for each test case:

- reads the sizes of two matrices and the numbers of columns and rows that the upper matrix should be shifted by, from the standard input;
- computes the number of yellow stains on the canvas;
- writes the result to the standard output.

Input

The number of test cases t is in the first line of input, then t test cases follow separated by an empty line

There is one integer n, $0 <= n <= 100$ in the first line of each test case. This number is the size of matrices used for production of paintings. In the second line there is one integer x and in the third line one integer y, where $0 <= x,y <= 2^n$. The integer x is the number of columns and y is the number of rows that the upper matrix should be shifted by.

Output

For each test case your program should produce one line with exactly one integer - the number of stains on the canvas.

Example

Sample input:

```
1
2
2
2
```

Sample output:

```
3
```

Added by:	Michał Czuczman
Date:	2004-08-10
Time limit:	3s
Source limit:	50000B
Languages:	All
Resource:	5th Polish Olympiad in Informatics, stage 1 (Wojciech Rytter)

SPOJ Problem Set (classical)

175. Polygon

Problem code: POLY1

We say that two triangles intersect if their interiors have at least one common point. A polygon is called convex if every segment connecting any two of its points is contained in this polygon. A triangle whose vertices are also vertices of a convex polygon is called an elementary triangle of this polygon. A triangulation of a convex polygon is a set of elementary triangles of this polygon, such that no two triangles from the set intersect and a union of all triangles covers the polygon. We are given a polygon and its triangulation. What is the maximal number of triangles in this triangulation that can be intersected by an elementary triangle of this polygon?

Example

Consider the following triangulation:

[IMAGE]

The elementary triangle (1,3,5) intersects all the triangles in this triangulation.

Task

Write a program that for each test case:

- reads the number of vertices of a polygon and its triangulation;
- computes the maximal number of triangles intersected by an elementary triangle of the given polygon;
- writes the result to standard output.

Input

The number of test cases t is in the first line of input, then t test cases follow separated by an empty line

In the first line of a test case there is a number n, $3 <= n <= 1000$, which equals the number of vertices of the polygon. The vertices of the polygon are numbered from 0 to n-1 clockwise. The following n-2 lines describe the triangles in the triangulation. There are three integers separated by single spaces in the (i+1)-st line, where $1 <= i <= n$-2. These are the numbers of the vertices of the i-th triangle in the triangulation.

Output

For each test case your program should produce one line with exactly one integer - the maximal number of triangles in the triangulation, that can be intersected by a single elementary triangle of the input polygon.

Example

Sample input:

```
1
6
0 1 2
2 4 3
0 5 4
2 4 0
```

Sample output:

```
4
```

Added by: Michał Czuczman
Date: 2004-08-10
Time limit: 5s
Source limit: 50000B
Languages: All
Resource: 5th Polish Olympiad in Informatics, stage 1 (Krzysztof Diks)

SPOJ Problem Set (classical)

176. Sum of one-sequence

Problem code: SUM1SEQ

We say that a sequence of integers is a one-sequence if the difference between any two consecutive numbers in this sequence is 1 or -1 and its first element is 0. More precisely: $[a_1, a_2, ..., a_n]$ is a one-sequence if

- for any k, such that $1 <= k < n : |a_k - a_{k+1}| = 1$ and
- $a_1 = 0$

Task

Write a program that for each test case:

- reads two integers describing the length of the sequence and the sum of its elements;
- finds a one-sequence of the given length whose elements sum up to the given value or states that such a sequence does not exist;
- writes the result to the standard output.

Input

The number of test cases t is in the first line of input, then t test cases follow separated by an empty line.

In the first line of a test case there is a number n, such that $1 <= n <= 10\ 000$, which is the number of elements in the sequence. In the second line there is a number S, which is the sum of the elements of the sequence, such that $|S| <= 50\ 000\ 000$.

Output

For each test case there should be written n integers (each integer in a separate line) that are the elements of the sequence (k-th element in the k-th line) whose sum is S or the word "No" if such a sequence does not exist. If there is more than one solution your program should output any one.

Consequent test cases should by separated by an empty line.

Example

Sample input:

```
1
8
4
```

Sample output:

```
0
1
2
```

```
1
0
-1
0
1
```

Added by:	Michał Czuczman
Date:	2004-08-10
Time limit:	3s
Source limit:	50000B
Languages:	All
Resource:	5th Polish Olympiad in Informatics, stage 1 (Grzegorz Jakacki)

SPOJ Problem Set (classical)

177. AB-words

Problem code: ABWORDS

Every sequence of small letters a and b (also the empty sequence) is called an ab-word. If $X = [x_1, ..., x_n]$ is an ab-word and i, j are integers such that $1 <= i <= j <= n$ then $X[i..j]$ denotes the subword of X consisting of the letters $x_i, ..., x_j$. We say that an ab-word $X = [x_1 ..x_n]$ is nice if it has as many letters a as b and for all $i = 1, ..., n$ the subword $X[1..i]$ has at least as many letters a as b.

Now, we give the inductive definition of the similarity between nice ab-words.

- Every two empty ab-words (i.e. words with no letters) are similar
- Two non-empty nice ab-words $X = [x_1, ..., x_n]$ and $Y = [y_1, ..., y_m]$ are similar if they have the same length ($n = m$) and one of the following conditions if fulfilled:
  1. $x_1 = y_1, x_n = y_n$ and $X[2..n-1]$ and $Y[2..n-1]$ are similar ab-words and they are both nice;
  2. there exists i, $1 <= i <= n$, such that $X[1..i]$, $X[i+1..n]$ are nice ab-words and
     a) $Y[1..i]$, $Y[i+1..n]$ are nice ab-words and $X[1..i]$ is similar to $Y[1..i]$ and $X[i+1..n]$ is similar to $Y[i+1..n]$, or
     b) $Y[1..n-i]$, $Y[n-i+1..n]$ are nice ab-words and $X[1..i]$ is similar to $Y[n-i+1..n]$ and $X[i+1..n]$ is similar to $Y[1..n-i]$.

A **level of diversity** of a non-empty set S of nice ab-words is the maximal number of ab-words that can be chosen from S in such a way that for each pair w_1, w_2 of chosen words, w_1 is not similar to w_2.

Task

Write a program that for each test case:

- reads elements of S from standard input;
- computes the level of diversity of the set S;
- writes the result to standard output.

Input

The number of test cases t is in the first line of input, then t test cases follow separated by an empty line.

In the first line of a test case there is a number n of elements of the set S, $1 <= n <= 1000$; in the following n lines there are elements of the set S, i.e. nice ab-words (one word in each line); the first letter of every ab-word is the first symbol in line and there are no spaces between two consecutive letters in the word; the length of every ab-word is an integer from the range [1..200].

Output

For each test case your program should output one line with one integer - the level of diversity of *S*.

Example

Sample input:

```
1
3
aabaabbbab
abababaabb
abaaabbabb
```

Sample output:

```
2
```

Added by:	Michał Czuczman
Date:	2004-08-10
Time limit:	13s
Source limit:	50000B
Languages:	All
Resource:	5th Polish Olympiad in Informatics, stage 1 (Krzysztof Diks)

SPOJ Problem Set (classical)

178. Road net

Problem code: ROADNET

A diskette was enclosed to a road map. The diskette contains the table of the shortest ways (distances) between each pair of towns on the map. All the roads are two-way. The location of towns on the map has the following interesting property: *if the length of the shortest way from town A to town B equals the sum of the lengths of the shortest ways from A to C and C to B then town C lies on (certain) shortest way from A to B*. We say that towns A and B are neighbouring towns if there is no town C such that the length of the shortest way from A to B equals the sum of the lengths of the shortest ways from A to C and C to B. Find all the pairs of neighbouring towns.

Example

For the table of distances:

	A	**B**	**C**
A	0	1	2
B	1	0	3
C	2	3	0

the neighbouring towns are A, B and A, C.

Task

Write a program that for each test case:

- reads the table of distances from standard input;
- finds all the pairs of neighbouring towns;
- writes the result to standard output.

Input

The number of test cases t is in the first line of input, then t test cases follow separated by an empty line.

In the first line of each test case there is an integer n, $1 <= n <= 200$, which equals the number of towns on the map. Towns are numbered from 1 to n.

The table of distances is written in the following n lines. In the $(i+1)$-th line, $1 <= i <= n$, there are n non-negative integers not greater than 200, separated by single spaces. The j-th integer is the distance between towns i and j.

Output

For each test case your program should write all the pairs of the neighbouring towns (i.e. their numbers). There should be one pair in each line. Each pair can appear only once. The numbers in each pair should be given in increasing order. Pairs should be ordered so that if the pair (a, b) precedes the pair (c, d) then $a < c$ or ($a = c$ and $b < d$).

Consequent test cases should by separated by an empty line.

Example

Sample input:

```
1
3
0 1 2
1 0 3
2 3 0
```

Sample output:

```
1 2
1 3
```

Added by: Michał Czuczman
Date: 2004-08-10
Time limit: 5s
Source limit: 50000B
Languages: All
Resource: 5th Polish Olympiad in Informatics, stage 2 (Piotr Chrząstowski-Wachtel)

SPOJ Problem Set (classical)

179. Word equations

Problem code: WORDEQ

Every non-empty sequence of elements 0 and 1 is called a binary word. A word equation is an equation of the form $x_1 x_2 ... x_l = y_1 y_2 ... y_r$, where x_i and y_j are binary digits (0 or 1) or variables i.e. small letters of English alphabet. For every variable there is a fixed length of the binary words that can be substituted for this variable. This length is called a length of variable. In order to solve a word equation we have to assign binary words of appropriate length to all variables (the length of the word assigned to the variable x has to be equal to the length of this variable) in such a way that if we substitute words for variables then both sides of the equation (which are binary words after substitution) become equal.

For a given equation compute how many distinct solutions it has.

Example

Let `a`, `b`, `c`, `d`, `e` be variables and let 4, 2, 4, 4, 2 be their lengths (4 is the length of `a`, 2 is the length of `b` etc.). Consider the equation:

```
1bad1 = acbe
```

This equation has 16 distinct solutions.

Input

The number of equations t is in the first line of input, then t descriptions of equations follow separated by an empty line.

Each description consists of 6 lines. An equation is described in the following way: in the first line of the description there is an integer k, 0 <= k <= 26, which denotes the number of distinct variables in the equation. We assume that variables are the first k small letters of English alphabet. In the second line there is a sequence of k positive integers separated by single spaces. These numbers denote the lengths of variables `a`, `b`, ... from the equation (the first number is the length of `a`, the second - `b`, etc.). There is an integer l in the third line of the description, which is the length of the left size of equation, i.e. the length of the word built of digits 0 or 1 and variables (single letters). The left side of the equation is written in the next line as a sequence of digits and variables with no spaces between them. The next two lines contain the description of the right side of the equation. The first of these lines contains a positive integer r, which is the length of the right side of the equation. The second line contains the right side of the equation which is encoded in the same way as the left side. The number of digits plus sum of the lengths of variables (we count all appearances of variables) on each side of the equation is not greater than 10000.

Output

For each equation your program should output one line with the number of distinct solutions.

Example

Sample input:

```
1
5
4 2 4 4 2
5
1bad1
4
acbe
```

Sample output:

```
16
```

Added by:	Michał Czuczman
Date:	2004-08-10
Time limit:	5s
Source limit:	50000B
Languages:	All
Resource:	5th Polish Olympiad in Informatics, stage 2 (Wojciech Rytter)

SPOJ Problem Set (classical)

180. How to pack containers

Problem code: CONTPACK

Products of a factory are packed into cylindrical boxes. All boxes have the same bases. A height of a box is a non-negative integer being a power of 2, i.e. it is equal to 2^i for some $i = 0, 1, 2, \ldots$. The number i (exponent) is called a size of a box. All boxes contain the same goods but their value may be different. Goods produced earlier are cheaper. The management decided, that the oldest (cheapest) goods should be sold out first. From the warehouse goods are transported in containers. Containers are also cylindrical. A diameter of each container is a little bigger than a diameter of a box, so that boxes can be easily put into containers. A height of a container is a non-negative power of 2. This number is called a size of a container. For safe transport containers should be tightly packed with boxes, i.e. the sum of heights of boxes placed in a container have to be equal to the height of this container. A set of containers was delivered to the warehouse. Check if it is possible to pack all the containers tight with boxes that are currently stored in the warehouse. If so, find the minimal value of goods that can be tightly packed into these containers.

Consider a warehouse with 5 boxes. Their sizes and values of their contents are given below:

```
1 3
1 2
3 5
2 1
1 4
```

Two containers of size 1 and 2 can be tightly packed with two boxes of total value 3, 4 or 5, or three boxes with total value 9. The container of size 5 cannot be tightly packed with boxes from the warehouse.

Task

Write a program that for each test case:

- reads descriptions of boxes (size, value) from a warehouse and descriptions of containers (how many containers of a given size we have);
- checks if all containers can be tightly packed with boxes from the warehouse and if so, computes the minimal value of goods that can be tightly packed into these containers;
- writes the result.

Input

The number of test cases t is in the first line of input, then t test cases follow separated by an empty line.

In the first line of a test case there is an integer n, $1 <= n <= 10000$, which is the number of boxes in the warehouse. In each of the following n lines there are written two non-negative integers separated by a single space. These numbers describe a single box. First of them is the size of the box and the second - the value of goods contained in this box. The size is not greater than 1000 and the value is not

greater than 10000. The next line contains a positive integer q, which is the number of different sizes of containers delivered to the warehouse. In each of the following q lines there are two positive integers separated by a single space. The first integer is the size of a container and the second one is the number of containers of this size. The maximal number of containers is 5000, a size of a container is not greater than 1000.

Output

For each test case your program should output exactly one line containing:

- a single word "No" if it is not possible to pack the containers from the given set tight with the boxes from the warehouse, or
- a single integer equal to the minimal value of goods in boxes with which all the containers from the given set can be packed tight.

Example

Sample input:

```
1
5
1 3
1 2
3 5
2 1
1 4
2
1 1
2 1
```

Sample output:

```
3
```

Added by: Michał Czuczman
Date: 2004-08-10
Time limit: 5s
Source limit: 50000B
Languages: All
Resource: 5th Polish Olympiad in Informatics, stage 2 (Wojciech Rytter)

SPOJ Problem Set (classical)

181. Scuba diver

Problem code: SCUBADIV

A scuba diver uses a special equipment for diving. He has a cylinder with two containers: one with oxygen and the other with nitrogen. Depending on the time he wants to stay under water and the depth of diving the scuba diver needs various amount of oxygen and nitrogen. The scuba diver has at his disposal a certain number of cylinders. Each cylinder can be described by its weight and the volume of gas it contains. In order to complete his task the scuba diver needs specific amount of oxygen and nitrogen. What is the minimal total weight of cylinders he has to take to complete the task?

Example

The scuba diver has at his disposal 5 cylinders described below. Each description consists of: volume of oxygen, volume of nitrogen (both values are given in litres) and weight of the cylinder (given in decagrams):

```
3 36 120
10 25 129
5 50 250
1 45 130
4 20 119
```

If the scuba diver needs 5 litres of oxygen and 60 litres of nitrogen then he has to take two cylinders of total weight 249 (for example the first and the second ones or the fourth and the fifth ones).

Task

Write a program that for each test case:

- reads scuba diver's demand for oxygen and nitrogen, the number of accessible cylinders and their descriptions;
- computes the minimal total weight of cylinders the scuba diver needs to complete his task;
- outputs the result.

Note: the given set of cylinders always allows to complete the given task.

Input

The number of test cases c is in the first line of input, then c test cases follow separated by an empty line.

In the first line of a test case there are two integers t, a separated by a single space, $1 <= t <= 21$ and $1 <= a <= 79$. They denote volumes of oxygen and nitrogen respectively, needed to complete the task. The second line contains one integer n, $1 <= n <= 1000$, which is the number of accessible cylinders. The following n lines contain descriptions of cylinders; i-th line contains three integers t_i, a_i, w_i separated by single spaces, ($1 <= t_i <= 21$, $1 <= a_i <= 79$, $1 <= w_i <= 800$). These are respectively:

volume of oxygen and nitrogen in the *i*-th cylinder and the weight of this cylinder.

Output

For each test case your program should output one line with the minimal total weight of cylinders the scuba diver should take to complete the task.

Example

Sample input:

```
1
5 60
5
3 36 120
10 25 129
5 50 250
1 45 130
4 20 119
```

Sample output:

```
249
```

Added by: Michał Czuczman
Date: 2004-08-10
Time limit: 5s
Source limit: 50000B
Languages: All
Resource: 5th Polish Olympiad in Informatics, stage 2

SPOJ Problem Set (classical)

182. Window

Problem code: WINDOW1

We have a polygon chosen in the cartesian coordinate system. Sides of the polygon are parallel to the axes of coordinates. Every two consecutive sides are perpendicular and coordinates of every vertex are integers. We have also given a window that is a rectangle whose sides are parallel to the axes of coordinates. The interior of the polygon (but not its periphery) is coloured red. What is the number of separate red fragments of the polygon that can be seen through the window?

Example

Look at the figure below:

[IMAGE]

There are two separate fragments of the polygon that can be seen through the window.

Task

Write a program that for each test case:

- reads descriptions of a window and a polygon;
- computes the number of separate red fragments of the polygon that can be seen through the window;
- outputs the result.

Input

The number of test cases *t* is in the first line of input, then *t* test cases follow separated by an empty line.

In the first line of a test case there are four integers x_1, y_1, x_2, y_2 from the range [0..10000], separated by single spaces. The numbers x_1, y_1 are the coordinates of the top-left corner of the window. The numbers x_2, y_2 are the coordinates of the bottom-right corner of the window. The next line of the input file contains one integer *n*, 4 <= *n* <= 5000, which equals the number of vertices of the polygon. In the following *n* lines there are coordinates of polygon's vertices given in anticlockwise direction, i.e. the interior of the polygon is on the left side of its periphery when we move along the sides of the polygon according to the given order. Each line contains two integers *x*, *y* separated by a single space, 0 <= *x* <= 10000, 0 <= *y* <= 10000. The numbers in the *i*-th line, are coordinates of the *i*-th vertex of the polygon.

Output

For each test case you should output one line with the number of separate red fragments of the polygon that can be seen through the window.

Example

Sample input:

```
1
0 5 8 1
24
0 0
4 0
4 2
5 2
5 0
7 0
7 3
3 3
3 2
2 2
2 4
1 4
1 5
2 5
2 6
3 6
3 5
4 5
4 6
5 6
5 4
7 4
7 7
0 7
```

Sample output:

```
2
```

Added by: Michał Czuczman
Date: 2004-08-10
Time limit: 5s
Source limit: 50000B
Languages: All
Resource: 5th Polish Olympiad in Informatics, stage 2 (Wojciech Guzicki)

SPOJ Problem Set (classical)

183. Assembler circuits

Problem code: ASCIRC

Bytetel Company decided to improve computers they produce. They want to replace assembler programs with special systems called assembler circuits. Assembler programs consist solely of assignments. Each assignment is determined by four elements:

- two registers from which data are taken,
- elementary operation that should be performed on the data,
- register to which the result should be written.

We assume that there are at most 26 registers. They are represented by small letters of English alphabet. There are at most 4 elementary operations and they are represented by capital letters A, B, C, D.

An assembler circuit has:

- inputs assigned to registers; initial value of appropriate register is passed to the input;
- outputs, also assigned to registers; their final values are passed to these registers.

There are gates inside a circuit. Each gate has two inputs and one output. The gate performs an elementary operation on data delivered on its inputs and passes the result to its output. Inputs of gates and outputs of the whole circuit are connected to outputs of other gates or inputs of the circuit. Outputs of gates and inputs of the circuit can be connected to many inputs of other gates or outputs of the circuit. Connections among gates cannot form cycles.

An assembler circuit is equivalent to an assembler program if for any initial state of registers the final state of registers produced by the program and the circuit are the same.

Task

Write a program that for each test case:

- reads a description of an assembler program;
- computes the minimal number of gates in an assembler circuit equivalent to the given program;
- writes the result.

Input

The number of test cases t is in the first line of input, then t test cases follow separated by an empty line.

In the first line of each test case there is one integer n ($1 <= n <= 1000$), which is the number of instructions in the program.

In the following n lines there are descriptions of consecutive instructions in the program. Each description is a four-letter word beginning with an elementary operation symbol: A, B, C or D. The second and the third letter (which are small letters of English alphabet) are names of registers, in which data are placed. The fourth letter is a name of a register, in which the result should be placed.

Output

For each test case you should output one line with the minimal number of gates in an assembler circuit equivalent to the given program.

Example

Sample input:

```
1
8
Afbc
Bfbd
Cddd
Bcbc
Afcc
Afbf
Cfbb
Dfdb
```

Sample output:

```
6
```

A circuit equivalent to the given program is shown in the figure.

[IMAGE]

Added by: Michał Czuczman
Date: 2004-08-10
Time limit: 5s
Source limit: 50000B
Languages: All
Resource: 5th Polish Olympiad in Informatics, stage 3 (Marcin Kubica)

SPOJ Problem Set (classical)

184. Automatic Teller Machines

Problem code: ATMS

Every member of Byteland Credit Society is entitled to loan any amount of Bytelandish ducats unless it is 10^{30} or more, but he has to return the whole amount within seven days. There are 100 ATMs in the Client Service Room of the Society. They are numbered from 0 to 99. Every ATM can perform one action only: it can pay or receive a fixed amount. The *i*-th ATM pays 2^i ducats if *i* is even or it receives 2^i ducats if *i* is odd. If a client is going to loan a fixed sum of money it is necessary to check if he is able to get the money using every ATM at most once. If so, numbers of ATMs he has to use should be determined. It is also necessary to check if the client can return the money in a similar way, and if so, to determine numbers of ATMs he has to use.

Example

A client who is going to loan 7 ducats gets 16 ducats from the ATM # 4 and 1 ducat from the ATM # 0 and then he returns 8 ducats in the ATM # 3 and 2 ducats in the ATM # 1. In order to return the amount of 7 ducats he receives 1 ducat from the ATM # 0 and then he returns 8 ducats in ATM # 3.

Task

Write a program that:

- reads the number of clients *n*, for every client reads from the same file the amount of money he is going to loan;
- checks for every client if he is able to get the money using every ATM at most once and if so, determines the numbers of ATMs he has to use;
- outputs the results.

Input

In the first line of input there is one positive integer *n* <= 10000, which equals the number of clients.

In each of the following *n* lines there is one positive integer less than 10^{30} (at most 30 decimal digits). The number in the *i*-th line describes the amount of ducats which the client *i* is going to loan.

Output

For each client you should output two lines with a decreasing sequence of positive integers from the range [0..99] separated by single spaces, or one word "`No`":

In the first line of the *i*-th pair of lines there should be numbers of ATMs (in decreasing order) that the client *i* should use to get his loan or one word "`No`" if the loan cannot be received according to the rules;

In the second line of the *i*-th pair there should be numbers of ATMs (in decreasing order) which the client *i* should use to return his loan or the word "No".

Example

Sample input:

```
2
7
6338253001141147007483516026 98
```

Sample output:

```
4 3 1 0
3 0
No
99 3 1
```

Added by: Michał Czuczman
Date: 2004-08-10
Time limit: 5s
Source limit: 50000B
Languages: All
Resource: 5th Polish Olympiad in Informatics, stage 3 (Piotr Chrząstowski-Wachtel)

SPOJ Problem Set (classical)

185. Chase

Problem code: CHASE1

Chase is a two-person board game. A board consists of squares numbered from 1 to n. For each pair of different squares it is known if they are adjacent to one another or they are not. Each player has a piece at his disposal. At the beginning of a game pieces of players are placed on fixed, distinct squares. In one turn a player can leave his piece on the square it stands or move it to an adjacent square.

A game board has the following properties:

- it contains no triangles, i.e. there are no three distinct squares such that each pair of them is adjacent,
- each square can be reached by each player.

A game consists of many turns. In one turn each player makes a single move. Each turn is started by player A. We say that player A is caught by player B if both pieces stand on the same square. Decide, if for a given initial positions of pieces, player B can catch player A, independently of the moves of his opponent. If so, how many turns player B needs to catch player A if both play optimally (i.e. player A tries to run away as long as he can and player B tries to catch him as quickly as possible).

Example

[IMAGE]

Consider the board in the figure. Adjacent squares (denoted by circles) are connected by edges. If at the beginning of a game pieces of players A and B stand on the squares 9 and 4 respectively, then player B can catch player A in the third turn (if both players move optimally). If game starts with pieces on the squares 8 (player A) and 4 (player B) then player B cannot catch player A (if A plays correctly).

Task

Write a program that for each test case:

- reads the description of a board and numbers of squares on which pieces are placed initially.
- decides if player B can catch player A and if so, computes how many turns he needs (we assume that both players play optimally);
- outputs the result.

Input

The number of test cases t is in the first line of input, then t test cases follow separated by an empty line.

In the first line of a test case there are four integers n, m, a and b separated by single spaces, where $2 <= n <= 3000$, $n-1 <= m <= 15000$, $1 <= a, b <= n$. These are (respectively): the number of squares of the board, the number of adjacent (unordered) pairs, the number of the square on which the piece of player A is placed, the number of the square on which the piece of player B is placed. In each of the following lines there are two distinct positive integers separated by a single space, which denote numbers of adjacent squares.

Output

For each test case you should output one line containing:

- one word "No", if player B cannot catch player A, or
- one integer - the number of turns needed by B to catch A (if B can catch A).

Example

Sample input:

```
1
9 11 9 4
1 2
3 2
1 4
4 7
7 5
5 1
6 9
8 5
9 8
5 3
4 8
```

Sample output:

```
3
```

Added by: Michał Czuczman
Date: 2004-08-10
Time limit: 5s
Source limit: 50000B
Languages: All
Resource: 5th Polish Olympiad in Informatics, stage 3 (Adam Borowski)

SPOJ Problem Set (classical)

186. The lightest language

Problem code: LITELANG

Alphabet A_k consists of k initial letters of English alphabet. A positive integer called a weight is assigned to each letter of the alphabet. A weight of a word built from the letters of the alphabet A_k is the sum of weights of all letters in this word. A language over an alphabet A_k is any finite set of words built from the letters of this alphabet. A weight of a language is the sum of weights of all its words. We say that the language is prefixless if for each pair of different words w, v from this language w is not a prefix of v.

We want to find out what is the minimal possible weight of an n-element, prefixless language over an alphabet A_k.

Example

Assume that $k = 2$, the weight of the letter `a` is $W($`a`$) = 2$ and the weight of the letter `b` is $W($`b`$) = 5$. The weight of the word `ab` is $W($`ab`$) = 2 + 5 = 7$. $W($`aba`$) = 2 + 5 + 2 = 9$. The weight of the language $J =$ {`ab`, `aba`, `b`} is $W(J) = 21$. The language J is not prefixless, since the word `ab` is a prefix of `aba`. The lightest three-element, prefixless language over the alphabet A_2 (assuming that weights of the letters are as before) is {`b`, `aa`, `ab`}; its weight is 16.

Task

Write a program that for each test case:

- reads two integers n, k and the weights of k letters of an alphabet A_k;
- computes the minimal weight of a prefixless, n-element language over the alphabet A_k;
- outputs the result.

Input

The number of test cases t is in the first line of input, then t test cases follow separated by an empty line.

In the first line of a test case there are two positive integers n and k separated by a single space, ($2 <= n <= 10000$, $2 <= k <= 26$). These are the number of words in a language and the number of letters in an alphabet respectively. The second line contains k positive integers separated by single spaces. Each of them is not greater than 10000. The i-th number is the weight of the i-th letter.

Output

For each test case you should output one line with the weight of the lightest prefixless n-element language over the alphabet A_k.

Example

Sample input:

```
1
3 2
2 5
```

Sample output:

```
16
```

Added by: Michał Czuczman
Date: 2004-08-10
Time limit: 5s
Source limit: 50000B
Languages: All
Resource: 5th Polish Olympiad in Informatics, stage 3 (Wojciech Rytter)

SPOJ Problem Set (classical)

187. Flat broken lines

Problem code: FLBRKLIN

We have a cartesian coordinate system drawn on a sheet of paper. Let us consider broken lines that can be drawn with a single pencil stroke from the left to the right side of the sheet. We also require that for each segment of the line the angle between the straight line containing this segment and the OX axis belongs to [-45°, 45°] range. A broken line fulfilling above conditions is called a flat broken line. Suppose we are given *n* distinct points with integer coordinates. What is the minimal number of flat broken lines that should be drawn in order to cover all the points (a point is covered by a line if it belongs to this line)?

Example

[IMAGE]

For 6 points whose coordinates are (1,6), (10,8), (1,5), (2,20), (4,4), (6,2) the minimal number of flat broken lines covering them is 3.

Task

Write a program that for each test case:

- reads the number of points and their coordinates;
- computes the minimal number of flat broken lines that should be drawn to cover all the points;
- outputs the result.

Input

The number of test cases *t* is in the first line of input, then *t* test cases follow separated by an empty line.

In the first line of a test case there is one positive integer *n*, not greater than 30000, which denotes the number of points. In the following *n* lines there are coordinates of points. Each line contains two integers *x*, *y* separated by a single space, 0 <= *x* <= 30000, 0 <= *y* <= 30000. The numbers in the *i*-th line are the coordinates of the *i*-th point.

Output

For each test case you should output one line with the minimal number of flat broken lines that should be drawn to cover all the points.

Example

Sample input:

```
1
6
1 6
10 8
1 5
2 20
4 4
6 2
```

Sample output:

```
3
```

Added by:	Michał Czuczman
Date:	2004-08-10
Time limit:	5s
Source limit:	50000B
Languages:	All
Resource:	5th Polish Olympiad in Informatics, stage 3 (Grzegorz Jakacki, Krzysztof Sobusiak)

SPOJ Problem Set (classical)

188. Rectangles

Problem code: RECTNG1

There are *n* rectangles drawn on the plane. Each rectangle has sides parallel to the coordinate axes and integer coordinates of vertices.

We define a block as follows:

- each rectangle is a block,
- if two distinct blocks have a common segment then they form the new block otherwise we say that these blocks are separate.

Examples

The rectangles in Figure 1 form two separate blocks.

Figure 1

[IMAGE]

The rectangles in Figure 2 form a single block

Figure 2

[IMAGE]

Task

Write a program that for each test case:

- reads the number of rectangles and coordinates of their vertices;
- finds the number of separate blocks formed by the rectangles;
- writes the result to the standard output.

Input

The number of test cases *t* is in the first line of input, then *t* test cases follow separated by an empty line.

In the first line of a test case there is an integer *n*, 1 <= *n* <= 7000, which is the number of rectangles. In the following *n* lines there are coordinates of rectangles. Each rectangle is described by four numbers: coordinates *x*, *y* of the bottom-left vertex and coordinates *x*, *y* of the top-right vertex. All these coordinates are non-negative integers not greater than 10000.

Output

For each test case you should output one line with the number of separate blocks formed by the given rectangles.

Example

Sample input:

```
1
9
0 3 2 6
4 5 5 7
4 2 6 4
2 0 3 2
5 3 6 4
3 2 5 3
1 4 4 7
0 0 1 4
0 0 4 1
```

Sample output:

```
2
```

Added by:	Michał Czuczman
Date:	2004-08-10
Time limit:	5s
Source limit:	50000B
Languages:	All
Resource:	5th Polish Olympiad in Informatics, stage 3 (Wojciech Rytter)

SPOJ Problem Set (classical)

196. Musketeers

Problem code: MUSKET

In the time of Louis XIII and his powerful minister cardinal Richelieu in the Full Barrel Inn *n* musketeers had consumed their meal and were drinking wine. Wine had not run short and therefore the musketeers were eager to quarrel, a drunken brawl broke out, in which each musketeer insulted all the others.

A duel was inevitable. But who should fight who and in what order? They decided (for the first time since the brawl they had done something together) that they would stay in a circle and draw lots in order. A drawn musketeer fought against his neighbor to the right. A looser "quit the game" and to be more precise his corpse was taken away by servants. The next musketeer who stood beside the looser became the neighbor of a winner.

After years, when historians read memories of the winner they realized that a final result depended in a crucial extent on the order of duels. They noticed that a fence practice had indicated, who against who could win a duel. It appeared that (in mathematical language) the relation "**A** wins **B**" was not transitive! It could happen that the musketeer **A** fought better than **B**, **B** better than **C** and **C** better than **A**. Of course, among three of them the first duel influenced the final result. If **A** and **B** fight as the first, **C** wins eventually. But if **B** and **C** fight as the first, **A** wins finally. Historians fascinated by their discovery decided to verify which musketeers could survive. The fate of France and the whole civilized Europe indeed depended on that!

Task

N persons with consecutive numbers from *1* to *n* stay in a circle. They fight *n-1* duels. In the first round one of these persons (e.g. with the number *i*) fights against its neighbor to the right, i.e. against the person numbered *i+1* (or, if *i=n*, against the person numbered *1*). A looser quits the game, and the circle is tighten so that the next person in order becomes a winner's neighbor. We are given the table with possible duels results, in the form of a matrix. If **A***i,j* = 1 then the person with the number *i* always wins with the person *j*. If **A***i,j* = 0 the person *i* looses with *j*. We can say that the person *k* may win the game if there exists such a series of *n-1* drawings, that *k* wins the final duel.
Write a program which:

- reads matrix **A** from the standard input,
- computes numbers of persons, who may win the game,
- writes them into the standard output.

Input

The number of test cases t is in the first line of input, then t test cases follow separated by an empty line. In the first line of each test case integer *n* which satisfies the inequality *3<=n<=100* is written. In each of the following *n* lines appears one word consisting of *n* digits 0 or 1. A digit on *j*-th position in *i*-th line denote **A***i,j*. Of course **A***i,j* = 1 - **A***j,i*, for *i<>j*. We assume that **A***i,i* = 1, for each *i*.

Output

For each test case in the first line there should be written *m* - the number of persons, who may win the game. In the following *m* lines numbers of these persons should be written in ascending order, one number in each line.

Example

Sample input:

```
1
7
1111101
0101100
0111111
0001101
0000101
1101111
0100001
```

Sample output:

```
3
1
3
6
```

The order of duels: 1-2, 1-3, 5-6, 7-1, 4-6, 6-1 gives a final victory to the person numbered 6. You can also check that only two persons more (1 and 3) may win the game.

Added by: Piotr Łowiec
Date: 2004-09-13
Time limit: 7s
Source limit: 50000B
Languages: All
Resource: 6th Polish Olympiad in Informatics, stage 1

SPOJ Problem Set (classical)

199. Empty Cuboids

Problem code: EMPTY

We call a cuboid **regular** if:

- one of its vertices is a point with coordinates (0,0,0),
- edges beginning in this vertex lie on the positive semi-axes of the coordinate system,
- the edges are not longer than 10^6

There is given a set **A** of points of space, whose coordinates are integers from the interval $[1..10^6]$. We try to find a regular cuboid of maximal volume which does not contain any of the points from the set **A**. A point belongs to the cuboid if it belongs to the interior of the cuboid, i.e. it is a point of the cuboid, but not of its wall.

Task

Write a program which:

- reads from the standard input the coordinates of points from the set **A**,
- finds one of the regular cuboids of maximal volume which does not contain any points from the set **A**,
- writes the result to standard output.

Input

Input begins with a line containing integer t<=10, the number of test cases. t test cases follow.

In the first line of each test case one non-negative integer *n* is written (*n* <= *5000*). It is the number of elements in the set **A**. In the following *n* lines of the input there are triples of integers from the interval $[1..10^6]$, which are the X, Y and Z coordinates of points from **A**, repectively. Numbers in each line are separated by single spaces.

Output

For each test case there should be three integers separated by single spaces. These are the X, Y and Z coordinates (respectively) of the vertex of the regular cuboid of maximal volume. If there is more than one such a cuboid, choose whichever. We require that all coordinates be positive.

Example

Sample input:

```
1
4
3 3 300000
2 200000 5
90000 3 2000
```

```
2 2 1000
```

Sample output:

```
1000000 200000 1000
```

Added by: Piotr Łowiec
Date: 2004-09-13
Time limit: 7s
Source limit: 50000B
Languages: All
Resource: 6th Polish Olympiad in Informatics, stage 1

SPOJ Problem Set (classical)

200. Monodigital Representations

Problem code: MONODIG

Let *K* be a decimal digit different from *0*. We say that an arithmetic expression is a **K-representation of the integer X** if a value of this expression is X and if it contains only numbers composed of a digit K. (All the numbers are of course decimal). The following arithmetical operations are allowed in the expression: addition, subtraction, multiplication and division. Round brackets are allowed too. Division may appear only when a dividend is a multiple of a divisor.

Example

Each of the following expressions is the 5-representation of the number 12:

- 5+5+(5:5)+(5:5)
- (5+(5))+5:5+5:5
- 55:5+5:5
- (55+5):5

The **length** of the *K*-representation is the number of occurrences of digit *K* in the expression. In the example above the first two representations have the length 6, the third - 5, and the forth - 4.

Task

Write a program which:

- reads the digit *K* and the series of numbers from the standard input,
- verifies for each number from the series, whether it has a *K*-representation of length at most 8, and if it does, then the program finds the minimal length of this representation,
- writes results to the standard output.

Input

The number of test cases t is in the first line of input, then t test cases follow separated by an empty line. The first line of each test case contains digit *K*, *K* is en element of $\{1,...,9\}$. The second line contains number n, $1<=n<=10$. In the following *n* lines there is the series of natural numbers $a_1,...,a_n$, $1<=a_i<=32000$ (for $i=1,..,n$), one number in each line.

Output

The output for each test case composes of *n* lines. The *i*-th line should contain:

- exactly one number which is the minimal length of *K*-representation of a_i, assuming that such a representation of length not grater then 8 exists,
- one word NO, if the minimal length of the *K*-representation of the number a_i is grater than 8.

Example

Sample input:

```
1
5
2
12
31168
```

Sample output

```
4
NO
```

Added by: Piotr Łowiec
Date: 2004-09-13
Time limit: 7s
Source limit: 50000B
Languages: All
Resource: 6th Polish Olympiad in Informatics, stage 1

SPOJ Problem Set (classical)

201. The Game of Polygons

Problem code: POLYGAME

Two players take part in the game **polygons**. A convex polygon with *n* vertices divided by *n-3* diagonals into *n-2* triangles is necessary. These diagonals may intersect in vertices of the polygon only. One of the triangles is black and the remaining ones are white. Players proceed in alternate turns. Each player, when its turn comes, cuts away one triangle from the polygon. players are allowed to cut off triangles along the given diagonals. The winner is the player who cuts away the black triangle.
NOTE: We call a polygon **convex** if a segment joining any two points of the polygon is contained in the polygon.

Task

Write a program which:

- reads from the standard input the description of the polygon,
- verifies whether the player who starts the game has a winning strategy,
- writes the result to the standard output.

Input

The number of test cases t is in the first line of input, then t test cases follow separated by an empty line. The first line of each test case contains an integer *n*, *4 <= n <= 50000*. This is the number of vertices in the polygon. The vertices of the polygon are numbered, clockwise, from *0* to *n-1*.
The next *n-2* lines comprise descriptions of triangles in the polygon. In the*i+1*-th line, *1 <= i <= n-2*, there are three non-negative integers *a, b, c* separated by single spaces. Theses are numbers of vertices of the *i*-th triangle. The first triangle in a sequence is black.

Output

The output for each test case should have one line with the word:

- YES, if the player, who starts the game has a winning strategy,
- NO, if he does not have a winning strategy.

Example

Sample input:
```
1
6
0 1 2
2 4 3
4 2 0
0 5 4
```

Sample output:
```
YES
```

Warning: large Input/Output data, be careful with certain languages

---

Added by: Piotr Łowiec
Date: 2004-09-13
Time limit: 7s
Source limit: 50000B
Languages: All
Resource: 6th Polish Olympiad in Informatics, stage 1

SPOJ Problem Set (classical)

202. Rockets

Problem code: ROCKETS

There are two separate, *n*-element sets of points of a two dimensional map: **R** and **W**. None triple of points from the set **R**U**W** is collinear. Rockets earth-to-earth are located on points from the set **R**. Enemy objects, which should be destroyed, are located on points from the set **W**. The rockets may fly only in the straight line and their trajectories cannot intersect. We are about to find for each rocket a target to destroy.

Task

Write a program which:

- reads from the standard input coordinates of the points from the sets **R** and **W**,
- finds the set of *n* pairwise not-intersecting segments, so that one end of each segment belongs to the set **R**, while the other belongs to the set **W**,
- writes the result into the standard output.

Input

The number of test cases t is in the first line of input, then t test cases follow separated by an empty line. In the first line of each test case there is written one integer *n*, *1<=n<=10000*, equal to the number of elements of the sets **R** and **W**.

In each of the following *2n* lines of the input one pair of integer numbers from the interval [*-10000, 10000*] is written. Numbers in each pair are separated by a single space. They are coordinates of the point on a map (first coordinate *x*, then *y*). The first *n* lines comprise coordinates of the points from the set **R**, the last *n* lines comprise the points from the set **W**. In the (*i+1*)-th line there are coordinates of the point r_i, in the (*i+n+1*)-th line there are coordinates of the point w_i, *1<= i<= n*.

Output

The output for each test case should consist of *n* lines. In the *i*-th line there should be one integer *k(i)*, such that the segment $r_i\ w_{k(i)}$ belongs to the set of segments which your program found. (This means that the rocket from the point r_i destroys an object in the point $w_{k(i)}$).

Example

Sample input:

```
1
4
0 0
1 5
4 2
2 6
1 2
```

```
5 4
4 5
3 1
```

Sample output:

```
2
1
4
3
```

Warning: large Input/Output data, be careful with certain languages

---

Added by: Piotr Łowiec
Date: 2004-09-13
Time limit: 7s
Source limit: 50000B
Languages: All
Resource: 6th Polish Olympiad in Informatics, stage 2

SPOJ Problem Set (classical)

203. Potholers

Problem code: POTHOLE

A team of speleologists organizes a training in the Grate Cave ofByte Mountains. During the training each speleologist explores a route from Top Chamber to Bottom Chamber. The speleologists may move down only, i.e. the level of every consecutive chamber on a route should be lower then the previous one. Moreover, each speleologist has to start from Top Chamber through a different corridor and each of them must enter Bottom Chamber using different corridor. The remaining corridors may be traversed by more then one speleologist. How many speleologists can train simultaneously?

Task

Write a program which:

- reads the cave description from the standard input,
- computes the maximal number of speleologists that may train simultaneously,
- writes the result to the standard output.

Input

The number of test cases t is in the first line of input, then t test cases follow separated by an empty line. In the first line of each test case there is one integer *n* (*2<=n<=200*), equal to the number of chambers in the cave. The chambers are numbered with integers from *1* to *n* in descending level order - the chamber of grater number is at the higher level than the chamber of the lower one. (Top Chamber has number *1,* and Bottom Chamber has number *n*). In the following *n-1*lines (i.e. lines *2,3,...,n*) the descriptions of corridors are given. The (*i+1*)-th line contains numbers of chambers connected by corridors with the *i*-th chamber. (only chambers with numbers grater then *i* are mentioned). The first number in a line, *m*, *0<=m<=(n-i+1)*, is a number of corridors exiting the chamber being described. Then the following *m* integers are the numbers of the chambers the corridors are leading to.

Output

Your program should write one integer for each test case. This number should be equal to the maximal number of speleologists able to train simultaneously,

Example

Sample input:

```
1
12
4 3 4 2 5
1 8
2 9 7
2 6 11
1 8
2 9 10
2 10 11
```

```
1 12
2 10 12
1 12
1 12
```

Sample output:

```
3
```

The sample input corresponds to the following cave:
[IMAGE]

Added by: Piotr Łowiec
Date: 2004-09-13
Time limit: 7s
Source limit: 50000B
Languages: All
Resource: 6th Polish Olympiad in Informatics, stage 2

SPOJ Problem Set (classical)

204. Sleepwalker

Problem code: SLEEP

There is a building with flat square roof of size $3^k * 3^k$ and sides parallel to north-south and east-west directions. The roof is covered with square tiles of size *1* (with a side of length 1), but one of the tiles has been removed and there is a hole in the roof (big enough to fall in). The tiles form a rectangular mesh on the roof, so their positions may be specified with coordinates. The tile at the southwestern corner has coordinates *(1,1)*. The first coordinate increases while going eastwards, and the second while going northwards.

Sleepwalker wanders across the roof, in each step moving from the tile he is standing on to the adjacent one on the east(E), west(W), south(S), or north(N). The sleepwalker roof ramble starts from the southwestern corner tile. The description of the path is a word d_k built of the letters `N, S, E, W`denoting respectively a step to the north, south, east and west. For $k = 1$ the word describing the path of sleepwalker is

d_1 = `EENNWSWN`

For $k = 2$ the word describing the path of sleepwalker is

d_2 = `NNEESWSEENNEESWSEEEENNWSWNNEENNWSW -`
`NNEENNWSWNWWWSSENESSSSWWNENWWSSW -`
`WNENWNEENNWSWN.`

(See the picture that shows how the sleepwalker would go across a roof of dimension *3*3* or *9*9*.) Generally, if $k>=1$, the description of a sleepwalker's path on the roof of dimension $3^{k+1} * 3^{k+1}$ is a word:

d_{k+1} = a(d_k) E a(d_k) E d_k N d_k N d_k W c(d_k) S b(d_k) W b(d_k) N d_k

where functions **a**, **b** and **c** denote the following permutations of letters specifying directions:

```
a: E->N W->S N->E S->W
b: E->S W->N N->W S->E
c: E->W W->E N->S S->N
```

E.g. a(SEN)=WNE, b(SEN)=ESW, c(SEN)=NWS.

We start observing sleepwalker at the time he stands on the tile of coordinates (u_1, u_2). After how many steps will sleepwalker fall into the hole made after removing the tile of coordinates (v_1, v_2)?

Example

There are sleepwalker's paths on roofs of dimension *3*3* and *9*9* on the picture below. In the second case, the point at which the observation starts and the hole have been marked. The sleepwalker has exactly *20* steps to the hole (from the moment the observation starts).

[IMAGE] [IMAGE]

Task

Write a program which:

- reads from the standard input integer k denoting the size of the roof ($3^k * 3^k$), the position of the sleepwalker at the moment the observation starts and the position of the hole,
- computes the number of steps that the sleepwalker will make before he falls into the hole,
- writes the result to the standard output.

Input

The number of test cases t is in the first line of input, then t test cases follow separated by an empty line. In the first line of each test case one integer k, $1<=k<=60$, denoting the size of the roof ($3^k * 3^k$) is written. In each of the following two lines of the test case two natural numbers x, y separated with a space are written, $1<=x<=3^k$, $1<=y<=3^k$. The numbers in the second line are the coordinates of the tile the sleepwalker is standing on. The numbers in the third line are the coordinates of the hole. You may assume, that with these data the sleepwalker will eventually fall into the hole after some number of steps.

Output

The only line of output for each test case should contain the number of steps on the sleepwalker's path to the hole.

Example

Sample input:

```
1
2
3 2
7 2
```

Sample output

```
20
```

Added by: Piotr Łowiec
Date: 2004-09-13
Time limit: 7s
Source limit:50000B
Languages: All
Resource: 6th Polish Olympiad in Informatics, stage 2

SPOJ Problem Set (classical)

205. Icerink

Problem code: ICERINK

A skating competition was organized on the largest icerink in Byteland. The icerink is a square of size 10000 * 10000. A competitor begins skating at the START point chosen by referees and his task is to finish sliding at the FINISH point, also chosen by referees. The points of START and FINISH are different. One can slide in directions parallel to the sides of the icerink. There are some obstacles placed on the icerink. Each obstacle is a prism, which base is a polygon with sides parallel to the sides of the icerink. Each two adjacent sides of the base are always perpendicular. The obstacles do not have common points. Each slide finishes up at the point where a competitor, for the first time, meets the wall of an obstacle, which is perpendicular to the direction of the slide. In other words, one can stop only when he crashes on a wall or in the FINISH point. Falling out of the icerink causes disqualification. Competitor may slide along walls of an obstacle.

[IMAGE]

[IMAGE]

Decide, whether a competitor who slides according to the given rules may reach the finish point, assuming he begun sliding from the starting point. If so, what is the minimal number of slides he needs to do?

Task

Write a program which:

- reads the description of the icerink, obstacles, and the coordinates of the start and finish point from the standard input,
- verifies, whether a competitor who begins from the starting point and slides according the rules may reach the finish point, and if so, computes the minimal number of slides he needs to do,
- writes the result in the standard output.

Input

The number of test cases t is in the first line of input, then t test cases follow separated by an empty line. We define a system of coordinates to describe positions of objects on a rink. The rink is a square with vertices (0,0),(10000,0),(10000,10000),(0,10000). In the first line of each test case there are two integers z_1 and z_2 separated by a single space, $0<=z_1, z_2<=10000$. The pair (z_1, z_2) denotes coordinates of the START point. In the second line of the file there are two integers t_1 and t_2 separated by single space, $0<=t_1, t_2<=10000$. The pair (t_1, t_2) denotes coordinates of the FINISH point. The third line of the file contains one integer s, $1<=s<=2500$. This is the number of obstacles. The following lines comprise descriptions of s obstacles. Each description of an obstacle begins with the line containing one positive integer r equal to the number of walls (sides of the base) of the obstacle. In each of the following r lines there are two integers x and y separated by a single space.

These are the coordinates of the vertices of the obstacle's base, given in a clockwise order. (i.e. when going around the obstacle in this direction the inside is on the left-hand side). The total number of side walls of the obstacles does not exceed 10000.

Output

Your program should write for each test case:

- either one word 'NO' if it's impossible to get from the START point to the FINISH point
- or the minimal number of slides necessary to get to the FINISH point, if it is possible.

Example

Sample input:

```
1
40 10
5 40
3
6
0 15
0 60
20 60
20 55
5 55
5 15
12
30 55
30 60
60 60
60 0
0 0
0 5
55 5
55 35
50 35
50 40
55 40
55 55
6
30 25
15 25
15 30
35 30
35 15
30 15
```

Sample output:

```
4
```

The sample input corresponds to the following situation:

[IMAGE]

These are the possible sequences of slides of length 4:

[IMAGE] [IMAGE] [IMAGE]

Warning: large Input/Output data, be careful with certain languages

---

Added by: Piotr Łowiec
Date: 2004-09-13
Time limit: 7s
Source limit:50000B
Languages: All
Resource: 6th Polish Olympiad in Informatics, stage 2

SPOJ Problem Set (classical)

206. Bitmap

Problem code: BITMAP

There is given a rectangular bitmap of size *n*m*. Each pixel of the bitmap is either white or black, but at least one is white. The pixel in *i*-th line and *j*-th column is called the pixel (*i,j*). The distance between two pixels $\mathbf{p_1}=(i_1,j_1)$ and $\mathbf{p_2}=(i_2,j_2)$ is defined as:

$$d(\mathbf{p_1},\mathbf{p_2})=|i_1-i_2|+|j_1-j_2|.$$

Task

Write a program which:

- reads the description of the bitmap from the standard input,
- for each pixel, computes the distance to the nearest white pixel,
- writes the results to the standard output.

Input

The number of test cases t is in the first line of input, then t test cases follow separated by an empty line. In the first line of each test case there is a pair of integer numbers*n, m* separated by a single space, *1<=n <=182*, *1<=m<=182*. In each of the following *n* lines of the test case exactly one zero-one word of length *m,* the description of one line of the bitmap, is written. On the *j*-th position in the line (*i+1*), *1 <= i <= n*, *1 <= j <= m*, is '1' if, and only if the pixel (*i,j*) is white.

Output

In the *i*-th line for each test case, *1<=i<=n*, there should be written *m* integers f(*i,1*),...,f(*i,m*) separated by single spaces, where f(*i,j*) is the distance from the pixel (*i,j*) to the nearest white pixel.

Example

Sample input:

```
1

3 4
0001
0011
0110
```

Sample output:

```
3 2 1 0
2 1 0 0
1 0 0 1
```

Added by:	Piotr Łowiec
Date:	2004-09-13
Time limit:	4s
Source limit:	50000B
Languages:	All
Resource:	6th Polish Olympiad in Informatics, stage 2

SPOJ Problem Set (classical)

207. Three-coloring of binary trees

Problem code: THREECOL

A **tree** consists of a node and some (zero, one or two) subtrees connected to it. These subtrees are called children.

A **specification** of the tree is a sequence of digits. If the number of children in the tree is:

- zero, then the specification is a sequence with only one element ’0’;
- one, the specification begins with ’1’ followed by the specification of the child;
- two, the specification begins with ’2’ followed by the specification of the first child, and then by the specification of the second child.

Each of the vertices in the tree must be painted either red or green or blue.
However, we need to obey the following rules:

- the vertex and its child cannot have the same color,
- if a vertex has two children, then they must have different colors.

How many vertices may be painted green?

Task

Write a program which:

- reads the specification of the tree from the standard input,
- computes the maximal and the minimal number of vertices that may be painted green,
- writes the results in the standard output.

Input

The number of test cases t is in the first line of input, then t test cases follow separated by an empty line. Each test case consists of one word (no longer then 10000 characters), which is a specification of a tree.

Output

Your program should write for each test case exactly two integers separated by a single space, which respectively denote the maximal and the minimal number of vertices that may be painted green.

Example

Sample input:

```
1
```

```
1122002010
```

Sample output:

```
5 2
```

Added by: Piotr Łowiec
Date: 2004-09-13
Time limit: 7s
Source limit: 50000B
Languages: All
Resource: 6th Polish Olympiad in Informatics, stage 3

SPOJ Problem Set (classical)

208. Store-keeper

Problem code: STORE

The floor of a store is a rectangle divided into *n*m* square fields. Two fields are adjacent, if they have a common side. A parcel lays on one of the fields. Each of the remaining fields is either empty, or occupied by a case, which is too heavy to be moved by a store-keeper. The store-keeper has to shift the parcel from the starting field **P** to the final field **K**. He can move on the empty fields, going from the field on which he stands to a chosen adjacent field. When the store-keeper stays on a field adjacent to the one with the parcel he may push the parcel so that if moves to the next field (i.e. the field on the other side of the parcel), assuming condition that there are no cases on this field.

Task

Write a program, which:

- reads from the standard input a store scheme, a starting position of the store-keeper and a final position of the parcel,
- computes minimal number of the parcel shifts through borders of fields, which are necessary to put the parcel in the final position or decides that it is impossible to put the parcel there,
- writes the result into the standard output.

Input

The number of test cases t is in the first line of input, then t test cases follow separated by an empty line. In the first line of each test case two positive integers separated by a single space, *n,m<=100,* are written. These are dimensions of the store. In each of the following *n* lines there appears one *m*-letter word made of letters S, M, P, K, w. A letter on *i*-th position in *j*-th word denotes a type of the field with coordinates (*i,j*) and its meaning is following:

- S - case,
- M - starting position of the store-keeper,
- P - starting position of the parcel,
- K - final position of the parcel,
- w - empty field.

Each letter M, P and K appears in the test case exactly once.

Output

Your program should write to the standard output for each test case:

- exactly one word NO if the parcel cannot be put on the target field,
- exactly one integer, equal to the minimal number of the parcel shifts through borders of the fields, necessary to put a parcel on a final position, if it is possible to put the parcel there.

Example

Sample input:

```
1
10 12
SSSSSSSSSSS
Swwwwwww SSS
SwSSSSwwSSSS
SwSSSSwwSKSS
SwSSSSwwSwSS
SwwwwwPwwwww
SSSSSSSwSwSw
SSSSSSMwSwww
SSSSSSSSSSS
SSSSSSSSSSS
```

Sample output

```
7
```

Added by:	Piotr Łowiec
Date:	2004-09-13
Time limit:	7s
Source limit:	50000B
Languages:	All
Resource:	6th Polish Olympiad in Informatics, stage 3

SPOJ Problem Set (classical)

209. The Map

Problem code: MAP

After a new administrative division of Byteland cartographic office works on a new demographic map of the country. Because of technical reasons only a few colors can be used. The map should be colored so that regions with the same or similar population (number of inhabitants) have the same color. For a given color *k* let **A**(*k*) be the number, such that:

- at least half of regions with color *k* has population not greater than **A**(*k*)
- at least half of regions with color *k* has population not less than **A**(*k*)

A coloring error of a region is an absolute value of the difference between **A**(*k*) and the region's population. **A cumulative error** is a sum of coloring errors of all regions. We are looking for an optimal coloring of the map (the one with the minimal cumulative error).

Task

Write a program which:

- reads the population of regions in Byteland from the standard input,
- computes the minimal cumulative error,
- writes the result to the standard output.

Input

The number of test cases t is in the first line of input, then t test cases follow separated by an empty line. In the first line of each test case an integer *n* is written, which is the number of regions in Byteland, *10< n <3000*. In the second line the number *m* denoting the number of colors used to color the map is written, *2 <= m <= 10*. In each of the following *n* lines there is one non-negative integer - a population of one of the regions of Byteland. No population exceeds *2^30*.

Output

Your program should write for each test case one integer number equal to a minimal cumulative error, which can be achieved while the map is colored (optimally).

Example

Sample input:

```
1
11
3
21
14
6
18
10
```

```
2
15
12
3
2
2
```

Sample output:

```
15
```

Added by: Piotr Łowiec
Date: 2004-09-13
Time limit: 7s
Source limit: 50000B
Languages: All
Resource: 6th Polish Olympiad in Informatics, stage 3

SPOJ Problem Set (classical)

210. The Altars

Problem code: ALTARS

According to Chinese folk beliefs evil spirits can move only on a straight line. It is of a great importance when temples are built. The temples are constructed on rectangular planes with sides parallel to the north - south or east - west directions. No two of the rectangles have common points. An entrance is situated in the middle of one of four walls and its width is equal to the half of the length of the wall. An altar appears in the center of the temple, where diagonals of the rectangle intersect. If an evil spirit appears in this point, a temple will be profaned. It may happen only if there exists a ray which runs from an altar, through an entrance to infinity and neither intersects nor touches walls of any temple (on a plane parallel to the plane of a construction area), i.e. one can draw at a construction area a line which starts at the altar and runs to the infinity without touching any wall.

Task

Write a program which:

- reads descriptions of the temples from the standard input,
- verifies which temples could be profaned,
- writes their numbers to the standard output.

Input

The number of test cases t is in the first line of input, then t test cases follow separated by an empty line. In the first line of each test case one integer *n*, the number of temples *1 <= n <= 1000,* is written.

In each of the following *n* lines there is a description of one temple (in *i*-th line a description of the *i*-th temple). The description of a temple consists of four non-negative integers, not greater than 8000 and a letter E, W, S or N. Two first numbers are coordinates of a temple's northern-west corner and two following are coordinates of an opposite southern-east corner. In order to specify coordinates of a point first we give its geographical longitude, which increases from the west to the east, and then its latitude, which increases from the south to the north. The fifth element of the description indicates the wall with the entrance (E - Eastern, W - Western, S - Southern, N - Northern). The elements of the temple's description are separated by single spaces.

Output

In the following lines of the output for each test case your program should write in ascending order numbers of the temples, which may be profaned by an evil spirit. Each number is placed in a separate line. If there are no such numbers, you should write one word: NONE.

Example

Sample input

```
6
1 7 4 1 E
3 9 11 8 S
6 7 10 4 N
8 3 10 1 N
11 4 13 1 E
14 8 20 7 W
```

Sample output

```
1
2
5
6
```

The picture shows the temples described in the example. The dashed lines show possible routes of evil spirits.
[IMAGE]

Added by: Piotr Łowiec
Date: 2004-09-13
Time limit: 7s
Source limit:50000B
Languages: All
Resource: 6th Polish Olympiad in Informatics, stage 3

SPOJ Problem Set (classical)

211. Primitivus recurencis

Problem code: PRIMIT

A genetic code of the abstract primitivus (*Primitivus recurencis*) is a series of natural numbers $K=(a_1,...,a_n)$. **A feature** of primitivus we call each ordered pair of numbers (l,r), which appears successively in the genetic code, i.e. there exists such i that $l=a_i, r=a_{i+1}$. There are no (p,p) features in a primitivus' genetic code.

Task

Write a program which:

- reads the list of the features from the standard input,
- computes the length of the shortest genetic code having given features,
- writes the results to the standard output.

Input

The number of test cases t is in the first line of input, then t test cases follow separated by an empty line. In the first line of each test case one positive integer number n is written. It is the number of different features of the primitivus. In each of the following n lines there is a pair of natural numbers l and r separated by a single space, $1 <= l <= 1000, 1 <= r <= 1000$. A pair (l, r) is one of the primitivus' features. The features do not repeat in the input file

Output

Your program should write for each test case exactly one integer number equal to the length of the shortest genetic code of the primitivus, comprising the features from the input.

Example

Sample input:

```
1
12
2 3
3 9
9 6
8 5
5 7
7 6
4 5
5 1
1 4
4 2
2 8
```

```
8 6
```

Sample output:
```
15
```

All the features from the example are written in the following genetic code:
(8, 5, 1, 4, 2, 3, 9, 6, 4, 5, 7, 6, 2, 8, 6)

Warning: enormous Input/Output data, be careful with certain languages

---

Added by:	Piotr Łowiec
Date:	2004-09-13
Time limit:	7s
Source limit:	50000B
Languages:	All
Resource:	6th Polish Olympiad in Informatics, stage 3

SPOJ Problem Set (classical)

212. Water among Cubes

Problem code: WATER

On a rectangular mesh comprising *n*m* fields, *n*m* cuboids were put, one cuboid on each field. A base of each cuboid covers one field and its surface equals to one square inch. Cuboids on adjacent fields adhere one to another so close that there are no gaps between them. A heavy rain pelted on a construction so that in some areas puddles of water appeared.

Task

Write a program which:

- reads from the standard input a size of the chessboard and heights of cuboids put on the fields,
- computes maximal water volume, which may gather in puddles after the rain,
- writes results in the standard output.

Input

The number of test cases t is in the first line of input, then t test cases follow separated by an empty line. In the first line of each test case two positive integers 1 <= *n* <= 100, 1 <= *m* <= 100 are written. They are the size of the mesh. In each of the following *n* lines there are *m* integers from the interval [1..10000];*i*-th number in *j*-th line denotes a height of a cuboid given in inches put on the field in the *i*-th column and *j*-th raw of the chessboard.

Output

Your program should write for each tes case one integer equal to the maximal volume of water (given in cubic inches), which may gather in puddles on the construction.

Example

Sample input:

```
1
3 6
3 3 4 4 4 2
3 1 3 2 1 4
7 3 1 6 4 1
```

Sample output:

```
5
```

The picture below shows the mesh after the rain (seen from above). Puddles are drawn in gray.

[IMAGE] [IMAGE]

Added by:	Piotr Łowiec
Date:	2004-09-13
Time limit:	7s
Source limit:	50000B
Languages:	All
Resource:	6th Polish Olympiad in Informatics, stage 3

SPOJ Problem Set (classical)

215. Panic in the Plazas

Problem code: PANIC

Have you ever heard of the BBFO? The Bytelandian Bit-eating Fanatic Organisation regards itself as a collection of people with slightly unorthodox views on law and order in the world, and is regarded by others as the most wildly dangerous and unpredictable terrorist organisation which afflicts the small and otherwise peaceful country of Byteland.

Intelligence reports claim that the next act of violence to be performed by the BBFO is a widescale, distributed bomb attack in the Bytelandian capital. Therefore, all precautions have been undertaken to prevent any such action. The BBFO, seeing the futility of their original scheme, decided to change the plan of action. The new idea is endowed with devilish simplicity.

The capital of Byteland is a network of plazas, some of which (but not necessarily all) are connected by bidirectional streets of different length. Crowds of people are sitting at all the plazas, sipping coffee and generally relaxing. The terrorists plan to creep up to some of the plazas armed with inflatable paper bags. Then, exactly at midday, all the bags will be burst in such a way as to simulate the bang of a bomb. Panic will ensue at the plazas where the bags were burst, and will spread throughout some of the city. Panic breaks out at a plaza the moment a bag explodes in it, or immediately after a panicking crowd rushes into the plaza from at least one of the side streets. The people in the plaza then split up into crowds, which rush out by all possible streets except those by which people have just run in. After entering a street, a crowd runs along it at constant speed until it reaches the plaza at the other end, causing panic there, etc. If there is no possible way of escape from a plaza, everybody in it perishes. Similarly, if two crowds rushing in opposite directions collide in mid-street, all the people are lethally trampled.

A small illustration

Despite the panic, people in the city retain a little free will. They don't move at all until the panic reaches them, but when they have to escape, they can always choose the escape route from a plaza that suits them best. Assuming you were to sit in one of the plazas of Byteland at noon that fateful day... which plaza would you choose to sit in? All your normal preferences concerning the quality of coffee in the cafes are temporarily forgotten, and your only aim is to survive as long as possible.

Input

The first line of input contains a single integer t<=500, the number of test cases. t test cases follow. Each test case begins with a line containing three integers n m k (1<=n<=50000, 0<=m<=250000, 0<=k<=n) denoting the total number of plazas, the number of streets in the city, and the number of plazas in which bags are planted, respectively. Each of the following m lines contains 4 integers u v t_{uv} t_{vu} (1<=u,v<=n, 1<=t_{uv},t_{vu}<=1000) representing a single road in the city - leading from plaza u to plaza v and requiring t_{uv} time to cover when running at constant speed from u to v, and t_{vu} time when running the other way. The last line of a test case description contains a list of the k numbers of plazas at which bags explode at noon.

Output

For each test case, the output should contain a single line with a space separated increasing sequence of integers - the numbers of all the plazas which offer the maximum possible survival time to a person sitting there at noon.

Example

Input:

```
2

4 5 2
1 2 10 10
2 4 30 30
3 2 10 10
4 3 50 5
3 1 5  50
1 2

2 0 1
2
```

Output:

```
2 3 4
1
```

(In the first case the life expectancy is 22.5, in the second case it is more or less infinite.)

Warning: enormous Input/Output data, be careful with certain languages

---

Added by: Adrian Kosowski
Date: 2004-09-27
Time limit: 17s
Source limit: 50000B
Languages: All
Resource: DASM Programming League 2004 (problemset 2)

SPOJ Problem Set (classical)

217. Soldiers on Parade

Problem code: SOPARADE

Protocol is really weird in Byteland. For instance, it is required that, when presenting arms before an officer, soldiers should stand in a single row (at positions numbered from 1 to n). Soldiers may have one of 4 possible ranks, distinguished by the number of squiggles on the epaulets (between 1 and 4). Soldiers standing beside each other must have a difference in rank of at least two squiggles. Moreover, there are additional sets of rules (different for every province). Each rule states that soldiers standing at some given positions of the row must differ in rank by at least a squiggle.

Starting from the new year onwards, some provinces are changing their set of protocol rules. As the Senior Military Secretary of Protocol, it is your task to approve the new rules. To your surprise, some of the provinces have put forward protocol rules which are quite impossible to fulfill, even if the soldiers were to be specially selected for the purpose of presenting arms. Detect all such offending provinces and on no account approve their laws.

Input

The first line of input contains a single positive integer t<=10 - the number of provinces which are proposing new laws. t sets of rules follow, separated by empty lines.

Each set of rule begins with a line containing two non-negative integers n p (n<=100000, p<=100000) - the number of soldiers arranged and the number of rules proposed in the province, respectively. Each of the next p lines contains a single rule: an integer b_i (2<=b_i<=n), followed by b_i integers $a_1,a_2,...,a_{bi}$ (1<=a_k<=n). Such a rule means that soldiers standing at positions $a_1,a_2,...,a_{bi}$ must all be of different rank.

Output

For every set of rules presented at input, output a single line containing the word `rejected` if no unit of soldiers can be arranged in accordance with protocol, or the word `approved` in the opposite case.

Example

Input:

```
2

2 1
2 1 2

5 2
3 1 3 2
4 2 3 4 5
```

Output:

```
approved
rejected
```

Added by:	Adrian Kosowski
Date:	2004-10-08
Time limit:	9s
Source limit:	50000B
Languages:	All
Resource:	DASM Programming League 2004 (problemset 1)

SPOJ Problem Set (classical)

220. Relevant Phrases of Annihilation

Problem code: PHRASES

You are the King of Byteland. Your agents have just intercepted a batch of encrypted enemy messages concerning the date of the planned attack on your island. You immedietaly send for the Bytelandian Cryptographer, but he is currently busy eating popcorn and claims that he may only decrypt the most important part of the text (since the rest would be a waste of his time). You decide to select the fragment of the text which the enemy has strongly emphasised, evidently regarding it as the most important. So, you are looking for a fragment of text which appears in all the messages disjointly at least twice. Since you are not overfond of the cryptographer, try to make this fragment as long as possible.

Input

The first line of input contains a single positive integer t<=10, the number of test cases. t test cases follow. Each test case begins with integer n (n<=10), the number of messages. The next n lines contain the messages, consisting only of between 2 and 10000 characters 'a'-'z', possibly with some additional trailing white space which should be ignored.

Output

For each test case output the length of longest string which appears disjointly at least twice in all of the messages.

Example

Input:
```
1
4
abbabba
dabddkababa
bacaba
baba
```

Output:
```
2
```

(in the example above, the longest substring which fulfills the requirements is 'ba')

Added by: Adrian Kosowski
Date: 2004-10-11
Time limit: 9s
Source limit:50000B
Languages: All
Resource: DASM Programming League 2004 (problemset 1)

SPOJ Problem Set (classical)

224. Vonny and her dominos

Problem code: VONNY

Vonny loves playing with dominos. And so she owns a standard set of dominos. A standard set of dominos consists of 28 pieces called bones, tiles or stones. Each bone is a rectangular tile with a line dividing its face into two square ends. Each square is labeled with a number between 0 and 6. The 28 stones are labeled (0,0),(0,1),(0,2),(0,3),(0,4),(0,5),(0,6), (1,1),(1,2),...,(5,5),(5,6),(6,6). Tommy - the brother of Vonny - build a box for Vonny's dominos. This box is sized 7 x 8 squares. Every square is labeled with a number between 0 and 6. You can see a example box here.

```
0 3 0 2 2 0 2 3
1 5 6 5 5 1 2 2
3 4 1 4 5 4 4 4
6 6 1 0 5 2 3 0
4 0 3 2 4 1 6 0
1 4 1 5 6 6 3 0
1 2 6 5 5 6 3 3
```

Now Vonny wants to arrange her 28 stones in such way that her stones cover all squares of the box. A stone can only be placed on two adjacent squares if the numbers of the squares and of the domino stone are equal. Tommy asks Vonny in how many different ways she can arrange the dominos. Tommy assumes that Vonny need a lot of time to answer the question. And so he can take some of Vonny's candies while she solves the task. But Vonny is a smart and clever girl. She asks you to solve the task and keeps an eye on her candies.

Input

The first line of the input contains the number of testcases. Each case consists of 56 numbers (7 rows and 8 cols) between 0 and 6 which represents Tommy's box.

Output

For each testcase output a single line with the number which answers Tommy's question.

Example

Input:

```
2
0 3 0 2 2 0 2 3
1 5 6 5 5 1 2 2
3 4 1 4 5 4 4 4
6 6 1 0 5 2 3 0
4 0 3 2 4 1 6 0
1 4 1 5 6 6 3 0
1 2 6 5 5 6 3 3

5 3 1 0 0 1 6 3
0 2 0 4 1 2 5 2
1 5 3 5 6 4 6 4
```

```
0 5 0 2 0 4 6 2
4 5 3 6 0 6 1 1
2 3 5 3 4 4 5 3
2 1 1 6 6 2 4 3
```

Output:

```
18
1
```

Added by: Simon Gog
Date: 2004-10-18
Time limit: 20s
Source limit:50000B
Languages: All

SPOJ Problem Set (set2)

226. Jewelry and Fashion

Problem code: JEWELS

You work for a small jewelers' company, renowned for the exquisite necklaces and multi-colored amber strings it produces. For the last three centuries, the sales of strings alone have been enough to keep business going without a hitch. Now however, the influence of fashion is greater than ever, and you face the prospect of imminent bankruptcy unless you adapt to the needs and fancies of the rather unusual part of society who constitute your main clientele. These elderly ladies have recently decided that fashion has changed: strings are out, and earrings are in. There is nothing to be done about it -- you have to comply and switch to the production of earrings.

One problem remains: what to do with the impressive heap of amber strings piled up in your shop? One of your assistants has a bright idea: he recommends cutting the strings into two parts, removing some stones to make both parts have an identical color pattern (either immediately, or after rotation by 180 degrees), and selling what remains as pairs of earrings. After a moment's thought, you decide to go ahead with the plan. But your careful managerial eye tells you that minimising the number of wasted (removed) stones may not be as easy as it sounds...

Example of string2earring conversion ;)

Input

The first line of input contains a single integer t<=500, the number of test cases. The next t lines contain one test case each, in the form of a string of at most 8000 characters 'a'-'z' (terminated by a new line, optionally preceded by whitespace which should be ignored). The i-th character of the line corresponds to the design on the i-th stone in the amber string it represents. The total length of the input file is not more than 100kB.

Output

For each test case output two numbers: the largest possible total length of the pair of earrings which can be produced from the string, and a positive integer denoting the number of the stone after which the string ought to be cut so as to achieve this. If more than one cutting position is possible, output the leftmost (smallest) one.

Example

Input:
```
3
abcacdd
acbddabedff
abcbca
```

Output:
```
4 3
6 4
4 2
```

(the first case is illustrated in the figure, in the second case we produce a pair of earrings of the form 'abd', in the third - a pair of earrings which look like 'ab' after rotating the second one by 180 degrees).

Added by: Adrian Kosowski
Date: 2004-10-29
Time limit: 25s
Source limit: 50000B
Languages: All except: C99 strict
Resource: DASM Programming League 2004, problemset 2

SPOJ Problem Set (classical)

227. Ordering the Soldiers

Problem code: ORDERS

As you are probably well aware, in Byteland it is always the military officer's main worry to order his soldiers on parade correctly. In Bitland ordering soldiers is not really such a problem. If a platoon consists of n men, all of them have different rank (from 1 - lowest to n - highest) and on parade they should be lined up from left to right in increasing order of rank.

Sounds simple, doesn't it? Well, Msgt Johnny thought the same, until one day he was faced with a new command. He soon discovered that his elite commandos preferred to do the fighting, and leave the thinking to their superiors. So, when at the first rollcall the soldiers lined up in fairly random order it was not because of their lack of discipline, but simply because they couldn't work out how to form a line in correct order of ranks. Msgt Johnny was not at all amused, particularly as he soon found that none of the soldiers even remembered his own rank. Over the years of service every soldier had only learned which of the other soldiers were his superiors. But Msgt Johnny was not a man to give up easily when faced with a true military challenge. After a moment's thought a solution of brilliant simplicity struck him and he issued the following order: "`men, starting from the left, one by one, do: (step forward; go left until there is no superior to the left of you; get back in line).`". This did indeed get the men sorted in a few minutes. The problem was solved... for the time being.

The next day, the soldiers came in exactly the same order as the day before, and had to be rearranged using the same method. History repeated. After some weeks, Msgt Johnny managed to force each of his soldiers to remember how many men he passed when going left, and thus make the sorting process even faster.

If you know how many positions each man has to walk to the left, can you try to find out what order of ranks the soldiers initially line up in?

Input

The first line of input contains an integer t<=50, the number of test cases. It is followed by t test cases, each consisting of 2 lines. The first line contains a single integer n (1<=n<=200000). The second line contains n space separated integers w_i, denoting how far the i-th soldier in line must walk to the left when applying Msgt Johnny's algorithm.

Output

For each test case, output a single line consisting of n space separated integers - the ranks of the soldiers, given from left to right in their initial arrangement.

Example

Input:

```
2
3
0 1 0
5
0 1 2 0 1
```

Output:

```
2 1 3
3 2 1 5 4
```

Warning: large Input/Output data, be careful with certain languages

---

Added by: Adrian Kosowski
Date: 2004-10-30
Time limit: 13s
Source limit: 50000B
Languages: All
Resource: DASM Programming League 2004, problemset 2

SPOJ Problem Set (classical)

228. Shamans

Problem code: SHAMAN

In the far bare land there lives a mysterious tribe. They suffer from drought every year but they stick to their faith in god that they will never leave their home land. To counter the dry weather the shamans in the tribe must pray during the hard time and hope the blessed rain will aid their production of food.

There are 4 chief shamans in the tribe and each of them will choose a summit in the territory to proceed with his praying. The area in which the shamans' spells take effect will be the quadrangle they form, each of them being one of its vertices (which the god will see when he looks down from the high heavens). The land is quite full of pinch and punch and the tribe has selected quite a few peaks for the shamans to pray on. Of course the area of the quadrangle is expected to be as large as possible so before the shamans actually go out, they will have to choose the 4 peaks that best suit their purpose.

Input

One integer in the first line, stating the number of test cases, followed by a blank line. There will be not more than 80 tests.

For each test case, the first line is an integer n (4 <= n <= 2000) stating the number of peaks. Then n lines follow, each presenting the position of a peak, with two integers x, y (-20000 <= x, y <= 20000).

The test cases will be separated by a single blank line.

Output

A floating point number with exactly 1 digit precision: the maximum area the shamans can cover.

Example

Input:

```
2

4
0 0
1 0
1 1
0 1

4
0 0
0 1
1 1
1 0
```

Output:

```
1.0
1.0
```

Added by:	Neal Zane
Date:	2004-11-02
Time limit:	3s
Source limit:	50000B
Languages:	All
Resource:	Neal Zane

SPOJ Problem Set (classical)

229. Sorting is easy

Problem code: SORTING

Do you think sorting is easy?
try your luck
in brainfuck

For those who don't know that brainfuck is a programming language: Take a look at the converter to C. It will ignore every unknown command, therefore submitting a program in any other language won't necessarily lead to compile error, but certainly not to Accepted.

Input

The input consists of a line of up to 1000 uppercase letters, terminated with a '\n' character (ASCII value 10).

Output

The output should contain a line consisting of the same characters as the input line, but in non-descending order.

Example

Input:

```
BRAINFUCK
```

Output:

```
ABCFIKNRU
```

Added by: Adrian Kuegel
Date: 2004-11-04
Time limit: 1s
Source limit:500B
Languages: BF
Resource: own problem

SPOJ Problem Set (classical)

231. The Zebra Crossing

Problem code: ZEBRA

Have you ever wondered why people collide with each other at pedestrian crossings? The reasons are probably difficult to analyse from a scientific point of view, but we can hazard a guess. A part of the problem seems to be that the statistical pedestrian, when faced with a red light, will either cross at once (this category of pedestrians doesn't really interest us, since they tend to collide with cars rather than with each other), or will stop dead and stand still until the light changes. Only when the light turns green does he begin to act rationally and heads for his destination using the shortest possible path. Since this usually involves crossing the road slightly on the bias, he will inevitably bump into someone going across and heading another way.

One day, while you are approaching the traffic lights you usually cross at, you begin to wonder how many other people you could possibly collide with if you really wanted. All the people are standing at different points on the same side of the street as you are. From past observations you can predict the exact angle and speed at which individual pedestrians are going to cross. You can decide at which point along the side of the street you will wait for the green light (any real coordinate other than a place where some other pedestrian is already standing) and at what angle and at what speed you intend to cross. There is an upper bound on the speed you may cross at.

Assume that once the light turns green, all pedestrians start moving along straight lines, at constant speed, and that collisions, however painful they may be, have no effect on their further progress. Since you wouldn't like to arouse anyone's suspicions, you also have to cross in accordance with these rules. A collision only occurs if at a given moment of time you have exactly the same x and y coordinates as some other pedestrian.

Input

Input starts with a single integer t, the number of test cases (t<=100). t test cases follow.

Each test case begins with a line containing three integers n w v, denoting the number of people other than you who wish to cross the street, the width of the street expressed in meters, and the maximum speed you can walk at given in meters per second, respectively (1<=n<=10000, 1<=w<=100, 1<=v<=10000). Each of the next n lines contains three integers x_i t_i a_i, which describe the starting position of the i-th pedestrian measured in meters, the time (in seconds) he takes to cross the street, and the angle at which he is walking with respect to the line normal to the sides of the street, expressed in 1/60 parts of a degree (-10000<=x_i<=10000, 1<=t_i<=10000, -5000<=a_i<=5000).

Illustration of problem input

Output

For each test case output a single integer -- the maximum number of people you can collide with by the time you reach the opposite side of the street.

Example

Input:
```
1
5 20 2
-20 10 2700
20 10 -2700
-5 1 4000
-4 1 4000
5 1 -4000
```

Output:
```
2
```

(In the example, due to the imposed speed limit, it is only possible to collide with the first two pedestrians while crossing the street, at the last possible moment.)

Added by: Adrian Kosowski
Date: 2004-11-13
Time limit: 17s
Source limit: 50000B
Languages: All
Resource: DASM Programming League 2004, problemset 7

SPOJ Problem Set (classical)

234. Getting Rid of the Holidays (Act I)

Problem code: HOLIDAY1

King Johnny of Byteland has in his short period of sovereignty established quite a few national holidays (close on thirty, in fact) in honour of... more or less anything he could think of. Each of these holidays occurs every a fixed number of days (possibly different for every holiday), and is accompanied by feasts, cabaret shows, and general merrymaking. Sometimes more than one holiday occurs on a single day, and once in a while all holidays take place on the same day. If this happens, the celebrations are combined and even more festive. After one such party, king Johnny started behaving strangely and had to be temporarily isolated from society.

For the period of king Johnny's absence (about 48 hours) you have been appointed Regent of Byteland. As a true patriot, you know that holidays are not good for the people, and would like to remove some before king Johnny returns (he won't mind, he never remembers anything after a party anyway). The people however, very sadly, don't know what is good for them, and will revolt if you remove more than k holidays. Try to choose the holidays you remove in such a way as to guarantee that the number of days which elapse between two consecutive holiday parties is as long as possible.

Solve the problem in at most 4kB of source code.

Input

The first line of input contains a single integer t<=200 - the number of test cases. t test case descriptions follow.

For each test case, the first line contains two space separated integers n k (1<=k < n<=30), denoting the total number of holidays and the number of holidays to be removed. The next line contains n space separated integers, the i-th being t_i ($1<=t_i<=10^{18}$) - the number of days every which the i-th holiday occurs.

Output

For each test case, output one line containing an increasing sequence of exactly k integers - the numbers of the holidays to be removed (holidays are numbered in the input order from 1 to n).

Example

Input:
```
1
5 2
6 13 10 15 7
```

Output:
```
2 5
```

(The shortest period between two successive holiday parties is 2 days.)

Added by: Adrian Kosowski
Date: 2004-11-25
Time limit: 17s
Source limit: 4096B
Languages: All
Resource: DASM Programming League 2004, problemset 4

SPOJ Problem Set (classical)

235. Very Fast Multiplication

Problem code: VFMUL

Multiply the given numbers.

Input

n [the number of multiplications <= 101]

l1 l2 [numbers to multiply (at most 300000 decimal digits each)]

Text grouped in [ ] does not appear in the input file.

Output

The results of multiplications.

Example

```
Input:
5
4 2
123 43
324 342
0 12
9999 12345

Output:
8
5289
110808
0
123437655
```

Warning: large Input/Output data, be careful with certain languages

---

Added by: Darek Dereniowski
Date: 2004-11-27
Time limit: 3s
Source limit:50000B
Languages: All
Resource: PAL

SPOJ Problem Set (classical)

236. Converting number formats

Problem code: ROMAN

Given the number n of test cases, convert n positive integers less than 2^32 (given one per line) from one representation to another. For convenience, n is given in the same format as the other numbers.

Input

Input is given by spelling the number in english digits (all upper case letters). Thus the range of (32-bit) input values permissible extends from ZERO (or OH) through FOUR TWO NINE FOUR NINE SIX SEVEN TWO NINE FIVE.

Output

Output 2 lines for each test case. Output is in the form of "extended" Roman numerals (also called "butchered" Roman numerals), with an overline (see sample for details) indicating the value below is "times 1000", and lower-case letters indicating "times 1000000". Thus, the range of (32-bit) output values possible is from through ivccxcivCMLXVIICCXCV, where there is a line above iv and CMLXVII. Note: For values whose residues modulo 1000000 are less than 4000, M is used to represent 1000; for values whose residues are 4000 or greater, I is used. Thus 3999 would read out as MMMCMXCIX while 4000 would readout as IV with an overline. Similar rules apply to the use of M and i for 1000000, and to that of m and i for 1000000000.

WARNING: This problem has a somewhat strict source limit

Example

Input:

```
THREE
FOUR OH
ONE NINE NINE NINE NINE NINE NINE NINE NINE NINE
ONE TWO THREE ZERO FOUR FIVE
```

Output:

```
XL

      ______
mcmxcixCMXCIXCMXCIX

___
CXXMMMXLV
```

Added by: Robin Nittka
Date: 2004-11-30
Time limit: 9s
Source limit: 2048B
Languages: All

SPOJ Problem Set (classical)

237. Sums in a Triangle

Problem code: SUMITR

Let us consider a triangle of numbers in which a number appears in the first line, two numbers appear in the second line etc. Develop a program which will compute the largest of the sums of numbers that appear on the paths starting from the top towards the base, so that:

- on each path the next number is located on the row below, more precisely either directly below or below and one place to the right;
- the number of rows is strictly positive, but less than 100;
- all numbers are positive integers between O and 99.

Take care about your fingers, do not use more than **256** bytes of code.

Input

In the first line integer n - the number of test cases (equal to about 1000). Then n test cases follow. Each test case starts with the number of lines which is followed by their content.

Output

For each test case write the determined value in a separate line.

Example

Input:

```
2
3
1
2 1
1 2 3
4
1
1 2
4 1 2
2 3 1 1
```

Output:

```
5
9
```

Warning: large Input/Output data, be careful with certain languages

---

Added by:	Łukasz Kuszner
Date:	2004-12-01
Time limit:	2s
Source limit:	256B
Languages:	All
Resource:	6-th International Olympiad In Informatics July 3-10. 1994. Stockholm - Sweden, Problem 1

SPOJ Problem Set (classical)

238. Getting Rid of the Holidays (Act II)

Problem code: HOLIDAY2

As King Johnny's temporary indisposition lengthens from days to weeks, and you still hold the office of Regent of Byteland, you begin to feel that acting king is not all that much fun. You encounter various absurdly weird problems. For instance, you find that contrary to your expectations the recent removal of holidays brought about a decrease in the efficiency of the kingdom's workforce.

There appears to be only one rational explanation for all this. It seems that although every holiday occurs every a fixed number of days, the periods between consecutive holidays are long and very irregular. And it is the lack of regularity that is the root of the problem.

So, you decide it is time to tackle the problem once again, and solve it properly this time. Your main purpose is to establish an r-day working rhythm (for some integer r). Workers will work for (r-1) days, have a single day off, work for another (r-1) days, and so on. The rhythm must be arranged in such a way that holidays only ever occur on the day off work. Choose exactly k of the n holidays to remove in such a way as to be able to establish a working rhythm of the maximum possible length r.

Solve the problem in at most 4kB of source code.

Input

The first line of input contains a single integer t<=100 - the number of test cases. t test case descriptions follow.

For each test case, the first line contains two space separated integers n k (1<=k < n<=100), denoting the total number of holidays and the number of holidays to be removed. The next line contains n space separated integers, the i-th being t_i ($1<=t_i<=10^{18}$) - the number of days every which the i-th holiday occurs.

Output

For each test case, output one line containing an increasing sequence of exactly k integers - the numbers of the holidays to be removed (holidays are numbered in the input order from 1 to n).

Example

Input:

```
2
6 4
1 3 4 5 6 1
8 4
200 125 200 999 380 500 200 500
```

Output:

```
1 3 4 6
2 4 5 6
```

(In the first test case r is equal to 3 days, in the second case it is equal to 100 days. For the second test case the output ’1 2 4 5’, ’2 3 4 5’, ’2 4 5 6’, ’2 4 5 7’ or ’2 4 5 8’ is also correct.)

Added by: Adrian Kosowski
Date: 2004-12-07
Time limit: 17s
Source limit: 4096B
Languages: All
Resource: DASM Programming League 2004, problemset 4

SPOJ Problem Set (classical)

239. Tour de Byteland

Problem code: BTOUR

As the mayor of Byteland's term of office draws to a close, he starts his preparations for reelection. For the first time in the 40 years of his political career his chances of victory seem somewhat uncertain. His main cause of worry are the disturbing results of an opinion poll which state that over 90% of the citizens regard the mayor as a portly, heavily smoking individual who sleeps in his armchair more or less all day.

After careful consultation with his public relations director, the mayor has decided to change his image. He is going to organise, sponsor and compete in... Byteland's first bicycle race! Quite naturally, the only relevant part of the race is the media coverage of the mayor; everything else is to be done at minimum cost. The street-map of Byteland consists of a not necessarily planar system of bi-directional street segments connecting intersections, in such a way that between 0 and 4 street segments meet at an intersection. The cyclists are to ride round and round a simple loop (a fixed, closed route consisting of several street segments, such that a cyclist goes along a street and through an intersection exactly once in each round). For innumerable reasons (not so difficult to guess at) the mayor would like to choose the shortest possible route for the race (in the sense of total street length). Help him determine the length of such a loop, and tell him how many different shortest loops he can choose from when organising the race.

Input

The input starts with a line containing a single integer t<=200, the number of test cases. t test cases follow.

Each test case begins with a line with two integers n m, denoting the number of intersections and the number of streets in Byteland, respectively (1<=n<=1000). m lines follow, each containing three integers u_i v_i d_i, denoting the end points and the length of the i-th street segment, respectively ($1<=u_i<=v_i<=n$, $1<=d_i<=10^6$).

Output

For each test case output a single line containing exactly two space separated non-negative integers d c - the length of the shortest possible race loop, and the number of routes of this length in the graph. Output 0 0 if the race cannot be held.

Example

Input:

```
2
3 2
1 2 1
1 3 2
4 6
1 2 5
1 4 5
```

```
2 3 4
2 4 5
3 4 5
3 1 5
```

Output:

```
0 0
14 2
```

Added by: Krzysztof Kluczek
Date: 2004-12-09
Time limit: 17s
Source limit: 50000B
Languages: All
Resource: DASM Programming League 2004, problemset 4

SPOJ Problem Set (classical)

241. Arranging the Blocks

Problem code: BLOCKS

A group of n children are playing with a set of n^2 flat square blocks. Each block is painted from above with one colour, and there are no more than 2 blocks of each colour. The blocks are initially arranged in an n x n square forming some sort of picture.

The children have been provided with some other n x n picture and asked to rearrange the blocks to that form. Since this is not really what they enjoy doing most, they intend to solve the task together and spend as little time on it as possible. Thus, every minute each child chooses a single 1 x n row or n x 1 column of blocks to rearrange. This row/column may never intersect with rows/columns chosen by other children in the same minute. A child takes one minute to perform any rearrangement (permutation) of the blocks within its row/column it likes.

Determine whether the children can perform their task of converting one block image into the other, and if so -- find the minimum possible time in minutes required to achieve this.

Input

The input starts with a line containing a single integer t<=200, the number of test cases. t test cases follow. Each test case begins with a line containing integer n (1<=n<=500). The next n lines contain n integers $P_{i,j}$ each, forming a bitmap matrix representing the colours of the blocks in their initial configuration ($1<=P_{i,j}<=n^2$). The following n lines contain n integers $Q_{i,j}$ each, corresponding to the matrix for the final configuration ($1<=Q_{i,j}<=n^2$).

Output

For each test case output a line with a single non-negative integer corresponding to the number of minutes required to transform matrix P into matrix Q, or the word `no` if no such transformation is possible.

Example

Input:

```
3
3
1 3 4
2 1 3
2 5 5
3 1 3
2 1 2
4 5 5
3
1 2 3
4 5 6
7 8 9
1 5 6
4 2 9
```

```
7 8 3
2
1 2
1 2
1 3
1 2
```

Output:

```
2
1
no
```

The actions taken in the first test case are illustrated below.

2 step transformation: 134 213 255 -> 413 312 255 -> 313 212 455

Warning: enormous Input/Output data, be careful with certain languages

---

Added by: Adrian Kosowski
Date: 2004-12-09
Time limit: 17s
Source limit: 50000B
Languages: All
Resource: DASM Programming League 2004, problemset 4

SPOJ Problem Set (classical)

243. Stable Marriage Problem

Problem code: STABLEMP

There are given *n* men and *n* women. Each woman ranks all men in order of her preference (her first choice, her second choice, and so on). Similarly, each man sorts all women according to his preference. The goal is to arrange *n* marriages in such a way that if a man *m* prefers some woman *w* more than his wife, then *w* likes her husband more than *m*. In this way, no one leaves his partner to marry somebody else. This problem always has a solution and your task is to find one.

Input

The first line contains a positive integer *t*<=100 indicating the number of test cases. Each test case is an instance of the stable marriage problem defined above. The first line of each test case is a positive integer *n*<=500 (the number of marriages to find). The next *n* lines are the woman's preferences: *i*th line contains the number *i* (which means that this is the list given by the *i*th woman) and the numbers of men (the first choice of *i*th woman, the second choice,...). Then, the men's preferences follow in the same format.

Output

For each test case print *n* lines, where each line contains two numbers *m* and *w*, which means that the man number *m* and the woman number *w* should get married.

Example

Input:

```
2
4
1 4 3 1 2
2 2 1 3 4
3 1 3 4 2
4 4 3 1 2
1 3 2 4 1
2 2 3 1 4
3 3 1 2 4
4 3 2 4 1
7
1 3 4 2 1 6 7 5
2 6 4 2 3 5 1 7
3 6 3 5 7 2 4 1
4 1 6 3 2 4 7 5
5 1 6 5 3 4 7 2
6 1 7 3 4 5 6 2
7 5 6 2 4 3 7 1
1 4 5 3 7 2 6 1
2 5 6 4 7 3 2 1
3 1 6 5 4 3 7 2
4 3 5 6 7 2 4 1
5 1 7 6 4 3 5 2
6 6 3 7 5 2 4 1
```

```
7 1 7 4 2 6 5 3
```

Output:

```
1 3
2 2
3 1
4 4
1 4
2 5
3 1
4 3
5 7
6 6
7 2
```

Warning: large Input/Output data, be careful with certain languages

---

Added by: Darek Dereniowski
Date: 2004-12-13
Time limit: 1s-3s
Source limit:50000B
Languages: All
Resource: problem known as the *Stable Marriage Problem*

SPOJ Problem Set (classical)

245. Square Root

Problem code: SQRROOT

In this problem you have to find the Square Root for given number. You may assume that such a number exist and it will be always an integer.

Solutions to this problem can be submitted in C, C++, Pascal, Algol, Fortran, Ada, Ocaml, Prolog, Whitespace, Brainfk and Intercal only.**

Input

t - the number of test cases [t <= 50]
then *t* positive numbers follow, each of them have up to 800 digits in decimal representation.

Output

Output must contain exactly t numbers equal to the square root for given numbers. See sample input/output for details.

Example

Input:
```
3
36
81
226576
```

Output:
```
6
9
476
```

Added by:	Roman Sol
Date:	2004-12-15
Time limit:	5s
Source limit:	50000B
Languages:	C C99 strict C++ PAS gpc PAS fpc ASM D FORT ADA SCM guile CAML PRLG WSPC BF ICK
Resource:	ZCon 2005

SPOJ Problem Set (classical)

247. Chocolate

Problem code: CHOCOLA

We are given a bar of chocolate composed of *m*n* square pieces. One should break the chocolate into single squares. Parts of the chocolate may be broken along the vertical and horizontal lines as indicated by the broken lines in the picture.

A single break of a part of the chocolate along a chosen vertical or horizontal line divides that part into two smaller ones. Each break of a part of the chocolate is charged a cost expressed by a positive integer. This cost does not depend on the size of the part that is being broken but only depends on the line the break goes along. Let us denote the costs of breaking along consecutive vertical lines with x_1, x_2, ..., x_{m-1} and along horizontal lines with y_1, y_2, ..., y_{n-1}.

The cost of breaking the whole bar into single squares is the sum of the successive breaks. One should compute the minimal cost of breaking the whole chocolate into single squares.

[IMAGE]

For example, if we break the chocolate presented in the picture first along the horizontal lines, and next each obtained part along vertical lines then the cost of that breaking will be $y_1+y_2+y_3+4*(x_1+x_2+x_3+x_4+x_5)$.

Task

Write a program that for each test case:

- Reads the numbers x_1, x_2, ..., x_{m-1} and y_1, y_2, ..., y_{n-1}
- Computes the minimal cost of breaking the whole chocolate into single squares, writes the result.

Input

One integer in the first line, stating the number of test cases, followed by a blank line. There will be not more than 20 tests.

For each test case, at the first line there are two positive integers *m* and *n* separated by a single space, 2 <= *m*,*n* <= 1000. In the successive *m*-1 lines there are numbers x_1, x_2, ..., x_{m-1}, one per line, 1 <= x_i <= 1000. In the successive *n*-1 lines there are numbers y_1, y_2, ..., y_{n-1}, one per line, 1 <= y_i <= 1000.

The test cases will be separated by a single blank line.

Output

For each test case : write one integer - the minimal cost of breaking the whole chocolate into single squares.

Example

Input:

```
1

6 4
2
1
3
1
4
4
1
2
```

Output:

```
42
```

Added by: Thanh-Vy Hua
Date: 2004-12-23
Time limit: 3s
Source limit: 50000B
Languages: All
Resource: 10th Polish Olympiad in Informatics, stage 1

SPOJ Problem Set (classical)

260. Containers

Problem code: CTAIN

We are given n containers, where 1 <= n <= 4. At the beginning all of them are full of water. The liter capacity of the i-th container is a natural number o_i satisfying inequalities 1 <= o_i <= 49.
Three kinds of moves can be made:

1. Pouring the whole content of one container into another. This move can be made unless there is too little room in the second container.
2. Filling up one container with part of the water from another one.
3. Pouring away the whole content of one container into a drain.

Task

Write a program that for each test case:

- Reads the number of containers n, the capacity of each container and the requested final amount of water in each container.
- Verifies, whether there exist a series of moves which leads to the requested final situation, and if there is one, the program computes the minimal number of moves leading to the requested situation,
- Writes the result. The result should be the minimal number of moves leading to the requested final situation, or one word "NO" if there is no such a sequence of moves.

Input

One integer in the first line, stating the number of test cases, followed by a blank line. There will be not more than 20 tests.

For each test case, at the first line, one positive integer n is written, n <= 4, this is the number of containers. There are n positive integers written in the second line. These are the capacities of the containers (the i-th integer o_i denotes the capacity if the i-th container,1 <= o_i <= 49). In the third line there are written n numbers. These are the requested final volumes of water in the containers (the i-th integer w_i denotes the requested final volume of water in the i-th container, 0 <= w_i <= o_i). All integers in the second and the third line are separated by single spaces.

The test cases will be separated by a single blank line.

Output

For each test case : write one integer - the minimal number of moves which lead to the requested final situation or write only one word "NO" if it is not possible to reach the requested final situation making only allowed moves.

Example

Input:

```
2

3
3 5 5
0 0 4

2
20 25
10 16
```

Output:

```
6
NO
```

Added by:	Thanh-Vy Hua
Date:	2004-12-24
Time limit:	5s
Source limit:	50000B
Languages:	All
Resource:	3rd Polish Olympiad in Informatics, stage 1

SPOJ Problem Set (classical)

261. Triangle Partitioning

Problem code: TRIPART

A triangle can be divided into two equal triangles by drawing a median on its largest edge (in the figure below such a division is shown with the red line). Then the smaller two triangles can be divided in similar fashion into equal triangles (shown in the picture below with blue lines). This process can continue forever.

[IMAGE]

Some mathematicians have found that when we split a triangle into smaller ones using the method specified above we have only some "styles" of triangles that only differ in size. So now given the lengths of the sides of the triangle your job is to find out how many different styles of small triangles we have. (Two triangles are of same style if they are similar.)

Input

First line of the input file contains an integer *N (0 < N < 35)* that indicates how many lines of input there are.

Each line contains three integers *a, b, c (0 < a,b,c < 100)* which indicate the sides of a valid triangle. (A valid triangle means a real triangle with positive area.)

Output

For each line of input you should produce an integer *T*, which indicates the number of different styles of small triangles, formed for the triangle at input. Look at the example for details. You can safely assume that for any triangle *T* will be less than *100*.

Example

Input:
```
2
3 4 5
12 84 90
```

Output:
```
3
41
```

Added by: Thanh-Vy Hua
Date: 2004-12-24
Time limit: 1s
Source limit:50000B
Languages: All
Resource: Thanh Vy Hua Le, special thanks to my friends in EPS

SPOJ Problem Set (classical)

262. Connections

Problem code: CONNECT

Byteotian Ministry of Infrastructure has decided to create a computer program that helps to find quickly the lengths of routes between arbitrary towns. It would be small wonder if the inhabitants of Byteotia always wanted to find the shortest route. However, it happens that they want to know the *k*-th shortest route. Moreover, cycles in routes are possible, i.e. routes that have recurring towns.

For example, if there are 4 routes between two towns and their lengths are 2, 4, 4 and 5, then the length of the shortest connection is 2, the second shortest is 4, the third is 4, and the fourth is 5.

Task

Write a program that for each test case:

- Reads a description of Byteotian road network and queries concerning lengths of journey routes.
- Computes and writes answers to the queries read.

Input

One integer in the first line, stating the number of test cases, followed by a blank line. There will be not more than 15 tests.

For each test case, at the first line, there are two positive integers *n* and *m*, separated by a single space, $1 <= n <= 100$, $0 <= m <= n^2-n$. They are the number of towns in Byteotia and the number of roads connecting the towns, respectively. The towns are numbered from 1 to *n*.

In each of *m* successive lines there are three integers separated by single spaces: *a*, *b* and *l*, $a <> b$, $1 <= l <= 500$. Each triple describes one one-way road of length *l* enabling to move from the town *a* to *b*. For each two towns there exist at most one road that enables to move in the given direction.

In the following line there is one integer *q*, $1 <= q <= 10000$, denoting the number of queries. In the successive *q* lines there are queries written, one per line. Each query has a form of three integers separated by single spaces: *c*, *d* and *k*, $1 <= k <= 100$. Such a query refers to the length of the *k*-th shortest route from the town *c* to the town *d*.

The test cases will be separated by a single blank line.

Output

For each test case, your program should write the answers to the queries read, one answer per line. In the i-th line the answer to the i-th query should be written: one integer equal to the length of the route being sought or -1, when such a route does not exist.

Each test case should be separated by a single blank line.

Example

Input:

```
1

5 5
1 2 3
2 3 2
3 2 1
1 3 10
1 4 1
8
1 3 1
1 3 2
1 3 3
1 4 2
2 5 1
2 2 1
2 2 2
1 1 2
```

Output:

```
5
8
10
-1
-1
3
6
-1
```

Added by: Thanh-Vy Hua
Date: 2004-12-25
Time limit: 5s
Source limit: 50000B
Languages: All
Resource: 10th Polish Olympiad in Informatics, stage 2

SPOJ Problem Set (classical)

263. Period

Problem code: PERIOD

For each prefix of a given string ***S*** with ***N*** characters (each character has an ASCII code between 97 and 126, inclusive), we want to know whether the prefix is a periodic string. That is, for each ***i*** (2 <= ***i*** <= ***N***) we want to know the largest ***K*** > 1 (if there is one) such that the prefix of ***S*** with length ***i*** can be written as A^K , that is ***A*** concatenated ***K*** times, for some string ***A***. Of course, we also want to know the period ***K***.

Input

The first line of the input file will contains only the number *T (1 <= T <= 10)* of the test cases.

Each test case consists of two lines. The first one contains ***N*** (2 <= ***N*** <= 1 000 000) - the size of the string ***S***. The second line contains the string ***S***.

Output

For each test case, output "Test case #" and the consecutive test case number on a single line; then, for each prefix with length **i** that has a period **K** > 1, output the prefix size **i** and the period **K** separated by a single space; the prefix sizes must be in increasing order. Print a blank line after each test case.

Example

Input:

```
2
3
aaa
12
aabaabaabaab
```

Output:

```
Test case #1
2 2
3 3

Test case #2
2 2
6 2
9 3
12 4
```

Added by: Thanh-Vy Hua
Date: 2004-12-26
Time limit: 3s
Source limit:50000B
Languages: All
Resource: ACM South Eastern European Region 2004

SPOJ Problem Set (classical)

264. Corporative Network

Problem code: CORNET

A very big corporation is developing its corporate network. At the beginning, each of the **N** enterprises of the corporation, numbered from 1 to **N**, organized its own computing and telecommunication center. Soon, for amelioration of the services, the corporation started to collect some enterprises in clusters, each of them served by a single computing and telecommunication center as follows. The corporation chose one of the existing centers **I** (serving the cluster **A**) and one of the enterprises **J** in some other cluster **B** (not necessarily the center) and linked them with a telecommunication line. The length of the line between the enterprises **I** and **J** is |**I** - **J**|(mod 1000). In such a way two old clusters are joined to form a new cluster, served by the center of the old cluster **B**. Unfortunately after each join the sum of the lengths of the lines linking an enterprise to its serving center could be changed and the end users would like to know what is the new length.

Write a program to keep trace of the changes in the organization of the network that is able at each moment to answer the questions of the users.

Input

The first line of the input file will contains only the number **T** of the test cases (1 <= **T** <= 5). Each test will start with the number **N** of enterprises (5<=**N**<=20000). Then some number of lines (no more than 200000) will follow with one of the commands:

E I- asking the length of the path from the enterprise **I** to its serving center at the moment; **I I J** - informing that the serving center **I** is linked to the enterprise **J**. The test case finishes with a line containing the word **O**. There are fewer **I** commands than **N** commands.

Output

The output should contain as many lines as the number of **E** commands in all test cases. Each line must contain a single number - the requested sum of lengths of lines connecting the corresponding enterprise with its serving center.

Example

Input:

```
1
4
E 3
I 3 1
E 3
I 1 2
E 3
I 2 4
E 3
O
```

Output:

```
0
2
3
5
```

Added by: Thanh-Vy Hua
Date: 2004-12-27
Time limit: 1s
Source limit:50000B
Languages: All
Resource: ACM South Eastern European Region 2004

SPOJ Problem Set (classical)

272. Cave Exploration

Problem code: CAVE

A long time ago one man said that he had explored the corridors of one cave. This means that he was in all corridors of the cave. Corridors are really horizontal or vertical segments. A corridor is treated as visited if he was in at least one point of the corridor.

Now you want to know if this is true. You have a map of the cave, and you know that the explorer used the following algorithm: he turns left if he can, if he can't he goes straight, if he can't he turns right, if he can't he turns back. Exploration ends when the man reaches the entry point for the second time. Your task to count how many corridors weren't visited by explorer.

Input

In the first line there is an integer **T (T <= 20)** - the number of different maps. For each map in the first line there is an integer **N (N <= 1000)** - the number of corridors. It is known that no two vertical corridors have a common point and no two horizontal corridors have a common point. The next **N** lines contain the following information: the line starts with one of the characters 'V' or 'H' - vertical or horizontal corridor. Then one Y-coordinate and two X-coordinates are given for a horizontal corridor or one X-coordinate and two Y-coordinates for a vertical corridor. The last line for each map contains the X and Y coordinates of the entry point (start and end point of travel) and the direction ('W' - left, 'E' - right, 'N' - up and 'S' - down). You may assume that: the entry point is not located at the cross-point of two corridors, and the explorer can always move forward in the direction given in the input. All coordinates are integers and do not exceed *32767* by absolute value and there are no more than *500* vertical corridors and no more than *500* horizontal corridors.

Output

For each map the program has to print the number of unvisited corridors (in a separate line).

Example

Input:

```
2
6
H 0 6 0
H 2 1 6
V 1 0 4
V 5 3 0
V 3 0 2
H 1 2 4
6 0 W
1
V 0 -5 5
0 0 S
```

Output:

```
1
0
```

An example of a cave

Added by: Thanh-Vy Hua
Date: 2004-12-31
Time limit: 3s
Source limit: 50000B
Languages: All
Resource: ACM South Eastern European Region 2004

SPOJ Problem Set (classical)

274. Johnny and the Watermelon Plantation

Problem code: WMELON

Shortly after his abdication from the Bytelandian throne Johnny decided to go into farming. Water melons were a natural choice as his first crop ever, since they seemed easy enough to grow and look after. So, he sold all his beer bottles and for the money he purchased a 1km x 1km square field. Here it was that he planted the water melon seeds. (The word 'planted' is really a bit of a euphemism for walking across a field gorging on a water melon and spitting out the pips but, for the sake of politeness, let us leave it this way).

To everyone's surprise a lot of the seeds sprouted stems, and soon enough many of the plants showed signs of fruit (and some had even more than one!). Then quite unexpectedly, when the water melons were still a little too unripe to eat, winter set in. Johnny knows that he has to construct a green house to protect the field but, with his rather limited budget, he cannot afford the glass to cover the whole area. He has decided that it is enough that k fruit survive the ordeal under a glazed roof. For reasons of architectural planning in Byteland it is necessary that the green house be a rectangle with sides parallel to the edges of the plot.

You have been requested to help Johnny minimise investement costs. Since glass is paid for by the square meter, design a green house with the smallest possible area fulfilling the imposed conditions.

Input

The first line of input contains the integer t<=100, the number of test cases. t test cases follow.

Every test case begins with a line containing two integers n k, denoting the total number of plants and the number of water melon fruit to be protected, respectively (1<=n<=1000, 1<=k<=10^6, k doesn't exceed the total number of fruit in the plantation). Each of the next n lines describes a single plant, the i-th line containing three integers x_i y_i f_i - the X and Y coordinates of the plant, and the number of water melon fruit on it, respectively (1<=x_i, y_i, f_i<=1000).

Output

For each test case output a single integer, denoting the area of the smallest possible rectangular glass house with horizontal and vertical edges, sufficient to cover at least k fruit of the plantation.

Example

Input:

```
1
6 11
1 1 2
1 2 2
3 1 2
3 2 3
4 2 5
```

```
3 3 2
```

Output:

```
2
```

Illustration of sample test data

Added by:	Adrian Kosowski
Date:	2005-01-03
Time limit:	17s
Source limit:	50000B
Languages:	All
Resource:	DASM Programming League 2004, problemset 5 (acknowledgement to Thanh Vy Hua Le)

SPOJ Problem Set (set5)

275. The Water Ringroad

Problem code: WATERWAY

There is a land far, far away were the entire population dwells in walled cities at the peaks of mountains on the circumference of a plateau known as The Circle. The High Councillors of the cities developed an intricate system of communication: the cities were connected into a cycle by a perfectly round waterway. If need arose, a small paper boat with a message tied to its sail was released into the waterway and was guided by its solitary crew member (a small tin soldier) from one city to the next, and so on, until it reached its destination. Some segments of the waterway were only passable in one direction (due to waterfalls), and so there may have been pairs of cities for which communication was impossible.

As the centuries went by, the system slowly began to show its weaknesses. The waterway was so narrow that two boats going in opposite directions could never pass each other. To make matters worse, some of the more enterprising cities replaced the tin soldier by a plastic one to increase the speed of the boat, and the faster boats had to queue up behind the slower ones, and everyone got very angry indeed. The councillors gathered to address the problem and found that the best course of action would be to construct two separate channels between every pair of communicating cities A and B: one for carrying messages from A to B, the other from B to A (if communication was impossible in some direction in the old waterway, it needn't be enabled in the new one).

The High Priests of the Circle were the first to protest against the plan. They insisted that any waterway ever built should be circular and go round all the cities in the same manner as the original one, and the route of any boat must always be a perfect arc between any two adjacent cities. So the newly designed channels would in fact have to be composed of sets of adjacent fragments of circles, without any two channels sharing an arc.

The engineers have quite rightly pointed out that the new circles will be prone to the same problem of waterfalls on the same sections as the original waterway. Bearing this in mind, given a map of the old waterway, calculate the smallest possible number of circles the new waterway may consist of.

Input

Input begins with integer t<=100, the number of test cases. t test cases follow.

Each test case consists of two lines. The first contains a single integer n (3<=n<=100000), the number of cities around the Circle. The second line is a description of the old waterway - a sequence of exactly n characters 'A', 'B' or 'C', without separating spaces, terminated by a new line. These characters correspond to the state of the arcs between cities 1 and 2, 2 and 3,..., n-1 and n, n and 1, respectively, and mean: 'A' - the arc is passable when going anticlockwise, 'B' - the arc is passable in both directions, 'C' - the arc is passable when going clockwise.

Output

For each test case output a line, containing a single integer - the number of circles required for the new waterway.

Example

Input:

```
2
3
AAA
4
BACB
```

Output:

```
3
5
```

A solution to the first test case which requires 3 circles is presented below.

Illustration to test case 1

Added by: Adrian Kosowski
Date: 2005-01-03
Time limit: 17s
Source limit: 50000B
Languages: All except: C99 strict
Resource: DASM Programming League 2004, problemset 5

SPOJ Problem Set (classical)

277. City Game

Problem code: CTGAME

Bob is a strategy game programming specialist. In his new city building game the gaming environment is as follows: a city is built up by areas, in which there are streets, trees, factories and buildings. There is still some space in the area that is unoccupied. The strategic task of his game is to win as much rent money from these free spaces. To win rent money you must erect buildings, that can only be rectangular, as long and wide as you can. Bob is trying to find a way to build the biggest possible building in each area. But he comes across some problems - he is not allowed to destroy already existing buildings, trees, factories and streets in the area he is building in.

Each area has its width and length. The area is divided into a grid of equal square units.The rent paid for each unit on which you're building stands is 3$.

Your task is to help Bob solve this problem. The whole city is divided into **K** areas. Each one of the areas is rectangular and has a different grid size with its own length **M** and width **N**. The existing occupied units are marked with the symbol **R**. The unoccupied units are marked with the symbol **F**.

Input

The first line of the input contains an integer **K** - determining the number of datasets. Next lines contain the area descriptions. One description is defined in the following way: The first line contains two integers-area length **M**<=1000 and width **N**<=1000, separated by a blank space. The next **M** lines contain **N** symbols that mark the reserved or free grid units,separated by a blank space. The symbols used are:

R - reserved unit

F - free unit

In the end of each area description there is a separating line.

Output

For each data set in the input print on a separate line, on the standard output, the integer that represents the profit obtained by erecting the largest building in the area encoded by the data set.

Example

Input:

```
2
5 6
R F F F F F
F F F F F F
R R R F F F
F F F F F F
F F F F F F
```

```
5 5
R R R R R
R R R R R
R R R R R
R R R R R
R R R R R
```

Output:

```
45
0
```

Added by: Thanh-Vy Hua
Date: 2005-01-08
Time limit: 3s
Source limit: 50000B
Languages: All
Resource: ACM South Eastern European Region 2004

SPOJ Problem Set (classical)

278. Bicycle

Problem code: BICYCLE

Peter likes to go to school by bicycle. But going by bicycle on sidewalks is forbidden and going along roads is dangerous. That's why Peter travels only along special bicycle lanes. Fortunately Peter's home and school are in the immediate proximity of such paths. In the city where Peter lives there are only two bycycle lanes. Both lanes have the form of a circle. At the points where they cross it is possible to move from one path to the other. Peter knows the point where he enters the road and the point at which it is necessary to leave to enter the school. Peter is interested in the question: "What is the minimal distance he needs to cover along the lanes to get to school?"

Input

t - the number of test cases [*t*<=100], then *t* test cases follow.
The first 2 lines of each test case contain the description of the bicycle lanes:
x1 y1 r1 - 3 integers (*x1, y1* - coordinates of the center of the 1st circle, *r1* - radius of 1st circle)
x2 y2 r2 - 3 integers (*x2, y2* - coordinates of the center of the 2nd circle, *r2* - radius of 2nd circle)
-200 <= *x1, x2, y1, y2* <= 200
0 <= *r1, r2* <= 200
Next 2 lines contain the coordinates of Peter's home and school:
px1, py1 - 2 real numbers
px2, py2 - 2 real numbers

You may assume that this points lie on the circle with high accuracy (10^{-8}). Both points may lie on the same circle.

Output

For each test case output the minimum distance that Peter needs to go from home to get to school. The precision of the answer must be under 0.0001. If it's impossible to get to school using the bicycle lanes output -1.

Example

Input:

```
3

0 0 5
4 0 3
3.0 4.0
1.878679656440357 -2.121320343559643

0 0 5
4 0 3
4.0 3.0
4.0 -3.0

0 0 4
```

```
10 0 4
4.0 0.0
6.0 0.0
```

Output:

```
8.4875540166
6.4350110879
-1
```

Illustration of sample test data

Added by: Roman Sol
Date: 2005-01-13
Time limit: 1s
Source limit: 50000B
Languages: All
Resource: 5th Russian National Command Olympiad for schoolboys in programming

SPOJ Problem Set (classical)

279. Interesting number

Problem code: INUMBER

For the given number *n* find the minimal positive integer divisable by *n*, with the sum of digits equal to *n*.

Input

t - the number of test cases, then *t* test cases follow. (t <= 50)
Test case description:
n - integer such that 0 < *n* <= 1000

Output

For each test case output the required number (without leading zeros).

Example

Input:
```
2
1
10
```

Output:
```
1
190
```

Added by: Roman Sol
Date: 2005-01-13
Time limit: 7s
Source limit: 4096B
Languages: All
Resource: XII team championship of St.-Petersburg in programming

SPOJ Problem Set (classical)

280. Lifts

Problem code: LIFTS

Serj likes old games very much. Recently he has found one arcade game in his computer. When controlling the hero it is necessary to move on a map and collect various items. At a certain stage of the game Serj has faced an unexpected problem. To continue his adventures the hero should get past over a chasm. For this purpose it is possible to use consistently located lifts which look like horizontal platforms. Each lift moves up-down vertically between some levels. The hero can pass between the next adjacent platform, however it can be done only at the moment when they are at the same level. Similarly, passing from the edge of a chasm onto the lift and vice versa is only possible at the moment when the lift appears on the level of the edge.

Each lift has a width equal to 4 meters. At the beginning the hero is in at a distance of two meters from the edge of a chasm. He should finish travel two meters after the opposite edge of the chasm. The hero moves at a speed of 2 meters a second. Thus, if the hero is in the initial position or in the center of the lift and wishes to pass to the next lift (or to descend from last lift onto the opposite edge of a chasm), he should begin movement exactly one second before they meet at one level. In two seconds the hero appears in the center of the next lift (or in the final position on the other side).

The edges of the chasm are at the same level. For each lift the range of heights between which it moves, its initial position and the direction of movement at the initial moment are given. All lifts move with a speed of one meter a second. Find out whether the hero can get over to the opposite edge of the chasm, and if so what the minimal time required for this purpose is.

A sample illustration

Input

t - the number of test cases, then *t* test cases follows.
[empty line]
A test case begins with *n* - the number of lifts, a positive integer ($n \leq 100$), then n lines follow. The i-th line ($0 < i \leq n$) contains four integers *li ui si di*, where: *li* - lowest position of the lift, *ui* - highest position of the lift, *si* - initial position of the lift, *di* - initial direction of movement (1 means up, -1 means down); ($-100 \leq li \leq si \leq ui \leq 100$, $l1 < ui$).

Output

For each test case output the minmal time in seconds, required to get to the opposite edge of the chasm. If it is impossible output -1.

Example

Input:

```
1

4
```

```
-1 2 1 -1
0 3 0 1
-4 0 0 -1
-2 1 0 -1
```

Output:

```
29
```

Added by: Roman Sol
Date: 2005-01-17
Time limit: 3s
Source limit: 10000B
Languages: All
Resource: 5th Russian National Command Olympiad for schoolboys in programming

SPOJ Problem Set (classical)

282. Muddy Fields

Problem code: MUDDY

Rain has pummeled on the cows' field, a rectangular grid of R rows and C columns (1 <= R <= 50, 1 <= C <= 50). While good for the grass, the rain makes some patches of bare earth quite muddy. The cows, being meticulous grazers, don't want to get their hooves dirty while they eat.

To prevent those muddy hooves, Farmer John will place a number of wooden boards over the muddy parts of the cows' field. Each of the boards is 1 unit wide, and can be any length long. Each board must be aligned parallel to one of the sides of the field.

Farmer John wishes to minimize the number of boards needed to cover the muddy spots, some of which might require more than one board to cover. The boards may not cover any grass and deprive the cows of grazing area but they can overlap each other.

Compute the minimum number of boards FJ requires to cover all the mud in the field.

Input

t - the number of test cases, then *t* test cases follows.
Each test case is of the following form:
Two space-separated integers: *R* and *C*, then R lines follows
Each line contains a string of C characters, with '*' representing a muddy patch, and '.' representing a grassy patch. No spaces are present.

Output

For each test case output a single integer representing the number of boards FJ needs.

Example

Input:
```
1
4 4
*.*.
.***
***.
..*.
```

Output:
```
4
```

Output details:
```
Boards 1, 2, 3 and 4 are placed as follows:
1.2.
.333
444.
..2.
Board 2 overlaps boards 3 and 4.
```

Added by: Roman Sol
Date: 2005-01-19
Time limit: 5s
Source limit:30000B
Languages: All
Resource: USACO January 2005 Gold Division

SPOJ Problem Set (classical)

283. Naptime

Problem code: NAPTIME

Goneril is a very sleep-deprived cow. Her day is partitioned into N (3 <= N <= 3,830) equal time periods but she can spend only B (2 <= B < N) not necessarily contiguous periods in bed. Due to her bovine hormone levels, each period has its own utility U_i (0 <= U_i <= 200,000), which is the amount of rest derived from sleeping during that period. These utility values are fixed and are independent of what Goneril chooses to do, including when she decides to be in bed.
With the help of her alarm clock, she can choose exactly which periods to spend in bed and which periods to spend doing more critical items such as writing papers or watching baseball. However, she can only get in or out of bed on the boundaries of a period.
She wants to choose her sleeping periods to maximize the sum of the utilities over the periods during which she is in bed. Unfortunately, every time she climbs in bed, she has to spend the first period falling asleep and gets no sleep utility from that period.
The periods wrap around in a circle; if Goneril spends both periods N and 1 in bed, then she does get sleep utility out of period 1.
What is the maximum total sleep utility Goneril can achieve?

Input

t - the number of test cases, then *t* test cases follow.
Each test case takes the following form:
Two space-separated integers: *N* and *B*, then *N* lines follows
Each line contains a single integer, *U_i*, between 0 and 200,000 inclusive

Output

For each test case output a single integer, the maximum total sleep utility Goneril can achieve.

Example

Input:
```
1
5 3
2
0
3
1
4
```

Output:
```
6
```

Input/Output details:
```
The day is divided into 5 periods, with utilities 2, 0, 3, 1, 4 in that
order. Goneril must pick 3 periods.
```

```
Goneril can get total utility 6 by being in bed during periods 4,
5, and 1, with utilities 0 [getting to sleep], 4, and 2
respectively.
```

Added by: Roman Sol
Date: 2005-01-19
Time limit: 5s
Source limit:50000B
Languages: All
Resource: USACO January 2005 Gold Division

SPOJ Problem Set (classical)

286. Selfish Cities

Problem code: SCITIES

Far, far away there is a world known as Selfishland because of the nature of its inhabitants. Hard times have forced the cities of Selfishland to exchange goods among each other. C1 cities are willing to sell some goods and the other C2 cities are willing to buy some goods (each city can either sell or buy goods, but not both). There would be no problem if not for the selfishness of the cities. Each selling city will sell its goods to one city only, and each buying city will buy goods from one city only.
Your goal is to connect the selfish cities in such a way that the amount of exchanged goods is maximalized.

Input

The first line contains a positive integer t<=1000 indicating the number of test cases. Each test case is an instance of the problem defined above. The first line of each test case is a pair of positive integers C1 and C2 (the number of cities wanting to sell their goods C1<=100 and the number of cities wanting to buy goods C2<=100). The lines that follow contain a sequence of (c1,c2,g) trios ending with three zeros. (c1,c2,g) means that the city c1 can offer the city c2 the amount of g<=100 goods.

Output

For each test case print the maximal amount of goods exchanged.

Example

Input:

```
3
3 2
1 1 10
2 1 19
2 2 11
3 2 1
0 0 0
4 4
1 1 6
1 2 6
2 1 8
2 3 9
2 4 8
3 2 8
4 3 7
0 0 0
3 2
1 1 10
2 1 21
2 2 11
3 2 1
0 0 0
```

Output:

```
21
29
22
```

Added by: Tomasz Niedzwiecki
Date: 2005-01-22
Time limit: 8s
Source limit: 50000B
Languages: All

SPOJ Problem Set (classical)

287. Smart Network Administrator

Problem code: NETADMIN

The citizens of a small village are tired of being the only inhabitants around without a connection to the Internet. After nominating the future network administrator, his house was connected to the global network. All users that want to have access to the Internet must be connected directly to the admin's house by a single cable (every cable may run underground along streets only, from the admin's house to the user's house). Since the newly appointed administrator wants to have everything under control, he demands that cables of different colors should be used. Moreover, to make troubleshooting easier, he requires that no two cables of the same color go along one stretch of street.

Your goal is to find the minimum number of cable colors that must be used in order to connect every willing person to the Internet.

Input

t [the number of test cases, t<=500]
n m k [*n* <=500 the number of houses (the index of the admin's house is 1)]
[*m* the number of streets, *k* the number of houses to connect]
$h_1\ h_2\ ...\ h_k$ [a list of *k* houses wanting to be conected to the network, 2<=h_i<=*n*]
[The next *m* lines contain pairs of house numbers describing street ends]
$e_{11}\ e_{12}$
$e_{21}\ e_{22}$
...
$e_{m1}\ e_{m2}$
[next cases]

Output

For each test case print the minimal number of cable colors necessary to make all the required connections.

Example

```
Input:
2
5 5 4
2 3 4 5
1 2
1 3
2 3
2 4
3 5
8 8 3
4 5 7
1 2
1 8
```

```
8 7
1 3
3 6
3 2
2 4
2 5
```

Output:

```
2
1
```

Illustration to the first example

Warning: large Input/Output data, be careful with certain languages

---

Added by: Tomasz Niedzwiecki
Date: 2005-01-23
Time limit: 17s
Source limit: 50000B
Languages: All
Resource: DASM Programming League 2004, problemset 6

SPOJ Problem Set (classical)

288. Prime or Not

Problem code: PON

Given the number, you are to answer the question: "Is it prime?"
Solutions to this problem can be submitted in C, C++, Pascal, Perl, Python, Ruby, Lisp, Hask, Ocaml, Prolog, Whitespace, Brainfk and Intercal only.**

Input

t - the number of test cases, then *t* test cases follows. [t <= 500]
Each line contains one integer: *N* [2 <= *N* <= 2^63-1]

Output

For each test case output string "YES" if given number is prime and "NO" otherwise.

Example

Input:
```
5
2
3
4
5
6
```

Output:
```
YES
YES
NO
YES
NO
```

Added by:	Roman Sol
Date:	2005-01-24
Time limit:	21s
Source limit:	5000B
Languages:	C C99 strict C++ PAS gpc PAS fpc PERL PYTH RUBY SCM guile SCM qobi LISP sbcl LISP clisp HASK CAML PRLG WSPC BF ICK
Resource:	ZCon 2005

SPOJ Problem Set (classical)

290. Polynomial Equations

Problem code: POLYEQ

You are given the polynomial F(x) as the sum of monomials. Each monomial has the form:
[*coefficient**]*x*[^*degree*] or [*coefficient*],
where *coefficient* and *degree* are integers such that -30000 <= *coefficient* <= 30000, 0 <= *degree* <= 6.
The parameters given in [] can be skipped.
In this problem you have to find all solutions of the equation: F(x)=0.

Input

t - the number of test cases, then *t* test cases follow. [*t* <= 100]
Each line contains one polynomial *F(x)* given as string *s* in the form described above.
The length of string *s* is not more than 300 characters.

Output

For each test case output all solutions (including repeated) of the given equation in non-decreasing order. All solutions lie within the interval [-100.0; 100.0]. Each solution must be given with an error of not more than 0.01. It's guaranteed that all solutions are real, not complex.

Example

Input:
```
2
x^4-6*x^3+11*x^2-6*x
-x^2+2*x-1
```

Output:
```
0.00 1.00 2.00 3.00
1.00 1.00
```

Added by: Roman Sol
Date: 2005-01-27
Time limit: 13s
Source limit:50000B
Languages: All
Resource: ZCon 2005

SPOJ Problem Set (classical)

291. Cube Root

Problem code: CUBERT

Your task is to calculate the cube root of a given positive integer. We can not remember why exactly we need this, but it has something in common with a princess, a young peasant, kissing and half of a kingdom (a huge one, we can assure you).

Write a program to solve this crucial task.

Input

The input starts with a line containing a single integer **t** <= 20, the number of test cases. **t** test cases follow.

The next lines consist of large positive integers of up to 150 decimal digits. Each number is on its own separate line of the input file. The input file may contain empty lines. Numbers can be preceded or followed by whitespaces but no line exceeds 255 characters.

Output

For each number in the input file your program should output a line consisting of two values separated by single space. The second value is the cube root of the given number, truncated (not rounded!) after the 10th decimal place. First value is a checksum of all printed digits of the cube root, calculated as the sum of the printed digits modulo 10.

Example

Input:

```
5
1

        8

   1000
2

33076161
```

Output:

```
1 1.0000000000
2 2.0000000000
1 10.0000000000
0 1.2599210498
6 321.0000000000
```

Added by: Thanh-Vy Hua
Date: 2005-01-29
Time limit: 5s
Source limit:50000B
Languages: All
Resource: ACM South Eastern European Region 2004

SPOJ Problem Set (classical)

292. Alibaba

Problem code: ALIBB

Alibaba the famous character of our childhood stories would like to be immortal in order to keep bringing happiness to children. In order to reach this status he needs to prove that he is still able to do some unusual things. There are **n** treasures, (**n** <= 10000) each in a different place located along a straight road. Each treasure has a time limit, after that it vanishes. Alibaba must take all the **n** treasures, and he must do it quickly. So he needs to figure out the order in which he should take the treasures before their deadlines starting from the most favorable position. Alibaba has the list of places and deadlines of the treasures. A place **i** is located at distance $\mathbf{d_i}$ from the leftmost end of the road. The time it takes to take a treasure is instantaneous.

Alibaba must find **the smallest time** by which he can take all the treasures.

Input

The first line of the input contains an integer **K** <= 10 - determining the number of datasets

Each data set in the input stands for a particular set of treasures. For each set of treasures the input contains the number of treasures, and the list of pairs place - deadline in increasing order of the locations. White spaces can occur freely between the numbers in the input.The input data are correct.

Output

For each set of data the program prints the result to the standard output on a separate line. The solution is represented by the smallest time by which Alibaba can take all the treasures before they vanish. If this is not possible then the output is "No solution".

Example

Input:

```
2
5
1 3
3 1
5 8
8 19
10 15
5
1 5
2 1
3 4
4 2
5 3
```

Output:

```
11
No solution
```

Added by: Thanh-Vy Hua
Date: 2005-01-29
Time limit: 5s
Source limit:50000B
Languages: All
Resource: ACM South Eastern European Region 2004

SPOJ Problem Set (classical)

293. Officers on the Beat

Problem code: OFBEAT

In the Middle Ages the capital of Byteland was surrounded by stout walls to protect the citizens from intruders. The gates of the city were well guarded and the drawbridge was lifted for the night, and everyone felt pretty happy and secure. At least, for a while.

With time the usual disadvantages of a walled city became apparent. As the population increased, crime flourished in the cramped living space. Eventually it all became so bad that the mayor decided to intervene. Some of the guards were reassigned from their usual occupation of reading newspapers in the guard posts near the gates, and told to start patrolling the city. Many of the officers were rather unhappy about all this, especially after the first men to go on the beat returned with bleeding noses and bumps on their heads. Sensing the low morale of the men, the Captain of the Guard, a bright young individual, decided to reinterpret the order he had received from the mayor. He decided that patrol officers would only go out in large groups and armed to the teeth, and would only move along a few carefully chosen streets from which they could see everything that was going on in the city without actually getting involved.

The city is laid out on a regular grid, with each street running North-South or East-West from one end of the city to the other (as far as the walls allow). Every point with integer coordinates is at an intersection of two streets, one leading North-South, the other East-West. The walls that surround the city form a simple polygon whose sides run directly alongside sections of some streets of the city.

Every street in the set of 'patrolled streets' chosen by the Captain intersects with at least one other patrolled street. Furthermore, if a point belongs to one of the streets of the city then it is visible from some point of one of the patrolled street (points see each other iff the line segment connecting them is a fragment of a street). Finally, the set of patrolled streets chosen by the Captain consists of the minimum possible number of streets.

Given a description of the capital of Byteland, find out how many of its streets were actually patrolled by guards after the Captain issued his order.

Input

The first line of input contains t - the number of test cases. t test cases follow.

For each test case, the first line contains a single integer n - the number of sections the city wall consists of (4 <= n <= 2000). The second line contains exactly n integers $a_1,...,a_n$ describing successive sections of wall (1<=|a_i|<=100000). Any two successive sections of wall are perpendicular to each other. The length of the i-th section is the absolute value of a_i, while its direction is described by the sign of a_i (positive means northbound or eastbound, negative - southbound or westbound when traversing the walls clockwise).

Output

For each test case output a single integer k - the number of elements of the patrolled set of streets selected by the Captain.

Example

Input:

```
1
14
+2 +2 +2 +2 -4 +2 +1 +2 -3 +2 -2 -8 +4 -2
```

Output:

```
4
```

Illustration of the sample test data. Blue lines indicate the set of patrolled streets

---

Added by: Adrian Kosowski
Date: 2005-02-05
Time limit: 17s
Source limit: 50000B
Languages: All
Resource: DASM Programming League 2004, problemset 6

SPOJ Problem Set (classical)

296. Teamwork is Crucial

Problem code: TWORK

In the late Middle Ages the University of Byteland was no different than any other university of the day. One of those gloomy places where philosophers brooded over the essence of life, theologians did likewise and quaralled with philosophers, while alchemists developed new caustic types of green shampoo in their futile search for gold. The thing that worried the Chancellor most was that none of the staff seemed to be in the least capable of making money in any form. When he complained about this to the Director of Human Resources, the Director came up with a brilliantly simple theory. He claimed that this lack of productivity was the direct consequence of the isolated model of work, and that wonders could be achieved by promoting teamwork.

The Director intends to assign every scientist to some 3-person workgroup. The members of the workgroup should then select which of them is to act as the group leader. And this of course is the root of the problem. Every scientist will tolerate either himself or one of his acquaintances as the leader of his group, but will never allow anyone else to have this privilege. So when creating workgroups it is necessary to bear in mind that every group should have at least one suitable candidate for the role of group leader, accepted by all its members.

Although everyone at the University knows *of* everyone else indirectly (as acquaintances of acquaintances of acquaintances of...), the number of direct acquaintances that every scientist has is relatively small - either equal to 2, or to 3. Even so, it ought to be possible to assign the vast majority of scientists to workgroups. Quite naturally, the dubious pleasure of performing this task has been left to you, the Acting University Algorithmist.

Input

Input starts with a single integer t, the number of test cases (t<=100). t test cases follow.

Each test case begins with a line containing two integers n m (4<=n<=m<=20000, n is the number of scientists and is divisible by 4). Exactly m lines follow containing a pair of integers a_i b_i each which denote that scientists a_i and b_i are acquaintances ($1<=a_i, b_i<=n$, each scientist has either 2 or 3 acquaintances). Acquaintanceship is mutual.

Output

For each test case, output a line containing a single integer k - the number of workgroups you have formed. In each of the next k lines output exactly 3 integers, representing the numbers of scientists belonging to respective workgroups.

Your solution will be regarded as incorrect if for some test case more than *25%* of all scientists are left without a valid assignment to a workgroup.

Example

Input:

```
1
8 10
1 2
1 3
2 5
4 6
3 7
2 3
5 6
6 7
7 8
8 4
```

Output:

```
2
1 3 7
4 5 6
```

Added by: Adrian Kosowski
Date: 2005-02-14
Time limit: 5s
Source limit: 50000B
Languages: All
Resource: DASM Programming League 2004, problemset 7

SPOJ Problem Set (classical)

297. Aggressive cows

Problem code: AGGRCOW

Farmer John has built a new long barn, with N (2 <= N <= 100,000) stalls. The stalls are located along a straight line at positions x1,...,xN (0 <= xi <= 1,000,000,000).
His C (2 <= C <= N) cows don't like this barn layout and become aggressive towards each other once put into a stall. To prevent the cows from hurting each other, FJ want to assign the cows to the stalls, such that the minimum distance between any two of them is as large as possible. What is the largest minimum distance?

Input

t - the number of test cases, then *t* test cases follows.
* Line 1: Two space-separated integers: N and C
* Lines 2..N+1: Line i+1 contains an integer stall location, xi

Output

For each test case output one integer: the largest minimum distance.

Example

Input:

```
1
5 3
1
2
8
4
9
```

Output:

```
3
```

Output details:

FJ can put his 3 cows in the stalls at positions 1, 4 and 8,
resulting in a minimum distance of 3.

Added by: Roman Sol
Date: 2005-02-16
Time limit: 2s
Source limit: 10000B
Languages: All
Resource: USACO February 2005 Gold Division

SPOJ Problem Set (classical)

300. Cable TV Network

Problem code: CABLETV

The interconnection of the relays in a cable TV network is bi-directional. The network is connected if there is at least one interconnection path between each pair of relays present in the network. Otherwise the network is disconnected. An empty network or a network with a single relay is considered connected. The safety factor **f** of a network with **n** relays is:

1. **n**, if the net remains connected regardless the number of relays removed from the net.
2. The minimal number of relays that disconnect the network when removed.

Exemplary illustration

For example, consider the nets from figure 1, where the circles mark the relays and the solid lines correspond to interconnection cables. The network (a) is connected regardless the number of relays that are removed and, according to rule (1), f=n=3. The network (b) is disconnected when 0 relays are removed, hence f=0 by rule (2). The network (c) is disconnected when the relays 1 and 2 or 1 and 3 are removed. The safety factor is 2.

Input

The input starts with a line containing a single integer **t** <= 20, the number of test cases. **t** test cases follow.

Write a program that computes the safety factor for the cable networks encoded by the data sets. Each data set starts with two integers: 0 <= **n** <=50,the number of relays in the net, and **m**, the number of cables in the net. Follow **m** data pairs (u,v), u < v, where u and v are relay identifiers (integers in the range 0..**n**-1). The pair (u,v) designates the cable that interconnects the relays u and v. The pairs may occur in any order. Except the (u,v) pairs, which do not contain white spaces, white spaces can occur freely in input. Input data terminate with an end of file and are correct.

Output

For each data set, prints from the beginning of a line, the safety factor of the encoded net.

Example

Input:

```
5
0 0
1 0
3 3 (0,1) (0,2) (1,2)
2 0
5 7 (0,1) (0,2) (1,3) (1,2) (1,4) (2,3) (3,4)
```

Output:

```
0
```

1
3
0
2

Added by: Thanh-Vy Hua
Date: 2005-02-27
Time limit: 5s
Source limit:50000B
Languages: All
Resource: ACM South Eastern European Region 2004

SPOJ Problem Set (classical)

301. Booklets

Problem code: BOOK

Bob has a difficult job. He must distribute advertising booklets for extra school activities in different schools. The booklets have different number of pages. Bob has a list with the number of pages of each booklet and the number of schools that he must visit. He has to distribute the booklets such that each school gets a number of booklets equal to either the lower integer part (LIP),or the upper integer part (UIP) of the number of booklets divided by the number of schools. Poor Bob must obey other rules too. He must distribute all the **UIP** number of booklets first and then the **LIP** number of booklets.

Any booklet **A** that is distributed to a school **S_i** must have fewer or at most an equal number of pages that any other booklet B that is distributed to a school **S_j**, if **S_i** gets the booklets before **S_j** (i.e if **i < j** then **pages(A) <= pages(B)**). When Bob distributes the booklets to a school he must distribute them in the same relative order in which they are on his list.

Moreover, he must distribute them very fast. When he comes back to the advertising company his boss verifies if he accomplished well his task, by asking him the number of pages of the first booklet distributed to a specific school, following the order in which Bob visited the schools (starting with 0). Difficult job, isn't it? Can you help him?

Input

The input starts with a line containing a single integer **t <=** 20, the number of test cases. **t** test cases follow.

Each data set in the input stands for a particular set of bookets. For each set of booklets the input contains the number of schools, the school specified by Bob's boss, the number of booklets (**less than 3000**), the number of pages of each booklet (fits in integer). White spaces can occur freely between the numbers in the input. The input data are correct.

Output

For each set of data the program prints the result to the standard output on a separate line. The solution is represented by the number of pages of the first booklet distributed to the specified school.

Example

Input:

```
1
3
2
7
3 5 9 1 11 14 2
```

Output:

```
11
```

Added by: Thanh-Vy Hua
Date: 2005-02-27
Time limit: 5s
Source limit:50000B
Languages: All
Resource: ACM South Eastern European Region 2004

SPOJ Problem Set (classical)

302. Count on Cantor

Problem code: CANTON

One of the famous proofs of modern mathematics is Georg Cantor's demonstration that the set of rational numbers is enumerable. The proof works by using an explicit enumeration of rational numbers as shown in the diagram below.

```
1/1 1/2 1/3 1/4 1/5 ...
2/1 2/2 2/3 2/4
3/1 3/2 3/3
4/1 4/2
5/1
```

In the above diagram, the first term is 1/1, the second term is 1/2, the third term is 2/1, the fourth term is 3/1, the fifth term is 2/2, and so on.

Input

The input starts with a line containing a single integer **t** <= 20, the number of test cases. **t** test cases follow.

Then, it contains a single number per line.

Output

You are to write a program that will read a list of numbers in the range from 1 to 10^7 and will print for each number the corresponding term in Cantor's enumeration as given below.

Example

Input:

```
3
3
14
7
```

Output:

```
TERM 3 IS 2/1
TERM 14 IS 2/4
TERM 7 IS 1/4
```

Added by: Thanh-Vy Hua
Date: 2005-02-27
Time limit: 5s
Source limit:50000B
Languages: All
Resource: ACM South Eastern European Region 2004

SPOJ Problem Set (classical)

303. The Unstable Cube

Problem code: UCUBE

A large cube (of size NxNxN) is given. At the beginning it consists of small blocks (1x1x1) and each block is painted in some color (different blocks may have the same color). But in the process of exploitation some blocks have disappeared. Given 6 photos of the unstable cube you have to calculate the maximum possible number of blocks that still remain in the unstable cube. It is possible that the unstable cube consists of more than one part.

Input

t - the number of test cases, then *t* test cases follow.
N - size of the big cube *[1 <= N <= 10]*
In the next *N* lines views of the cube from 6 sides are described (in the following order: from the front, left, back, right, from above, from below). Each such view is represented by a table of size *NxN* in which different letters denote different colors, and the symbol "." (point) means that it is possible to see all the way through the cube at this point. Consecutive views are separated by exactly one space.

The bottom border of the top view corresponds to the top border of the front view, and the top border of the bottom view - to the bottom border of the front view. For the front, back, left and right views the top and bottom sides of a view correspond to the top and bottom of the cube.

The input file is correct, i.e. each test case describes a possible configuration.

Output

For each test case output one integer: the required maximum number of blocks remaining in the unstable cube.

Example

Input:
```
2
3
.R. YYR .Y. RYY .Y. .R.
GRB YGR BYG RBY GYB GRB
.R. YRR .Y. RRY .R. .Y.
2
ZZ ZZ ZZ ZZ ZZ ZZ
ZZ ZZ ZZ ZZ ZZ ZZ
```

Output:
```
11
8
```

Added by: Roman Sol
Date: 2005-03-01
Time limit: 1s
Source limit: 20000B
Languages: All
Resource: The Moscow Olympiad on computer science 2004/05. Correspondence round.

SPOJ Problem Set (classical)

309. The Room Pattern

Problem code: RATTERN

It was decided to make a parquet floor in a room of size NxM. The idea is to lay out some pattern on the floor. The parquet tiles with which the floor of the room looks best consist of squares 1x1, each of which can be either white or black. The required color of each square of the room is specified on the map of the room.

There are four different forms of parquet tiles:
Illustration of parquet tiles
Squares of one parquet tile can be painted differently. Some types of tiles can be of identical shape, but painted differently. Tiles of different types can have different cost. The number of available tiles of each type is not limited. Tiles are allowed to be turned around somehow (by an angle which is a multiple of 90 degrees), but it is not permitted to break a tile or to put it face sheet downwards. Initially, any part of the floor can be already laid out by tiles. You are requested to calculate the minimal cost of the tiles necessary to pave the remaining part of the room.

Input

t - the number of test cases, then *t* test cases follow.
In the first line of each test case three numbers are written: *N, M* (the sizes of the room) and *K* (number of accessible types of tiles). *[1<= N, M <= 8], [1 <= K <= 10]*. Next there is a description of the desired painting of the floor. The description is given in the form of *N* lines of *M* numbers each, where 0 denotes the color white, 1 - the color black, 2 - a square which has already been covered by a tile. In the last K lines the descriptions of available types of tiles are given in the following format:
[Form] [cost] [painting] where:
[Form] is a number from 1 to 4, describing the form of a tile (see figure above)
[Cost] is an integer not larger than 10000, describing the cost of one tile of the type.
[Painting] is a sequence of between one and three numbers 0 or 1. Its length is the same as the number of squares of which the tile consists, and the respective numbers describe colors of square tiles in the order in which the squares are numbered in the figure.

Output

For each test case output one integer: the minimal cost of laying the remaining part of the parquet, or -1 if the task cannot be performed.

Example

Input:

```
1
4 3 3
2 2 2
2 0 0
2 1 2
2 2 2
2 10 0 0
1 5 1
4 6 0 0 1
```

Output:

```
15
```

Added by: Roman Sol
Date: 2005-03-05
Time limit: 17s
Source limit: 20000B
Languages: All
Resource: The Moscow Olympiad on computer science 2004/05. Correspondence round.

SPOJ Problem Set (classical)

318. Pythagorean Legacy

Problem code: PITPAIR

It is necessary to find a minimal integer value R which is equal to the length of the hypotenuse (the side opposite the right angle) of N non-identical rectangular triangles with integer lengths of sides.

Input

t - number of test cases [t <= 100], than t lines follow, each line contains one integer - N, equal to the required number of different rectangular triangles. [1 <= N <= 2000]

Output

For each test case your program should output a number R in a separate line (R fits in a 64-bit integer), equal to the minimal integer value of a hypotenuse for which exactly N different rectangular triangles can be constructed; then in separate lines follow exactly N numbers equal to the shorter cathetus (side adjacent to the right angle) of each of the rectangular triangles, in ascending order.

Example

Input:
```
2
1
2
```

Output:
```
5
3
25
7
15
```

Added by: Roman Sol
Date: 2005-03-01
Time limit: 9s
Source limit:8192B
Languages: All
Resource: ZCon 2005

SPOJ Problem Set (set9)

325. The Tall Windmills

Problem code: WINDMILL

In the later days of his career Johnny purchased a long and narrow strip of land on which he intended to erect a row of windmills, and live off the electrical energy produced by his little power plant. To his dismay, he soon discovered that he had been badly cheated - throughout most of the year the wind blew lengthwise through the strip, rather than in a perpendicular direction. As a result, the wind was certain to lose most of its force on the first windmill it encountered, leaving all the others idle. Johnny could only see one way of coping with this problem, namely - to vary the height of windmills situated relatively close to each other. More precisely, Johnny intends to build exactly n windmills along a straight line, with equal spacing (of one Bytelandian furlong) between adjacent windmills. It has been established by a team of experts that if two windmills are k Bytelandian furlongs apart from each other, their height must differ by at least $n-k$ Bytelandian yards. No windmill may ever be lower than 1 Bytelandian yard, and some, obviously, may need to be considerably higher. But tall windmills are far more expensive to construct, and thus you have been asked to choose the heights of Johnny's windmills in such a way as to guarantee that the tallest windmill has the minimum possible height.

Input

Input starts with a single integer t, the number of test cases (t<=100). t test cases follow.

Each test case consists of exactly one integer n (1<=n<=100) - the number of windmills Johnny intends to construct.

Output

For each test case output a line with exactly n numbers, denoting the heights of successive windmills given in the order in which they are arranged along the road.

Example

Input:

```
3
1
2
3
```

Output:

```
1
1 2
2 4 1
```

Added by: Adrian Kosowski
Date: 2005-04-13
Time limit: 17s
Source limit: 50000B
Languages: All except: C99 strict
Resource: DASM Programming League 2004, problemset 9

SPOJ Problem Set (classical)

327. Platon and Socrates

Problem code: PLATON

Platon and Socrates one day decided to play a new game. They asked their friend to think of two numbers between 1 and 5000, not equal. Then they asked him to tell the product to Platon and the sum to Socrates. After that they tried to figure out what these numbers are. They played a lot of times but none of them could guess these numbers. Finally they made it!! Here is the dialogue:

- [P] : I don't know the answer.
- [S] : I knew you wouldn't know, I don't know it either.
- [P] : Now I know it.
- [S] : I know it too.

Your task is to find all pairs Platon and Socrates could have been thinking about. Numbers are limited to the given range.

Input

Input starts with a single integer t, the number of test cases (t<=2000). t test cases follow. Each test case consists of one line containing two integers *l r* seperated by a single space, denoting the range of numbers ($1 <= l < r <= 5000$, $r-l < 200$).

Output

For the i-th test case output a line with the text *case i*. In the next line print *n* - number of pairs from range *(l,r)*. Then exactly *n* lines follow with two numbers seperated by single space. The first number is not greater than the second. Pairs are printed in increasing sum order.

Example

Input

```
2
1 10
2 8
```

Output

```
case 1
0
case 2
0
```

Added by: Bogusław K. Osuch
Date: 2005-04-14
Time limit: 15s
Source limit:50000B
Languages: All except: C99 strict
Resource: :P

SPOJ Problem Set (classical)

328. Bishops

Problem code: BISHOPS

Yesterday was Sam’s birthday. The most interesting gift was definitely the chessboard. Sam quickly learned the rules of chess and defeated his father, all his friends, his little sister, and now no one wants to play with him any more.
So he decided to play with another birthday gift - a Book of Math Problems for Young Mathematicians. He opened the book somewhere in the middle and read the following problem: "How many knights can be placed on a chessboard without threatening each other?" After a while he realized that this was trivial and moved on to the next problem: "How many bishops can be placed on a chessboard without threatening each other?". Sam is in trouble here. He is not able to solve this problem and needs your help.
Sam’s chessboard has size N *x* N. A bishop can move to any distance in any of the four diagonal directions. A bishop threatens another bishop if it can move to the other bishop’s position. Your task is to compute the maximum number of bishops that can be placed on a chessboard in such a way that no two bishops threaten each other.

Input

The input file consists of several lines. The line number i contains a single number N representing the size of the i-th chessboard. [N <= 10^100]

Output

The output file should contain the same number of lines as the input file. The i-th line should contain one number - the maximum number of bishops that can be placed on i-th chessboard without threatening each other.

Example

Input:
```
2
3
```

Output:
```
2
4
```

Added by: Roman Sol
Date: 2005-04-17
Time limit: 1s
Source limit: 10000B
Languages: All
Resource: IPSC 2004

SPOJ Problem Set (classical)

329. Calls

Problem code: CALLS

A young archeologist Senoj Anaidni recently made a very important discovery which will make him famous (or at least he thinks so). He found several scraps of paper resembling advertisement flyers of an ancient phone company. His research showed that modern phone companies follow a few basic rules to compose their flyers (and there is no reason to assume that old companies were an exception).
Each company operates certain number of phone lines. Each phone line connects a pair of cities, and it can be used in both directions. The cost of using each line is a fixed positive number. A call from city A to city B may be routed through one or more other cities, in which case the cost of the call is the sum of the costs of all lines used. (In fact, sometimes it is cheaper to route the call through several other cities than to use the direct connection, even if there exists one.)
To make the information comprehensible to the customer, the phone company lists the cost of the cheapest possible call between every pair of cities serviced by the company. To impress the customer even more, the company also lists the number of lines it operates.
Indeed, each of Senoj's ancient flyers start like this: "Using our 47 telephone lines, we serve 10 most important cities of the world! A call from Sparta to Troja costs 12 dennario, Sparta to Athens is 15 dennario, ...". The list of all pairs of cities and the respective costs of the cheapest possible call between them follows.
This supports Senoj's hypothesis about the origin of the papers, but he is not sure whether they are really genuine. Other archeologists often play dirty jokes on him by making ridiculous forgeries in a hope, that he will make a fool of himself. Luckily, they are often not very meticulous, so we can safely assume, that a flyer is a forgery if and only if it could not have been published by any phone company.

Input

The first line of the input file gives the number *t* of flyers found by Senoj. [$t <= 50$] Each flyer is described in a separate block starting with a line containing two integers - N and K - where N is the number of cities and K is the number of phone lines. [N <= 300 K <= 1200] The block continues with N-1 lines giving the costs of the cheapest calls between all pairs of cities. In particular, the i-th line contains (N-i) numbers, where j-th number represents the cost of a call between the cities i and (i+j).

Output

For every input block, output a line containing either "YES" or "NO". "YES" should be printed, if it is possible to assign costs to the phone lines operated by the company so that the cheapest calls are as advertised in the flyer. "NO" should be printed if this is not possible.

Example

Input:

```
2
3 3
1 2
2
3 2
```

```
1 2
2
```

Output:

```
YES
NO
```

Added by:	Roman Sol
Date:	2005-04-17
Time limit:	3s
Source limit:	10000B
Languages:	All
Resource:	IPSC 2004

SPOJ Problem Set (classical)

332. Hard Question

Problem code: HARDQ

Students of computer science in Bratislava enjoy hiking and camping during their long summer breaks. They love walking silently in the groves, visiting sparkling waterfalls, exploring dark caves, climbing steep hills, or just sleeping in a tent. Some of them already visited all the national parks in Slovakia and nearby countries.
With no more new national parks to visit, frustrated students decided to set up a new national park (NP) by themselves. After long arguing, they finally agreed on the boundary of the NP. Now they want to purchase all the land needed for NP from present owners. Their funds are limited (after all, they are only students), therefore they do not want to buy any land outside the NP.
The NP can be described as a polygon with N vertices. There is a set P of M rectangular plots of land available for sale by their owners. The rectangles are mutually disjoint and axis-parallel. Your task is to decide whether it is possible to purchase subset of plots P exactly covering the proposed NP.

Input

Input file consists of several test cases separated by a blank line. Each test case starts with two integers N and M. Next N lines contain the coordinates of the vertices of the NP. Each of the following M lines describes one plot. For each plot, the coordinates of two opposite corners of the rectangle are given. The values N=0, M=0 end the input and should not be processed. [N, M <= 3000]

Output

For each test case output either ’YES’ or ’NO’ depending on whether it is possible to set up the NP using P or not.

Example

Input:

```
4 2
0 0
0 2
2 2
2 0
1 0 0 2
1 0 2 2

3 1
0 0
2 2
2 0
0 0 1 1

0 0
```

Output:

```
YES
NO
```

Added by: Roman Sol
Date: 2005-04-17
Time limit: 5s
Source limit: 30000B
Languages: All
Resource: IPSC 2004

SPOJ Problem Set (classical)

334. The Philosophical Dispute

Problem code: PHDISP

One day, mathematician and philosopher were engaged in a heated dispute.
Philosopher said:
- Ideal line has only length and no width, therefore, no line can have an area.
Mathematician replied:
- That's as it may be, but still you can ll a square with a line in such a way that there will be no gaps. And you can't deny that a square has an area, and he grinned.
But Philosopher still wasn't convinced:
- Show me this line, then.
- With pleasure... - responded Mathematician and scribbled some equations on a piece of paper:
[IMAGE]
- With t increasing, the point (x, y) will move around the square, forming a line.
- So what? - asked Philosopher. How is it going to ll the entire square?
- Indeed, it will, - said Mathematician, - Whichever point inside the square you draw, the line will eventually cross that point.
- No, - replied Philosopher indignantly, - Anyway, I don't believe. When will the line cross this point? - and he put a thick dot inside the square.
Give Philosopher an answer.

Input

t - number of tests [*t* <= 150], than t test cases follows.
The first line of each test case contains the coordinates (x0, y0) of the dot center (-1 <= x0, y0 <= 1).
The second line contains eps <= 0.0001 - the radius of the dot (the dot is essentially a small circle).

Output

For each test case output any value of t in the segment [0, 10^12], which corresponds to the line crossing the dot, or "FAIL", if the line doesn't cross the dot.

Example

Sample input:
```
1
0.744 0.554
0.01
```

Sample output:
```
5.3
```

Added by:	Roman Sol
Date:	2005-04-25
Time limit:	3s
Source limit:	20000B
Languages:	All
Resource:	IX Ural Championship (Round II)

SPOJ Problem Set (classical)

336. Exchange Operations

Problem code: EOPERA

Given a sequence of 12 numbers consisting of 0 and the first 11 natural numbers. Suppose number 0 is in the i-th position of the sequence (positions are numbered from 0 to 11). You can swap it with the number in the j-th position if the following conditions hold:

- $| i - j | = d_k$, where k=1..3 and $(d_1,d_2,d_3,d_4)=(1;3;6;12)$
- $\text{floor}(i/d_{k+1})=\text{floor}(j/d_{k+1})$

Your task is to find the minimum number of exchange operations required to sort the sequence in increasing order.

Input

The first line of the input file contains an integer representing the number of test cases to follow. Each test case contains a sequence of twelve numbers consisting of 0,1,2,..,11, separated by single space. You can assume that the given sequence can always be sorted in increasing order by using the exchange operations

Output

For each test case, output the minimum number of exchange operations required to sort the given sequence in increasing order.

Example

Input:

```
2
1 10 2 3 0 5 7 4 8 6 9 11
6 4 1 0 3 5 9 7 2 10 11 8
```

Output:

```
8
9
```

Added by: Walrus
Date: 2005-04-28
Time limit: 19s
Source limit:50000B
Languages: All
Resource: Based on a problem from acm.uva.es

SPOJ Problem Set (classical)

339. Recursive Sequence

Problem code: SEQ

Sequence (a_i) of natural numbers is defined as follows:
$a_i = b_i$ (for $i <= k$)
$a_i = c_1 a_{i-1} + c_2 a_{i-2} + ... + c_k a_{i-k}$ (for $i > k$)
where b_j and c_j are given natural numbers for $1<=j<=k$. Your task is to compute a_n for given n and output it modulo 10^9.

Input

On the first row there is the number C of test cases (equal to about 50).
Each test contains four lines:
k - number of elements of (c) and (b) ($1 <= k <= 10$)
$b_1,...,b_k$ - k natural numbers where $0 <= b_j <= 10^9$ separated by spaces $c_1,...,c_k$ - k natural numbers where $0 <= c_j <= 10^9$ separated by spaces
n - natural number ($1 <= n <= 10^9$)

Output

Exactly C lines, one for each test case: a_n modulo 10^9

Example

Input:
```
3
3
5 8 2
32 54 6
2
3
1 2 3
4 5 6
6
3
24 354 6
56 57 465
98765432
```

Output:
```
8
714
257599514
```

Added by:	Paweł Dobrzycki
Date:	2005-04-29
Time limit:	2s
Source limit:	8196B
Languages:	All
Resource:	IV Podlasian Contest in Team Programming

SPOJ Problem Set (classical)

344. Poker

Problem code: POKER

In poker, you have 5 cards. There are 10 kinds of poker hands (from highest to lowest):

- royal flush - ace, king, queen, jack and ten, all in the same suit
- straight flush - five cards of the same suit in sequence, such as 10,9,8,7,6 of clubs; ace can be counted both as the highest card or as the lowest card - A,2,3,4,5 of hearts is a straight flush. But 4,3,2,A,K of hearts is not a straight flush - it's just a flush.
- four of a kind - four cards of the same rank, such as four kings.
- full house - three cards of one rank plus two cards of another rank
- flush - five cards of the same suit (but not a straight flush)
- straight - five cards in order - just like the straight flush, but mixed suits
- three of a kind - three cards of one rank and two other cards
- two pairs - two cards of one rank, two cards of another rank, and one more card
- pair - two cards of the same rank
- high card - none of the above

Write a program that will help you play poker by telling you what kind of hand you have.

Input

The first line of input contains the number of test cases (no more than 20). Each test case consists of one line - five space separated cards. Each card is represented by a two-letter (or digit) word. The first character is the rank (A,K,Q,J,T,9,8,7,6,5,4,3 or 2), the second character is the suit (S,H,D,C standing for spades, hearts, diamonds and clubs). The cards can be in any order (but they will not repeat).

Output

For each test case output one line describing the type of a hand, exactly like in the list above.

Example

Input:

```
3
AH KH QH TH JH
KH 5S 3C 5C 7D
QH QD 2S QC 2C
```

Output:

```
royal flush
pair
full house
```

Added by: Tomek Czajka
Date: 2005-05-03
Time limit: 7s
Source limit: 50000B
Languages: All
Resource: Purdue Programming Contest Training

SPOJ Problem Set (classical)

345. Mixtures

Problem code: MIXTURES

Harry Potter has n mixtures in front of him, arranged in a row. Each mixture has one of 100 different colors (colors have numbers from 0 to 99).

He wants to mix all these mixtures together. At each step, he is going to take two mixtures that stand next to each other and mix them together, and put the resulting mixture in their place.

When mixing two mixtures of colors a and b, the resulting mixture will have the color (a+b) mod 100.

Also, there will be some smoke in the process. The amount of smoke generated when mixing two mixtures of colors a and b is a*b.

Find out what is the minimum amount of smoke that Harry can get when mixing all the mixtures together.

Input

There will be a number of test cases in the input.

The first line of each test case will contain n, the number of mixtures, 1 <= n <= 100.

The second line will contain n integers between 0 and 99 - the initial colors of the mixtures.

Output

For each test case, output the minimum amount of smoke.

Example

Input:
```
2
18 19
3
40 60 20
```

Output:
```
342
2400
```

In the second test case, there are two possibilities:

- first mix 40 and 60 (smoke: 2400), getting 0, then mix 0 and 20 (smoke: 0); total amount of smoke is 2400
- first mix 60 and 20 (smoke: 1200), getting 80, then mix 40 and 80 (smoke: 3200); total amount of smoke is 4400

The first scenario is a much better way to proceed.

Added by: Tomek Czajka
Date: 2005-05-03
Time limit: 9s
Source limit: 50000B
Languages: All
Resource: Purdue Programming Contest Training

SPOJ Problem Set (classical)

346. Bytelandian gold coins

Problem code: COINS

In Byteland they have a very strange monetary system.

Each Bytelandian gold coin has an integer number written on it. A coin n can be exchanged in a bank into three coins: n/2, n/3 and n/4. But these numbers are all rounded down (the banks have to make a profit).

You can also sell Bytelandian coins for American dollars. The exchange rate is 1:1. But you can not buy Bytelandian coins.

You have one gold coin. What is the maximum amount of American dollars you can get for it?

Input

The input will contain several test cases (not more than 10). Each testcase is a single line with a number n, 0 <= n <= 1 000 000 000. It is the number written on your coin.

Output

For each test case output a single line, containing the maximum amount of American dollars you can make.

Example

Input:
```
12
2
```

Output:
```
13
2
```

You can change 12 into 6, 4 and 3, and then change these into $6+$4+$3 = $13. If you try changing the coin 2 into 3 smaller coins, you will get 1, 0 and 0, and later you can get no more than $1 out of them. It is better just to change the 2 coin directly into $2.

Added by: Tomek Czajka
Date: 2005-05-03
Time limit: 9s
Source limit:50000B
Languages: All
Resource: Purdue Programming Contest Training

SPOJ Problem Set (classical)

348. Expedition

Problem code: EXPEDI

A group of cows grabbed a truck and ventured on an expedition deep into the jungle. Being rather poor drivers, the cows unfortunately managed to run over a rock and puncture the truck's fuel tank. The truck now leaks one unit of fuel every unit of distance it travels.

To repair the truck, the cows need to drive to the nearest town (no more than 1,000,000 units distant) down a long, winding road. On this road, between the town and the current location of the truck, there are N (1 <= N <= 10,000) fuel stops where the cows can stop to acquire additional fuel (1..100 units at each stop).

The jungle is a dangerous place for humans and is especially dangerous for cows. Therefore, the cows want to make the minimum possible number of stops for fuel on the way to the town. Fortunately, the capacity of the fuel tank on their truck is so large that there is effectively no limit to the amount of fuel it can hold. The truck is currently L units away from the town and has P units of fuel (1 <= P <= 1,000,000).

Determine the minimum number of stops needed to reach the town, or if the cows cannot reach the town at all.

Input

The first line of the input contains an integer t representing the number of test cases. Then t test cases follow. Each test case has the follwing form:

- Line 1: A single integer, N
- Lines 2..N+1: Each line contains two space-separated integers describing a fuel stop: The first integer is the distance from the town to the stop; the second is the amount of fuel available at that stop.
- Line N+2: Two space-separated integers, L and P

Output

For each test case, output a single integer giving the minimum number of fuel stops necessary to reach the town. If it is not possible to reach the town, output -1.

Example

Input:

```
1
4
4 4
5 2
11 5
15 10
25 10
```

Output:
```
2
```

Input details
```
The truck is 25 units away from the town; the truck has 10 units
of fuel.  Along the road, there are 4 fuel stops at distances 4,
5, 11, and 15 from the town (so these are initially at distances
21, 20, 14, and 10 from the truck).  These fuel stops can supply
up to 4, 2, 5, and 10 units of fuel, respectively.
```

Output details:
```
Drive 10 units, stop to acquire 10 more units of fuel, drive 4 more
units, stop to acquire 5 more units of fuel, then drive to the town.
```

Added by: Walrus
Date: 2005-05-03
Time limit: 15s
Source limit: 50000B
Languages: All
Resource: US Open International 2005 Gold Division

SPOJ Problem Set (classical)

349. Around the world

Problem code: AROUND

Over the years, FJ has made a huge number of farmer friends all around the world. Since he hasn't visited 'Farmer Ted' from England and 'Boer Harms' from Holland for a while, he'd like to visit them.

He knows the longitude of the farm where each of his worldwide friends resides. This longitude is an angle (an integer in the range 0..359) describing the farm's location on the Earth, which we will consider to be a circle instead of the more complex and traditional spherical representation. Except for the obvious discontinuity, longitudes increase when traveling clockwise on this circle.

FJ plans to travel by airplane to visit his N (1 <= N <= 5,000) friends (whose farms are uniquely numbered 1..N). He knows the schedules for M (1 <= M <= 25,000) bidirectional flights connecting the different farms. Airplanes always travel shortest paths on the Earth's surface (i.e., on the shortest arc of a circle).

There will always be a unique shortest path between two farms that are directly connected. No pair of antipodal farms (exactly opposite each other on the circle) is ever directly connected.

Each airplane flight can be described as traveling in clockwise or counterclockwise direction around the Earth's surface. For example, a flight from longitude 30 to longitude 35 would be clockwise, as would be a flight from longitude 350 to longitude 10. However, a flight from longitude 350 to longitude 200 follows a shortest path counterclockwise around the circle.

FJ would find it very cool if he could make a trip around the world, visiting some of his friends along the way. He'd like to know if this is possible and if so, what is the minimum number of flights he can take to do so.

He wants to start and finish his journey at the location of his best friend (the one listed first in the input below). In order to make sure he actually circles the Earth, he wants to ensure that the clockwise distance he travels is different from the counterclockwise distance he travels.

Input

The first line of the input contains an integer t representing the number of test cases. Then t test cases follow. Each test case has the following form:

- Line 1: Two space-separated integers: N and M
- Lines 2..N+1: Line i+1 contains one integer: the longitude of the i-th farm. Line 2 contains the location of the farm of his best friend.
- Lines N+2..N+M+1: Line i+N+1 contains two integers giving the indices of two farms that are connected by a flight.

Output

For each test case, output a single integer specifying the minimum number of flights FJ needs to visit to make a trip around the world. Every time FJ moves from one farm to another counts as one flight. If it is impossible to make such a trip, output the integer -1.

Example

Input:

```
1
3 3
0
120
240
1 2
2 3
1 3
```

Output:

```
3
```

Input details

```
Farmer John has three friends at longitudes 0, 120, and 240.  There are
three flights: 0<->120, 120<->240, and 0<->240.  The journey must start and
finish at longitude 0.
```

Output details

```
FJ must visit all 3 friends to make a full trip around the world.
```

Added by: Walrus
Date: 2005-05-03
Time limit: 9s
Source limit: 50000B
Languages: All
Resource: US Open International 2005 Gold Division

SPOJ Problem Set (classical)

350. Landscaping

Problem code: LANDSCAP

Farmer John is making the difficult transition from raising mountain goats to raising cows. His farm, while ideal for mountain goats, is far too mountainous for cattle and thus needs to be flattened out a bit. Since flattening is an expensive operation, he wants to remove the smallest amount of earth possible.

The farm is long and narrow and is described in a sort of two-dimensional profile by a single array of N (1 <= N <= 1000) integer elevations (range 1..1,000,000) like this:

1 2 3 3 3 2 1 3 2 2 1 2,

which represents the farm's elevations in profile, depicted below with asterisks indicating the heights:

```
    * * *     *
  * * * * *   * * *   *
* * * * * * * * * * * *
1 2 3 3 3 2 1 3 2 2 1 2
```

A contiguous range of one or more equal elevations in this array is a "peak" if both the left and right hand sides of the range are either the boundary of the array or an element that is lower in elevation than the peak. The example above has three peaks.

Determine the minimum volume of earth (each unit elevation reduction counts as one unit of volume) that must be removed so that the resulting landscape has no more than K (1 <= K <= 25) peaks. Note well that elevations can be reduced but can never be increased.

If the example above is to be reduced to 1 peak, the optimal solution is to remove 2 + 1 + 1 + 1 = 5 units of earth to obtain this set of elevations:

```
    * * *     -
  * * * * *   - - -   -
* * * * * * * * * * * *
1 2 3 3 3 2 1 1 1 1 1 1
```

where '-'s indicate removed earth.

Input

The first line of the input contains integer t representing the number of test cases. Then t test cases follow. Each test case has the following form:

- Line 1: Two space-separated integers: N and K
- Lines 2..N+1: Each line contains a single integer elevation. Line i+1 contains the elevation for index i.

Output

For each test case, output the minimum volume of earth that must be removed to reduce the number of peaks to K.

Example

Input:

```
1
12 1
1
2
3
3
3
2
1
3
2
2
1
2
```

Output:

```
5
```

Input details

```
This is the example used above.
```

Added by: Walrus
Date: 2005-05-03
Time limit: 9s
Source limit:50000B
Languages: All
Resource: US Open International 2005 Gold Division

SPOJ Problem Set (classical)

351. Ha-noi!

Problem code: HAN01

Little Sabrina loves solving puzzles. Last week she got a new puzzle: The "Tower of Hanoi" puzzle. This puzzle is based on an old legend: "The temple priests of hanoi have to transfer a tower consisting of 64 fragile disks of gold from one part of the temple to another, one disk at a time. The disks are arranged in order, no two of them the same size, with the largest on the bottom and the smallest on top. Because of their fragility, a larger disk may never be placed on a smaller one, and there is only one intermediate location where disks can be temporarily placed. It is said that before the priests complete their task the temple will crumble into dust and the world will vanish in a clap of thunder." Sabrina reconstructed the problem with some coins of different size. She solved the puzzle for three coins in 7 steps, for four coins in 15 steps,... after solving the problem with 7 coins she had the hang of it. Yesterday she started to solve the puzzle with 31 coins and her optimal strategy. After hours of moving coins from one pile to the other she was very tired and went to bed. This was a bad idea! Her little brother Robin discovered the towers of coins and - whoops! - threw it on the floor. Then he noticed a sheet of paper: "Don't touch this towers! Steps: 16543". "Oh no!" Robin has to reconstuct the tower because his sister can get very, very angry... Your task is to help Robin to reconstruct the towers. Sabrina started the game with all disks on peg number one and her goal was to move the disks to peg number two. She used her optimal strategy and noted the number of steps she had done.

Input

The first line of input contains one integer t: The number of testcases. t lines follow. Each line contains two integers n (2< n< 61) and k (0< k< 2^n). n is the number of disks of the hanoi puzzle and k the number of steps Sabrina had done.

Output

Output the reconstructed configuration of the towers after k steps. For each testcase output three lines. One for each tower. Each line consists of the tower identifier (1,2,3) a colon, one space and the disk numbers (n,n-1,...,2,1) which are seperated by a '|'-character.

Example

Input:

```
3
3 6
32 889397450
31 16543
```

Output:

```
1: 1
2: 3|2
3:
1: 32|31|28|25|18|17|14|3
2: 30|29|26|13|12|11|10|9|6|5|2
```

```
3: 27|24|23|22|21|20|19|16|15|8|7|4|1
1: 31|30|29|28|27|26|25|24|23|22|21|20|19|18|17|16|7|6
2: 15|8|5|4|3|2|1
3: 14|13|12|11|10|9
```

Added by: Simon Gog
Date: 2005-05-03
Time limit: 2s
Source limit: 8082B
Languages: All
Resource: Ulm Algorithm Course SoSe 2005

SPOJ Problem Set (classical)

359. Alpha Centauri Tennis

Problem code: ACT

As you may know, planets of Alpha Centauri (if they indeed do exist) would provide excellent conditions for intelligent life forms.
It is indeed true that there is a small Earthlike planet near Alpha Centauri, inhabited by a population of no particular significance. These humanlike creatures have much in common with us. Living in similar comunities and having similar body structure and behavioral patterns, they unsurprisingly appreciate (approximately) the same time-killing activities as we do. One of these, the second most popular after Alpha Centauri Croquet, is the Alpha Centauri Tennis.
Although its rules differ from Earth Tennis, the two player version of Alpha Centauri Tennis resembles it in many ways. Same as Earth Tennis, it is played on a rectangular court divided into two parts by a net. Two players, standing on opposite sides of it, use a stringed racket to hit a ball back and forth to each other. There are certain rules how to hit the ball. The player who forces his opponent to violate one of these rules wins the current ball. The aim of both players is to win enough balls to win a game, enough games to win a set and enough sets to win the whole match. In the N player version of the Alpha Centauri Tennis a ball can be won by any one of the N players. Although technical details of this can be difficult to imagine, Alpha Centaurians are extremely inventive.
In the general N-player version, players serve in turns, following order determined before the match. Moreover, they shift when starting individual games and sets. For example, the players are A, B and C. They are ordered alphabetically. Player A serves the first ball of the first game. When the ball is won by one of the players, its B's turn to serve. After the game is won by one of the players, player B starts the second game. Finally, when the first set is won by someone, player B starts the second set. This repeats, always shifted by one player, until the match ends.
For three players the serving order looks as follows:
Set 1:
Game 1: A,B,C,A,B,C...
Game 2: B,C,A,....
Game 3: C,A,B,....
Game 4: A,B,C,....
...
Set 2:
Game 1: B,C,A,B,....
Game 2: C,A,B,....
Game 3: A,B,C,A,...
...
There are exact rules for counting the number of balls/games/sets won by a player.
RULES FOR WINNING A GAME
The state of a game can be described by assigning a non-negative number of points to each of the players. At the beginning of a game, the score of each player is zero.
Note: In Earth terminology, 0 points is called "love", 1 point is a "fifteen", 2 points is a "thirty", 3 points is a "forty" and 4 points is an "advantage". Be glad that you don't have to learn the Centaurian terminology :)
When a player P just won a ball, the new score is determined by using the first rule from the list that applies to the situation.

If P currently has 3 points and no other player has more than 2 points, P wins the current game.
If P currently has 4 points, he wins the game.
If any other player currently has 4 points, that player loses one point. P gains a point.

RULES FOR WINNING A SET

The set is won by the first player that at the same time:
won at least 6 games in this set
won at least 2 games more than any other player

RULES FOR WINNING A MATCH

The winner is the first player to win at least three sets. A set in which no other player won a game counts as two won sets.

Problem specification

An observer from the Intergalactic Tennis Federation was watching a tournament in Alpha Centauri Tennis. Being unable to understand Alpha Centaurian language, he only managed to write down the winner of each ball. Now, for each match, knowing the sequence in which the players were winning the balls, he would like to somehow determine its winner.

Input

t - the number of test cases [t <= 150] than t test cases follows, each corresponding to one match. Each line contains the number of players N [N <= 10] and a string S consisting of uppercase letters [2 <= S <= 50000]. The players are represented by the first N letters of the English alphabet. If the i-th letter of S is X, it means the player X won the i-th ball from the beggining of the match. You may assume that the match transcripts are correct and complete.
The order in which the players serve is the same as the order of their letters in the English alphabet.

Output

For each line, output a single character, being the letter of the player who won the corresponding match.

Example

Input:

```
1
3 BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB
```

Output:

```
B
```

(B has won two sets, each of them by winning 6 games, while A and C won none. Thus each of these sets counts as two and B has won the match.)

Added by: Roman Sol
Date: 2005-05-13
Time limit: 1s
Source limit: 10000B
Languages: All
Resource: IPSC 2005

SPOJ Problem Set (classical)

362. Ignore the Garbage

Problem code: IGARB

Fred works as an IT consultant in an insurance company. As they always had a large amount of customers waiting and arguing at the front desk, management decided to deploy a ticket machine. Each customer would get a ticket with a number and there will be fancy LCD display over each desk showing the number of the next person. Fred was appointed to get this new enhancement working.
Because Fred is lazy when it comes to manual labor and as an IT consultant he wouldn't lower himself to the level of some hardware technician (except when upgrading his own computer), he asked few technicians to install the displays and prepared himself just to plug in the ticket machine and try it out. Unfortunately (for Fred) the technicians, either inspired by Mr.Bean or because of their carelessness, installed the display upside-down.
Being a software guy, Fred decided that the hardware should not be tampered with after it is installed (except for the case if he would be able to get back the technicians to repair it, but they were already angry at him for his nagging). Then he noted that from time to time the display shows a correct number even when it is upside-down. And hey, the ticket machine is an embedded device and contains a small processor! It would be just a sin for an IT guy not to try to meddle with it and try running an own version of Linux. Now we just need to figure out which readable numbers will the display show.

Task specification

In the beginning the display shows the number 1 on its display. Each second the number shown is increased by 1. We see the display upside-down and thus not everything we see will make sense. Your task is to compute the K-th valid number we will see on the display. The digits the display uses are shown on the images below. An upside-down 1 still count as 1. The number we see may have leading zeroes - e.g. turning the number 600 upside down leads to a valid number.
[IMAGE] [IMAGE] [IMAGE] [IMAGE] [IMAGE] [IMAGE] [IMAGE] [IMAGE] [IMAGE] [IMAGE]

Input

t - the number of test cases [t <= 2200], than t test cses follows. Each test case consist of one integer Ki [0 < Ki <= 10^200].

Output

For each Ki from the input file, output the Ki-th number shown on the display (including the leading zeroes, if there are some).

Example

Input:
```
8
1
2
3
```

```
4
5
6
8
98
```

Output:

```
1
2
5
9
8
6
11
002
```

Added by:	Roman Sol
Date:	2005-05-15
Time limit:	3s
Source limit:	30000B
Languages:	All
Resource:	IPSC 2005

SPOJ Problem Set (classical)

364. Pocket Money

Problem code: LISA

Young people spend a lot of money on things like sweets, music CDs, mobile phones and so on. But most young girls/boys have one problem: Their pocket money is not enough for all these jolly things. Little Lisa Listig is one of these poor girls with a small pocket money budget. Last month her pocket money lasted only one week. So she decided to enter into negotiations with her father. Her father Tomm - a mathematician - had an incredibly ingenious idea: He wrote down some fancy digits with operators (+,*) in between them on a sheet of paper and allowed Lisa to insert brackets. Then he declared that the result of that arithmetic expression is Lisa's new pocket money. Now it's Lisa's task to maximize her pocket money. As her father was surprised what a huge sum of money Lisa got for her result, he decided to minimize the result of the expression for his son Manfred. Now it's your task to calculate the results obtained by Lisa and her father.

Input

The first line of input contains the number of testcases k (k< 5000). Each of the following k lines consists of an arithemtic expression. This expression consists of numbers (0-9) seperated by one of the two operators '*' and '+'. There are no spaces between the characters. Each line contains less than 100 characters.

Output

For each expression output the result obtained by Lisa and the result obtained by her father separated by one space. The results of the calculations are smaller than 2^{64}.

Example

Input:

```
1
1+2*3+4*5
```

Output:

```
105 27
```

Two possible expressions for the first testcase:

```
105 = (1+2)*(3+4)*5
27  = 1+2*3+4*5
```

Added by: Simon Gog
Date: 2005-05-17
Time limit: 32s
Source limit: 8082B
Languages: All
Resource: Ulm Algorithm Course SoSe 2005

SPOJ Problem Set (classical)

365. Phidias

Problem code: PHIDIAS

Famous ancient Greek sculptor Phidias is making preparations to build another marvelous monument. For this purpose he needs rectangular marble plates of sizes W1 x H1, W2 x H2, ..., WN x HN.
Recently, Phidias has received a large rectangular marble slab. He wants to cut the slab to obtain plates of the desired sizes. Any piece of marble (the slab or the plates cut from it) can be cut either horizontally or vertically into two rectangular plates with integral widths andheights, cutting completely through that piece. This is the only way to cut pieces and piecescannot be joined together. Since the marble has a pattern on it, the plates cannot be rotated: if Phidias cuts a plate of size A ? B then it cannot be used as a plate of size B ? A unless A = B. He can make zero or more plates of each desired size. A marble plate is wasted if it is not of any of the desired sizes after all cuts are completed. Phidias wonders how to cut the initial slab so that as little of it as possible will be wasted.
As an example, assume that in the figure below the width of the original slab is 21 and the height of the original slab is 11, and the desired plate sizes are 10 x 4, 6 x 2, 7 x 5, and 15 x 10. The minimum possible area wasted is 10, and the figure shows one sequence of cuts with total waste area of size 10.

[IMAGE]

Your task is to write a program that, given the size of the original slab and the desired plate sizes, calculates the minimum total area of the original slab that must be wasted.

Input

t - the number of test cases, then t test cases follow [t <= 20]. The first line of each test case contains two integers: first W, the width of the original slab, and then H, the height of the original slab. The second line contains one integer N: the number of desired plate sizes. The following N lines contain the desired plate sizes. Each of these lines contains two integers: first the width Wi and then the height Hi of that desired plate size (1 <= i <= N). [1 <= W <= 600, 1 <= H <= 600, 0 < N <= 200, 1 <= Wi <= W, and 1 <= Hi <= H.]

Output

For each test case output one line with a single integer: the minimum total area of the original slab that must be wasted.

Example

Input:

```
1
21 11
4
10 4
6 2
7 5
```

```
15 10
```

Output:

```
10
```

Added by:	Roman Sol
Date:	2005-05-20
Time limit:	21s
Source limit:	30000B
Languages:	All
Resource:	IOI 2004

SPOJ Problem Set (classical)

366. Farmer

Problem code: FARMER

A farmer has a set of fields, each of which is surrounded by cypress trees. Also, the farmer has a set of strips of land, each of which has a row of cypress trees. In both fields and strips, between every two consecutive cypress trees is a single olive tree. All of the farmer's cypress trees either surround a field or are in a strip and all of the farmer's olive trees are between two consecutive cypress trees in a field or in a strip.
One day the farmer became very ill and he felt that he was going to die. A few days before he passed away he called his eldest son and told him, "I give you any Q cypress trees of your choice and all the olive trees which are between any two consecutive cypress trees you have chosen." >From each field and from each strip the son can pick any combination of cypress trees. Since the eldest son loves olives he wants to pick the Q cypress trees which will allow him to inherit as many olive trees as possible.

[IMAGE]

In Figure 1, assume that the son is given Q=17 cypress trees. To maximize his olive inheritance he should choose all the cypress trees in Field 1 and Field 2, inheriting 17 olive trees.
You are to write a program which, given the information about the fields and the strips and the number of cypress trees the son can pick, determines the largest possible number of olive trees the son may inherit.

Input

t - the number of test cases [t <= 20], then t test cses follows. The first line of each test case contains first the integer Q: the number of cypress trees the son is to select; then the integer M, the number of fields; and then the integer K, the number of strips. The second line contains M integers N1, N2,... NM, : the numbers of cypress trees in fields. The third line contains K integers R1, R2,... RK: the numbers of cypress trees in strips.
In all test cases, 0 <= Q <= 150000, 0 <= M <= 2000, 0 <= K <= 2000, 3 <= N1 <= 150, 3 <= N2 <= 150,... 3 <= NM <=150, 2 <= R1 <= 150, 2 <= R2 <= 150,... 2 <= RK <= 150. The total number of cypress trees in the fields and strips is at least Q. Additionally, in 50% of the test cases, Q <= 1500.

Output

For each test case output ont integer: largest possible number of olive trees the son may inherit.

Example

Input:

```
1
17 3 3
13 4 8
4 8 6
```

Output:

```
17
```

Added by:	Roman Sol
Date:	2005-05-22
Time limit:	50s
Source limit:	30000B
Languages:	All
Resource:	IOI 2004

SPOJ Problem Set (classical)

367. Empodia

Problem code: EMPODIA

The ancient mathematician and philosopher Pythagoras believed that reality is mathematical in nature. Present-day biologists study properties of biosequences. A biosequence is a sequence of M integers, which
* contains each of the numbers 0,1,...,M-1,
* starts with 0 and ends with M-1, and
* has no two elements E,E+1 in adjacent positions in this order.
A subsequence consisting of adjacent elements of a biosequence is called a segment..
A segment of a biosequence is called a framed interval if it includes all integers whose values are between the value of the first element, which must be the smallest element in the segment, and the last element, which must be the largest and different from the first. A framed interval is called an empodio if it does not contain any shorter framed intervals.
As an example, consider the biosequence (0,3,5,4,6,2,1,7). The whole biosequence is a framed interval. However, it contains another framed interval (3,5,4,6) and therefore it is not an empodio. The framed interval (3,5,4,6) does not contain a shorter framed interval, so it is an empodio. Furthermore, it is the only empodio in that biosequence.
You are to write a program that, given a biosequence, finds all empodia (plural for empodio) in that biosequence.

Input

t - the number of test cases [t <= 20], then t test cses follows. The first line of each test case contains a single integer M: the number of integers in the input biosequence. The following M lines contain the integers of the biosequence in the order of the sequence. Each of these M lines contains a single integer. In one test case, 1000000 <= M <= 1100000. In all other test cases, 1 <= M <= 60000. Additionally, in 50% of the test cases, M <= 2600.

Output

The first line for each test case is to contain one integer H: the number of empodia in the input biosequence. The following H lines describe all empodia of the input biosequence in the order of appearance of the starting point in the biosequence. Each of these lines is to contain two integers A and B (in that order) separated by a space, where the Ath element of the input biosequence is the first element of the empodio and the Bth element of the input biosequence is the last element of the empodio.

Example

Input:
```
1
8
0
3
5
```

```
4
6
2
1
7
```

Output:

```
1
2 5
```

Added by: Roman Sol
Date: 2005-05-22
Time limit: 40s
Source limit: 50000B
Languages: All
Resource: IOI 2004

SPOJ Problem Set (classical)

368. Cobbled streets

Problem code: CSTREET

The municipal chronicals of an unbelievable lordly major town in a land far, far away tell the following story:
> Once upon a time the new crowned king Günther decided to visit all towns in his kingdom. The people of the unbelievable lordly major town expected that king Günther would like to see some of the most famous buildings in their town. For the lordly citizens it seemed neccessary that all streets in the town that the king would have to use had to be cobbled with stone. Unfortunately the unbelievable lordly major town had not much money at that time as they used most of their savings to erect the highest cathedral the world had ever seen.
> Roumours were afloat that the real reason for their thriftiness was not that the town treasury was empty but that many people believed that king Günther came to the throne by deceiving his father king Erwin and that in his youth he made a pact with the devil. But anyway, the citizens of the unbelievable lordly major town decided to pave only as much streets as were absolutely necessary to reach every major building.
> Can you help the citizens of the unbelievable lordly major town to find out which streets should be paved?
>It might be usefull to know that all major buildings are either at the end of a street or at an intersection. In addition to that you can assume that all buildings are connected by the given streets.

Input

t [number of testcases (1 <= t <= 100)]
> p [price to pave one furlong of street (positive integer)]
> n [number of main buildings in the town (1 <= n <= 1000)]
> m [number of streets in the town (1 <= m <= 300000)]
> a b c [street from building a to building b with length c (lengths are given in furlong and the buildings are numbered from 1 to n)]

Output

For each testcase output the price of the cheapest possibility to reach all main buildings in the city on paved streets. You can assume that the result will be smaller than 2^32.

Example

Input:
```
1
2
5
7
1 2 1
2 3 2
2 4 6
5 2 1
5 1 3
```

```
4 5 2
3 4 3
```

Output:

```
12
```

Added by: Simon Gog
Date: 2005-05-24
Time limit: 15s
Source limit: 32211B
Languages: All
Resource: Ulm Algorithm Course SoSe 2005

SPOJ Problem Set (classical)

369. Math I

Problem code: MATH1

You are given n integers a_1, a_2,..., a_n (0<=a_i<=n). The sum a_1+ a_2+...+ a_n does not exceeded n. Your task is to find n other integers x_1, x_2,..., x_n (note that x_i may be negative numbers) satisfying the following conditions:

- (x_i - x_{i+1} + a_{i+1} = 0) or (x_i - x_{i+1} + a_{i+1} = 1) for i=1..n-1
- (x_n - x_1 + a_1 = 0) or (x_n - x_1 + a_1 = 1)
- |x_1|+|x_2|+...+|x_n| is minimized

Input

The first line of the input file contains an integer t representing the number of test cases (t<=20). Then t test cases follow. Each test case has the following form:

- The first line contains n (1<=n<=1000)
- The second line contains n integers a_1, a_2,..., a_n separated by single spaces

Output

For each test case output a single value: the minimum value of |x_1|+|x_2|+...+|x_n|

Example

Input:
```
2
4
2 1 0 0
5
0 1 2 2 0
```

Output:
```
1
3
```

Output Details:
In the former case, the optimal solution is (x_1=0, x_2=0, x_3=0, x_4=-1)
In the latter case, the optimal solution is (x_1=-1, x_2=-1, x_3=0, x_4=1, x_5=0)

Added by: Walrus
Date: 2005-05-25
Time limit: 20s
Source limit: 50000B
Languages: All

SPOJ Problem Set (classical)

370. Ones and zeros

Problem code: ONEZERO

Certain positive integers have their decimal representation consisting only of ones and zeros, and having at least one digit one, e.g. 101. If a positive integer does not have such a property, one can try to multiply it by some positive integer to find out whether the product has this property.

Input

Number *K* of test cases (*K* is approximately 1000);
in each of the next *K* lines there is one integer *n* (*1 <= n <= 20000)*

Output

For each test case, your program should compute the smallest multiple of the number *n* consisting only of digits 1 and 0 (beginning with 1).

Example

Input:
```
3
17
11011
17
```
Output:
```
11101
11011
11101
```

Added by: Paweł Dobrzycki
Date: 2005-05-26
Time limit: 8s
Source limit:4096B
Languages: All
Resource: II Polish Olympiad in Informatics, Ist Stage

SPOJ Problem Set (classical)

372. The Benefactor

Problem code: BENEFACT

Another chapter of the municipal chronicles of a well known unbelievable lordly major town (if this town is not well known to you, you might want to solve problem CSTREET first) tells us the following story:
Once upon a time the citizens of the unbelievable lordly major town decided to elect a major. At that time this was a very new idea and election campaigns were completely unknown. But of course several citizens wanted to become major and it didn't took long for them to find out, that promising nice things that never will become real tends to be useful in such a situation. One candidate to be elected as a major was Ivo sometimes called the benefactor because of his valuable gifts to the unbelievably lordly major towns citizens.
One day before the election day Ivo the benefactor made a promise to the citizens of the town. In case of his victory in the elections he would ensure that on one of the paved streets of the town street lamps would be erected and that he would pay that with his own money. As thrifty as the citizens of the unbelievable lordly major town were, they elected him and one day after the elections they presented him their decision which street should have street lamps. Of course they chose not only the longest of all streets but renamed several streets so that a very long street in the town existed.
Can you find how long this street was? To be more specific, the situation is as follows. You are presented a list of all paved streets in the unbelievable lordly major town. As you might remember from problem CSTREET in the town the streets are paved in a way that between every two points of interest in the town exactly one paved connection exists. Your task is to find the longest distance that exists between any two places of interest in the city.

Input

The first line of input contains the number of testcases t.
The first line of each testcase contains the number of places (2 <=n<=50000) in the town. Each street is given at one line by two places (1<=a,b<=n) and the length of the street (0<=l<20000).

Output

For each testcase output one line which contains the maximum length of the longest street in the city.

Example

Input:

```
1
6
1 2 3
2 3 4
2 6 2
6 4 6
6 5 5
```

Output:

```
12
```

Added by: Simon Gog
Date: 2005-06-06
Time limit: 10s
Source limit: 10000B
Languages: All
Resource: Ulm Algorithm Course SoSe 2005

SPOJ Problem Set (classical)

373. Greedy island

Problem code: GREED

Gon is on Greedy island. He wants to go home. But to get the ticket to leave the game, he has to get N cards labeled in a sequence from 1 to N (the order of the cards in his hand is irrelevant). He already has N cards, but not forming a sequence from 1 to N. So he wants you to help him. For some cards, he can change one card for another for one piece of gold. Help him to get the ticket at the minimum cost (using the minimum number of exchanges).

Input

The first line contains t, the number of tests (1<=t<=10). For each test case:

- the number of cards N is given is given in the first line (2<=N<=500).
- the next N lines contain the N cards owned by Gon.
- the following line contains e, the number of different allowed types of exchanges.
- the next e lines contain two integers x_i,y_i each which mean that we can exchange and replace the card marked x by the card marked y and vice versa.

There is a blank line after each test case.

Output

For each test case, output a line denoting the minimum required cost.

Example

Input:

```
1
4
1
2
2
2
2
2 3
3 4
```

Output:

```
3
```

Added by: Le Trong Dao
Date: 2005-06-08
Time limit: 50s
Source limit:50000B
Languages: All
Resource: Mr.Tran Minh Quan

SPOJ Problem Set (classical)

374. Count maximum matrices

Problem code: MATRIX

You are given a matrix A of M rows and N columns, consisting of numbers 0 and 1. For a rectangle in A (sides >= 1), X1 is the number of ones on its sides, X0 is the number of zeros on its sides, and its *value* is defined as X1 - X0. Let us consider W, the maximum value taken over submatrices of A, and S, the number of submatrices with value W. Your task is to find W and S.

Input

The first line of input contains the number of testcases t (t <= 15). The first line of each testcase contains the numbers M, N (1 <= M, N <= 200) Then M lines follow. In each line, there are N numbers 0 or 1.

Output

For each testcase, you should output a single line with numbers W and S.

Example

Input:

```
1
5 6
1 1 1 1 1 1
1 0 0 0 0 1
1 0 0 0 0 1
1 0 0 0 0 1
1 1 1 1 1 1
```

Output:

```
18 1
```

Added by: Le Đôn Khue
Date: 2005-06-08
Time limit: 10s
Source limit:50000B
Languages: All

SPOJ Problem Set (classical)

375. Query on a tree

Problem code: QTREE

You are given a tree (an acyclic undirected connected graph) with **N** nodes, and edges numbered 1, 2, 3...**N**-1.

We will ask you to perfrom some instructions of the following form:

- **CHANGE i ti** : change the cost of the i-th edge to ti
  or
- **QUERY a b** : ask for the maximum edge cost on the path from node a to node b

Input

The first line of input contains an integer **t**, the number of test cases (**t** <= 20). t test cases follow.

For each test case:

- In the first line there is an integer **N** (**N** <= 10000),
- In the next **N**-1 lines, the i-th line describes the i-th edge: a line with three integers **a b c** denotes an edge between **a**, **b** of cost **c** (**c** <= 1000000),
- The next lines contain instructions **"CHANGE i ti"** or **"QUERY a b"**,
- The end of each test case is signified by the string "**DONE**".

There is one blank line between successive tests.

Output

For each "**QUERY**" operation, write one integer representing its result.

Example

Input:

```
1

3
1 2 1
2 3 2
QUERY 1 2
CHANGE 1 3
QUERY 1 2
DONE
```

Output:

```
1
3
```

Added by: Thanh-Vy Hua
Date: 2005-06-08
Time limit: 5s
Source limit: 15000B
Languages: All except: C99 strict

SPOJ Problem Set (classical)

376. A concrete simulation

Problem code: ACS

You are given a matrix M of type 1234x5678. It is initially filled with integers 1...1234x5678 in row major order. Your task is to process a list of commands manipulating M. There are 4 types of commands:
"R x y" swap the x-th and y-th row of M ;
"C x y" swap the x-th and y-th column of M ;
"Q x y" write out M(x,y) ;
"W z" write out x and y where z=M(x,y).

Input

A list of valid commands. Input terminated by EOF.

Output

For each "Q x y" write out one line with the current value of M(x,y), for each "W z" write out one line with the value of x and y (interpreted as above) separated by a space.

Input:
```
R 1 2
Q 1 1
Q 2 1
W 1
W 5679
C 1 2
Q 1 1
Q 2 1
W 1
W 5679
```

Output:
```
5679
1
2 1
1 1
5680
2
2 2
1 2
```

Added by: Csaba Noszaly
Date: 2005-06-10
Time limit: 7s
Source limit: 7777B
Languages: All
Resource: Folklore

SPOJ Problem Set (classical)

377. Taxi

Problem code: TAXI

Besides the well known unbelievable lordly major town's history there also current problems in this town. But to understand these problems you should know some facts from the unbelievable lordly major town's history.

Once upon a time the population of the unbelievable lordly major town grew so much that the citizens were in need of building new houses. As it was not allowed to erect houses outside the city wall they decided to found a new little town directly beside the main town. We will refer to this second town as the new unbelievable lordly major town. In the new unbelievable lordly major town all streets were built as straight lines intersecting at right angles at fixed distances (see picture below).

Road map

Knowing this we can leave the town's history and can focus on nowadays problems. One of these problems is directly connected to the "somnolent naggy festival" (SONAFE). Despite its name it's one of the town's most popular events and nearly everybody wants to get a ticket. To give everyone the same chance of getting one of the few tickets the place and time of the advance sale are kept secret until some minutes before the ticket counter opens. Once the opening of the ticket counter is disclosed (by radio to give everyone a fair chance) everyone interested in getting some tickets tries to get to the counter immediately.

One of the most profiting citizens of this ticket selling procedure is the new unbelievable lordly major town's taxi service owner. At the time of the radio announcement all over the town people ring up the taxi central and ask for a ride. The problem for the taxi central is that a lot of people ask for a ride at the same time and that the taxis have to pick up the people very quickly.

Your task is to help the taxi central finding out how many passengers can be transported to the ticket counter. You have to adhere to following constraints:

- each taxi can only take one passenger
- passengers always wait at intersections of roads
- at the time of the radio broadcast all taxis also wait at intersections
- the taxi has to reach the passenger within a given time limit (otherwise he will be too late to get a ticket)

Input

The first line contains the number of testcases k (k<=250). The first line of each testcase contains the number of persons p (1<=p<=400), the number of taxicabs t (1<=t<=200) the speed s (1<=s<=2000) of the taxis in meters per seconds and the time c to collect a person in seconds (1<=c<=1000000). The next p lines contains the position of the persons. The next t lines contain the position of the taxicabs in the city.

Output

For each testcase output the maximal number of persons that visit the party.

Example

Input:

```
1
2 3 10 40
2 5
5 2
2 3
4 1
4 4
```

Output:

```
2
```

Added by:	Simon Gog
Date:	2005-06-20
Time limit:	22s
Source limit:	50000B
Languages:	All
Resource:	Ulm Algorithm Course SoSe 2005

SPOJ Problem Set (classical)

379. Ambiguous Permutations

Problem code: PERMUT2

Some programming contest problems are really tricky: not only do they require a different output format from what you might have expected, but also the sample output does not show the difference. For an example, let us look at permutations.
A **permutation** of the integers *1* to *n* is an ordering of these integers. So the natural way to represent a permutation is to list the integers in this order. With *n = 5*, a permutation might look like 2, 3, 4, 5, 1.
However, there is another possibility of representing a permutation: You create a list of numbers where the *i*-th number is the position of the integer *i* in the permutation. Let us call this second possibility an **inverse permutation**. The inverse permutation for the sequence above is 5, 1, 2, 3, 4.
An **ambiguous permutation** is a permutation which cannot be distinguished from its inverse permutation. The permutation 1, 4, 3, 2 for example is ambiguous, because its inverse permutation is the same. To get rid of such annoying sample test cases, you have to write a program which detects if a given permutation is ambiguous or not.

Input Specification

The input contains several test cases.
The first line of each test case contains an integer *n* (*1 <= n <= 100000*). Then a permutation of the integers *1* to *n* follows in the next line. There is exactly one space character between consecutive integers. You can assume that every integer between *1* and *n* appears exactly once in the permutation.
The last test case is followed by a zero.

Output Specification

For each test case output whether the permutation is ambiguous or not. Adhere to the format shown in the sample output.

Sample Input

```
4
1 4 3 2
5
2 3 4 5 1
1
1
0
```

Sample Output

```
ambiguous
not ambiguous
ambiguous
```

Added by:	Adrian Kuegel
Date:	2005-06-24
Time limit:	10s
Source limit:	50000B
Languages:	All
Resource:	own problem, used in University of Ulm Local Contest 2005

SPOJ Problem Set (classical)

380. Bullshit Bingo

Problem code: BINGO

Bullshit Bingo is a game to make lectures, seminars or meetings less boring. Every player has a card with 5 rows and 5 columns. Each of the 25 cells contains a word (the cell in the middle has always the word "BINGO" written in it). Whenever a player hears a word which is written on his card, he can mark it. The cell in the middle is already marked when the game starts. If a player has marked all the words in a row, a column or a diagonal, he stands up and shouts "BULLSHIT". After this, the game starts over again.

Sitting in a lecture, you observe that some students in the audience are playing Bullshit Bingo. You wonder what the average number of different words is until "BULLSHIT" is exclaimed. For the purpose of this problem, a word consists of letters of the English alphabet (’a’ to ’z’ or ’A’ to ’Z’). Words are separated by characters other than letters (for example spaces, digits or punctuation). Do the comparison of words case-insensitively, i.e. "Bingo" is the same word as "bingo". When counting the number of different words, ignore the word BULLSHIT (indicating the end of the game), and consider only the words of the current game, i.e., if a word has already occurred in a previous game, you may still count it in the current game. If the last game is unfinished, ignore the words of that game.

Input Specification

The input file consists of the text of the lecture, with "BULLSHIT" occurring occasionally. The first game starts with the first word in the input. Each occurrence of "BULLSHIT" indicates the end of one game.
You may assume, that

- the word "BULLSHIT" occurs only in uppercase letters
- every word has at most 25 characters, and each line has at most 100 characters
- there are at most 500 different words before a game ends
- the players follow the rules, so there is no need to check if a game is valid or not
- at least one game is completed

Output Specification

The output consists of one number: the average number of different words needed to win a game. Write the number as a reduced fraction in the format shown below. Reduced fraction means that there should be no integer greater than 1 which divides both the numerator and denominator. For example if there were 10 games, and the number of different words in each game summed up to 55, print "11 / 2".

Sample Input

```
Programming languages can be classified BULLSHIT into following types:
- imperative and BULLSHIT procedural languages
- functional languages
- logical BULLSHIT programming languages
- object-oriented BULLSHIT languages
```

Sample Output

```
9 / 2
```

In the sample input, there are 4 completed games. The number of different words is 5, 5, 4 and 4, respectively.

Added by: Adrian Kuegel
Date: 2005-06-24
Time limit: 5s
Source limit: 50000B
Languages: All
Resource: University of Ulm Local Contest 2005

SPOJ Problem Set (main)

381. 106 miles to Chicago

Problem code: CHICAGO

In the movie "Blues Brothers", the orphanage where Elwood and Jack were raised may be sold to the Board of Education if they do not pay 5000 dollars in taxes at the Cook Country Assessor's Office in Chicago. After playing a gig in the Palace Hotel ballroom to earn these 5000 dollars, they have to find a way to Chicago. However, this is not so easy as it sounds, since they are chased by the Police, a country band and a group of Nazis. Moreover, it is 106 miles to Chicago, it is dark and they are wearing sunglasses.
As they are on a mission from God, you should help them find the safest way to Chicago. In this problem, the safest way is considered to be the route which maximises the probability that they are not caught.

Input Specification

The input file contains several test cases.
Each test case starts with two integers *n* and *m* (*2 <= n <= 100* , *1 <= m <= n*(n-1)/2*). *n* is the number of intersections, *m* is the number of streets to be considered.
The next *m* lines contain the description of the streets. Each street is described by a line containing 3 integers *a*, *b* and *p* (*1 <= a, b <= n* , *a != b*, 1 <= p ≤ 100): *a* and *b* are the two end points of the street and *p* is the probability in percent that the Blues Brothers will manage to use this street without being caught. Each street can be used in both directions. You may assume that there is at most one street between two end points.
The last test case is followed by a zero.

Output Specification

For each test case, calculate the probability of the safest path from intersection *1* (the Palace Hotel) to intersection *n* (the Honorable Richard J. Daley Plaza in Chicago). You can assume that there is at least one path between intersection *1* and *n*.
Print the probability as a percentage with exactly 6 digits after the decimal point. The percentage value is considered correct if it differs by at most 10^{-6} from the judge output. Adhere to the format shown below and print one line for each test case.

Sample Input

```
5 7
5 2 100
3 5 80
2 3 70
2 1 50
3 4 90
4 1 85
3 1 70
0
```

Sample Output

```
61.200000 percent
```

The safest path for the sample input is 1 -> 4 -> 3 -> 5

Added by:	Adrian Kuegel
Date:	2005-06-24
Time limit:	5s
Source limit:	50000B
Languages:	All except: C99 strict
Resource:	University of Ulm Local Contest 2005

SPOJ Problem Set (classical)

382. Decorate the wall

Problem code: DECORATE

After building his huge villa, Mr. Rich cannot help but notice that the interior walls look rather blank. To change that, he starts to hang paintings from his wonderful collection. But soon he realizes that it becomes quite difficult to find a place on the wall where a painting can be placed without overlapping other paintings. Now he needs a program which would tell him, given the already placed paintings, where to place the next painting without moving any other paintings (or indicating that this is impossible). Paintings have a rectangular shape and are to be placed parallel to the side of the wall. If you do not mind a nice reward from Mr. Rich, go on and solve the problem.

Input Specification

The first line of the input file contains a number representing the number of test cases to follow. Each test case starts with a line containing three numbers *n*, *w* and *h*. *n* is the number of paintings already hanging on the wall, *w* is the width of the wall and *h* is the height of the wall.
The next *n* lines contain 4 integers x_1, y_1, x_2, y_2 each ($0 <= x_1 < x_2 <= w$, $0 <= y_1 < y_2 <= h$); the x-coordinates give the distance to the left end of the wall, the y-coordinates give the distance to the bottom of the wall. (x_1, y_1) is the position of the lower left corner of a painting, (x_2, y_2) is the position of the upper right corner. The last line of each test case contains the dimensions of the next painting to be placed, first its width *w'*, then its height *h'* ($1 <= w' <= w$, $1 <= h' <= h$). You are not allowed to rotate the painting.
You can assume that $0 <= n <= 200$ and $1 <=$ w, h $<=$ 1000000. Moreover, all paintings already hanging do not overlap.

Output Specification

Produce one line of output for each test case. Write "Fail!" if there is no place left on the wall where the painting could be placed without overlapping other paintings. Otherwise, write the coordinates where the lower left corner of the painting should be placed. In case there is more than one solution, select the solution with a minimum y-coordinate, and break ties using the minimum x-coordinate.

Sample Input

```
2
1 10 9
5 4 10 9
9 5
2 10 10
5 5 10 10
0 0 4 3
3 4
```

The following image illustrates the second sample test case:
[IMAGE]

Sample Output

```
Fail!
4 0
```

Added by: Adrian Kuegel
Date: 2005-06-24
Time limit: 6s
Source limit: 50000B
Languages: All
Resource: University of Ulm Local Contest 2005

SPOJ Problem Set (classical)

383. European railroad tracks

Problem code: EUROPEAN

As you may already know, different countries in Europe use different railroad systems. Not only do they use different voltages for their trains, but also the distance between the two rails (gauge) differs. The following table shows some railway gauges used:

Broad gauge (Spain):	**1674 mm**
Broad gauge (Portugal):	**1665 mm**
Broad gauge (Ireland):	**1600 mm**
Broad gauge (Finland):	**1524 mm**
Broad gauge (former USSR):	**1520 mm**
Standard gauge:	**1435 mm**
Narrow gauge (meter gauge):	**1000 mm**

A museum has trains from several countries. It needs tracks for every train type in order to show visitors the trains in use. However, since only one train is used at a time, a rail can be used by trains of different types. It follows that for *n* trains, each requiring a different railway gauge, *n + 1* rails are sufficient (each train uses the leftmost rail and a rail that has exactly the required distance to it). But sometimes it is possible to save even more rails.

Given the required railway gauges, your task is to construct a railway track that can be used by every train and requires the least number of rails. Note that a train can use any two rails, provided the distance between them is right.

Input Specification

The first line of the input file contains a number representing the number of test cases to follow. Each test case starts with an integer *n* (the number of different railway gauges required). The next line contains *n* integers between *1000* and *5000*, each defining one required railway gauge.
You can assume that *1* <= *n* <= *8*. Moreover, for every test case in the input file, there will be a solution requiring at most *5* rails.

Output Specification

The output for each test case consists of three lines:
The first line is of the form "Scenario #X", where X is the test case number starting with 1. The second line describes the solution your program has found; first your program should print how many rails are needed, followed by a colon, then the positions of each rail in increasing order (the first rail should be at position 0). The third line should be blank. If there are several solutions with the minimum number of rails, any one will do.

Sample Input

```
3
4
1524 1520 1609 1435
3
1000 1520 1600
6
1000 2000 3000 4000 1500 2500
```

Sample Output

```
Scenario #1
4: 0 1520 1609 3044

Scenario #2
4: 0 1000 1520 1600

Scenario #3
5: 0 1500 3000 4000 5000
```

Added by: Adrian Kuegel
Date: 2005-06-24
Time limit: 10s
Source limit: 50000B
Languages: All
Resource: University of Ulm Local Contest 2005

SPOJ Problem Set (classical)

384. Any fool can do it

Problem code: FOOL

Surely you know someone who thinks he is very clever. You decide to let him down with the following problem:

- "Can you tell me what the syntax for a set is?", you ask him.
- "Sure!", he replies, "a set encloses a possibly empty list of elements within two curly braces. Each element is either another set or a letter of the given alphabet. Elements in a list are separated by a comma."
- "So if I give you a word, can you tell me if it is a syntactically correct representation of a set?"
- "Of course, any fool can do it!" is his answer.

Now you got him! You present him with the following grammar, defining formally the syntax for a set (which was described informally by him):

```
Set          ::= "{" Elementlist "}"
Elementlist  ::= <empty> | List
List         ::= Element | Element "," List
Element      ::= Atom | Set
Atom         ::= "{" | "}" | ","
```

<empty> stands for the empty word, i.e. the list in a set can be empty.
Soon he realizes that this task is much harder than he has thought, since the alphabet consists of the characters which are also used for the syntax of the set. So he claims that it is not possible to decide efficiently if a word consisting of `"{"`, `"}"` and `","` is a syntactically correct representation of a set or not.
To disprove him, you need to write an efficient program that will decide this problem.

Input Specification

The first line of the input file contains a number representing the number of lines to follow.
Each line consists of a word, for which your program has to decide if it is a syntactically correct representation of a set. You may assume that each word consists of between *1* and *200* characters from the set { `"{", "}", ","` }.

Output Specification

Output for each test case whether the word is a set or not. Adhere to the format shown in the sample output.

Sample Input

```
4
```

```
{}
{{}}
{{}},{,}}
{,,}
```

Sample Output

```
Word #1: Set
Word #2: Set
Word #3: Set
Word #4: No Set
```

Added by:	Adrian Kuegel
Date:	2005-06-24
Time limit:	10s
Source limit:	50000B
Languages:	All
Resource:	University of Ulm Local Contest 2005

SPOJ Problem Set (classical)

385. Game schedule required

Problem code: GAME

Sheikh Abdul really loves football. So you better don't ask how much money he has spent to make famous teams join the annual tournament. Of course, having spent so much money, he would like to see certain teams play each other. He worked out a complete list of games he would like to see. Now it is your task to distribute these games into rounds according to following rules:

- In each round, each remaining team plays at most one game
- If there is an even number of remaining teams, every team plays exactly one game
- If there is an odd number of remaining teams, there is exactly one team which plays no game (it advances with a wildcard to the next round)
- The winner of each game advances to the next round, the loser is eliminated from the tournament
- If there is only one team left, this team is declared the winner of the tournament

As can be proved by induction, in such a tournament with *n* teams, there are exactly *n - 1* games required until a winner is determined.
Obviously, after round 1, teams may already have been eliminated which should take part in another game. To prevent this, for each game you also have to tell which team should win.

Input Specification

The input file contains several test cases.
Each test case starts with an integer *n* (*2 <= n <= 1000*), the number of teams participating in the tournament. The following *n* lines contain the names of the teams participating in the tournament. You can assume that each team name consists of up to *25* letters of the English alphabet ('a' to 'z' or 'A' to 'Z').
Then follow *n - 1* lines, describing the games the sheikh would like to see (in any order). Each line consists of the two names of the teams which take part in that game. You can assume that it is always possible to find a tournament schedule consisting of the given games.
The last test case is followed by a zero.

Output Specification

For each test case, write the game schedule, distributed in rounds.
For each round, first write "Round #X" (where X is the round number) in a line by itself. Then write the games scheduled in this round in the form: "A defeats B", where A is the name of the advancing team and B is the name of the team being eliminated. You may write the games of a round in any order. If a wildcard is needed for the round, write "A advances with wildcard" after the last game of the round, where A is the name of the team which gets the wildcard. After the last round, write the winner in the format shown below. Print a blank line after each test case.

Sample Input

```
3
A
B
C
A B
B C
5
A
B
C
D
E
A B
C D
A E
C E
0
```

Sample Output

```
Round #1
B defeats A
C advances with wildcard
Round #2
C defeats B
Winner: C

Round #1
A defeats B
C defeats D
E advances with wildcard
Round #2
E defeats A
C advances with wildcard
Round #3
E defeats C
Winner: E
```

Note that there is always more than one possible game schedule; you may print any of them.

Added by: Adrian Kuegel
Date: 2005-06-24
Time limit: 10s
Source limit:50000B
Languages: All
Resource: own problem, used in University of Ulm Local Contest 2005

SPOJ Problem Set (classical)

386. Help the problem setter

Problem code: HELP

Preparing a problem for a programming contest takes a lot of time. Not only do you have to write the problem description and write a solution, but you also have to create difficult input files. In this problem, you get the chance to help the problem setter to create some input for a certain problem.
For this purpose, let us select the problem which was not solved during last year's local contest. The problem was about finding the optimal binary search tree, given the probabilities that certain nodes are accessed. Your job will be: given the desired optimal binary search tree, find some access probabilities for which this binary search tree is the unique optimal binary search tree. Don't worry if you have not read last year's problem, all required definitions are provided in the following.
Let us define a **binary search tree** inductively as follows:

- The empty tree which has no node at all is a binary search tree
- Each non-empty binary search tree has a root, which is a node labelled with an integer, and two binary search trees as left and right subtree of the root
- A left subtree contains no node with a label >= than the label of the root
- A right subtree contains no node with a label <= than the label of the root

Given such a binary search tree, the following **search procedure** can be used to locate a node in the tree:
Start with the root node. Compare the label of the current node with the desired label. If it is the same, you have found the right node. Otherwise, if the desired label is smaller, search in the left subtree, otherwise search in the right subtree.
The **access cost** to locate a node is the number of nodes you have to visit until you find the right node.
An **optimal binary search tree** is a binary search tree with the minimum expected access cost.

Input Specification

The input file contains several test cases.
Each test case starts with an integer n ($1 <= n <= 50$), the number of nodes in the optimal binary search tree. For simplicity, the labels of the nodes will be integers from *1* to n. The following n lines describe the structure of the tree. The i-th line contains the labels of the roots of the left and right subtree of the node with label i (or -1 for an empty subtree). You can assume that the input always defines a valid binary search tree.
The last test case is followed by a zero.

Output Specification

For each test case, write one line containing the access frequency for each node in increasing order of the labels of the nodes. To avoid problems with floating point precision, the frequencies should be written as integers, meaning the access probability for a node will be the frequency divided by the sum of all frequencies. Make sure that you do not write any integer bigger than $2^{63} - 1$ (the maximum value fitting in the C/C++ data type *long long* or the Java data type *long*). Otherwise, you may produce any solution ensuring that there is exactly one optimal binary search tree: the binary search tree given

in the input.

Sample Input

```
3
-1 -1
1 3
-1 -1
10
-1 2
-1 3
-1 4
-1 5
-1 6
-1 7
-1 8
-1 9
-1 10
-1 -1
0
```

Sample Output

```
1 1 1
512 256 128 64 32 16 8 4 2 1
```

Note that the first test case in the sample input describes a tree looking like

```
  2
 / \
1   3
```

Added by: Adrian Kuegel
Date: 2005-06-24
Time limit: 5s
Source limit: 50000B
Languages: All
Resource: own problem, used in University of Ulm Local Contest 2005

SPOJ Problem Set (classical)

387. Travelling tours

Problem code: TOURS

In Hanoi, there are N beauty-spots (2 <= N <= 200), connected by M one-way streets. The length of each street does not exceed 10000. You are the director of a travel agency, and you want to create some tours around the city which satisfy the following conditions:

- Each of the N beauty-spots belongs to exactly one tour.
- Each tour is a cycle which consists of at least 2 places and visits each place once (except for the place we start from which is visited twice).
- The total length of all the streets we use is minimal.

Input

The first line of input contains the number of testcases t (t <= 15). The first line of each testcase contains the numbers N, M. The next M lines contain three integers U V W which mean that there is one street from U to V of length W.

Output

For each test case you shold output the minimal total length of all tours.

Example

Input:

```
2
6 9
1 2 5
2 3 5
3 1 10
3 4 12
4 1 8
4 6 11
5 4 7
5 6 9
6 5 4
5 8
1 2 4
2 1 7
1 3 10
3 2 10
3 4 10
4 5 10
5 3 10
5 4 3
```

Output:

```
42
40
```

Detailed explanation:

```
Test 1:
  Tour #1:  1 - 2 - 3 - 1  --> Length = 20
  Tour #2:  6 - 5 - 4 - 6  --> Length = 22

Test 2:
  Tour #1:  1 - 3 - 2 - 1  --> Length = 27
  Tour #2:  5 - 4 - 5      --> Length = 13
```

Added by: Le Đôn Khue
Date: 2005-06-25
Time limit: 5s
Source limit:50000B
Languages: All
Resource: 2nd round VOI 2004

SPOJ Problem Set (classical)

388. Menu

Problem code: MENU

Alfred wants to plan what to cook in the next days. He can cook various dishes. For each dish the costs of the ingredients and the benefit value is known. If a dish is cooked the second time in a row, the benefit value for the second time is 50 percent of the benefit value of first time, if it is prepared for the third or higher time in a row, the benefit value is 0. For example cooking a dish with benefit value v three times in a row leads to a total benefit value of 1.5*v.
Help him to build the menu which maximizes the benefit value under the constraint that his budget is not exceeded.

Input

The input consists of several test cases. Each test case begins with 3 integers in a line: The number of days k ($1 <= k <= 21$) Alfred wants to plan for, the number of dishes n ($1 <= n <= 50$) he can cook and his budget m ($0 <= m <= 100$). The following n lines describe the dishes Alfred can cook. The i-th line contains two integers: the costs c ($1 <= c <= 50$) and the benefit value v ($1 <= v <= 10000$) of the i-th dish.
The end of the input is signaled by a test case with $k = n = m = 0$. You don't need to process this test case.

Output

For each output, print the maximum benefit value reachable with 1 digit after the decimal point. Then print k integers with i-th integer being the number of the dish to cook on day i. Dishes are numbered from 1 to n. Print at least one space or new line character after each integer.
If there are several possible menus reaching the maximum benefit value, select the one with minimum costs, if there are several with minimum costs, you can print any of them.
If every menu exceeds the budget, print only the benefit value of 0.

Example

Input:
```
2 1 5
3 5
3 5 20
2 5
18 6
1 1
3 3
2 3
0 0 0
```

Output:
```
0.0

13.0
1 5 1
```

Added by:	Adrian Kuegel
Date:	2005-07-04
Time limit:	5s
Source limit:	50000B
Languages:	All
Resource:	own problem

SPOJ Problem Set (classical)

389. Use of Hospital Facilities

Problem code: HOSPITAL

County General Hospital is trying to chart its course through the troubled waters of the economy and shifting population demographics. To support the planning requirements of the hospital, you have been asked to develop a simulation program that will allow the hospital to evaluate alternative configurations of operating rooms, recovery rooms and operations guidelines. Your program will monitor the usage of operating rooms and recovery room beds during the course of one day.

County General Hospital has several operating rooms and recovery room beds. Each surgery patient is assigned to an available operating room and following surgery the patient is assigned to one of the recovery room beds. The amount of time necessary to transport a patient from an operating room to a recovery room is fixed and independent of the patient. Similarly, both the amount of time to prepare an operating room for the next patient and the amount of time to prepare a recovery room bed for a new patient are fixed.

All patients are officially scheduled for surgery at the same time, but the order in which they actually go into the operating rooms depends on the order of the patient roster. A patient entering surgery goes into the lowest numbered operating room available. For example, if rooms 2 and 4 become available simultaneously, the next patient on the roster not yet in surgery goes into room 2 and the next after that goes into room 4 at the same time. After surgery, a patient is taken to the available recovery room bed with the lowest number. A recovery room bed is only available if the preparation is already finished when the patient leaves surgery. If two patients emerge from surgery at the same time, the patient with the lower surgery room number will be the first assigned to a recovery room bed.

Input

The input file contains data for several simulation runs. Each run is separated by a blank line. All numeric data in the input file are integers, and successive integers on the same line are separated by blanks. The first line of each run is the set of hospital configuration parameters to be used for this run. The parameters are, in order:

- Number of operating rooms (maximum of 10)
- Number of recovery room beds (maximum of 30)
- Starting hour for 1st surgery of day (based on a 24-hour clock)
- Minutes to transport patient from operating room to recovery room
- Minutes to prepare operating room for next patient
- Minutes to prepare recovery room bed for next patient
- Number of surgery patients for the day (maximum of 100)

This initial configuration data will be followed by pairs of lines of patient data as follows:

- Line 1: Last name of patient (maximum of 8 characters)
- Line 2: Minutes required for surgery Minutes required in the recovery room

Patient records in the input file are ordered according to the patient roster, which determines the order in which patients are scheduled for surgery. The number of recovery room beds specified in any configuration will be sufficient to handle patients arriving from surgery (No queuing of patients for recovery room beds will be required). Computed times will not extend past 24:00.

Output

Correct output shows which operating room and which recovery room bed is used by each patient, and the time period that the patient uses the room and bed along with a summary of the utilization of hospital facilities for that day. The output file consists of several sets of two tables each describing the results of the simulation run. The first table is in columnar form with appropriate column labels to show the number of each patient (in the order the patient roster), the patient's last name, the operating room number, the time surgery begins and ends, the recovery bed number and the time the patient enters and leaves the recovery room bed.

The second table will also be in columnar form with appropriate column labels summarizing the utilization of operating rooms and recovery room beds. This summary indicates the facility type (room or bed), the facility number, the number of minutes used and percentage of available time utilized. Available time is defined as the time in minutes from the starting time for 1st surgery of day to the ending time of the last patient in a recovery room bed. Print a blank line after each run. Follow the output format shown on sample output.

Example

Input:

```
5 12 07 5 15 10 16
Jones
28 140
Smith
120 200
Thompson
23 75
Albright
19 82
Poucher
133 209
Comer
74 101
Perry
93 188
Page
111 223
Roggio
69 122
Brigham
42 79
Nute
22 71
Young
38 140
Bush
26 121
Cates
120 248
Johnson
86 181
```

```
White
92 140
```

Output:

```
 Patient          Operating Room          Recovery Room
 #  Name     Room#  Begin   End      Bed#  Begin    End
 ------------------------------------------------------
 1  Jones      1    7:00    7:28      3    7:33    9:53
 2  Smith      2    7:00    9:00      1    9:05   12:25
 3  Thompson   3    7:00    7:23      2    7:28    8:43
 4  Albright   4    7:00    7:19      1    7:24    8:46
 5  Poucher    5    7:00    9:13      5    9:18   12:47
 6  Comer      4    7:34    8:48      4    8:53   10:34
 7  Perry      3    7:38    9:11      2    9:16   12:24
 8  Page       1    7:43    9:34      6    9:39   13:22
 9  Roggio     4    9:03   10:12      9   10:17   12:19
10  Brigham    2    9:15    9:57      8   10:02   11:21
11  Nute       3    9:26    9:48      7    9:53   11:04
12  Young      5    9:28   10:06      3   10:11   12:31
13  Bush       1    9:49   10:15     10   10:20   12:21
14  Cates      3   10:03   12:03      8   12:08   16:16
15  Johnson    2   10:12   11:38      4   11:43   14:44
16  White      5   10:21   11:53      7   11:58   14:18

Facility Utilization
Type  # Minutes  % Used
------------------------
Room  1     165   29.68
Room  2     248   44.60
Room  3     258   46.40
Room  4     162   29.14
Room  5     263   47.30
Bed   1     282   50.72
Bed   2     263   47.30
Bed   3     280   50.36
Bed   4     282   50.72
Bed   5     209   37.59
Bed   6     223   40.11
Bed   7     211   37.95
Bed   8     327   58.81
Bed   9     122   21.94
Bed  10     121   21.76
Bed  11       0    0.00
Bed  12       0    0.00
```

Added by: Adrian Kuegel
Date: 2005-07-04
Time limit: 3s
Source limit:50000B
Languages: All
Resource: ACM ICPC World Finals 1991

SPOJ Problem Set (classical)

390. Billiard

Problem code: BILLIARD

In a billiard table with horizontal side **a** inches and vertical side **b** inches, a ball is launched from the middle of the table. After **s** > 0 seconds the ball returns to the point from which it was launched, after having made **m** bounces off the vertical sides and **n** bounces off the horizontal sides of the table. Find the launching angle **A** (measured from the horizontal), which will be between 0 and 90 degrees inclusive, and the initial velocity of the ball.

Assume that the collisions with a side are elastic (no energy loss), and thus the velocity component of the ball parallel to each side remains unchanged. Also, assume the ball has a radius of zero. Remember that, unlike pool tables, billiard tables have no pockets.

Input

Input consists of a sequence of lines, each containing five nonnegative integers separated by whitespace. The five numbers are: **a**, **b**, **s**, **m**, and **n**, respectively. All numbers are positive integers not greater than 10000.

Input is terminated by a line containing five zeroes.

Output

For each input line except the last, output a line containing two real numbers (accurate to two decimal places) separated by a single space. The first number is the measure of the angle **A** in degrees and the second is the velocity of the ball measured in inches per second, according to the description above.

Example

Input:

```
100 100 1 1 1
200 100 5 3 4
201 132 48 1900 156
0 0 0 0 0
```

Output:

```
45.00 141.42
33.69 144.22
3.09 7967.81
```

Added by: Adrian Kuegel
Date: 2005-07-04
Time limit: 3s
Source limit: 50000B
Languages: All
Resource: University of Waterloo Local Contest (Spring 1999)

SPOJ Problem Set (classical)

391. Railroads

Problem code: RAILROAD

It's Friday evening and Jill hates two things which are common to all trains:

1. They are always late.
2. The schedule is always wrong.

Nevertheless, tomorrow in the early morning hours Jill will have to travel from Hamburg to Darmstadt in order to get to the regional programming contest. Since she is afraid of arriving too late and being excluded from the contest she is looking for the train which gets her to Darmstadt as early as possible. However, she dislikes to get to the station too early, so if there are several schedules with the same arrival time then she will choose the one with the latest departure time.

Jill asks you to help her with her problem. You are given a set of railroad schedules from which you must compute the train with the earliest arrival time and the fastest connection from one location to another. One good thing: Jill is very experienced in changing trains. She can do this instantaneously, i.e., in zero time!!!

Input

The very first line of the input gives the number of scenarios. Each scenario consists of three parts.

Part one lists the names of all cities connected by the railroads. It starts with a number $1 < C <= 100$, followed by C lines containing city names. These names consist of letters.

Part two describes all the trains running during a day. It starts with a number $T <= 1000$ followed by T train descriptions. Each of them consists of one line with a number $t_i <= 100$ and t_i more lines with a time and a city name, meaning that passengers can get on or off the train at that time at that city.

Part three consists of three lines: Line one contains the earliest journey's starting time, line two the name of the city where she starts, and line three the destination city. The two cities are always different.

Output

For each scenario print a line containing "Scenario i", where i is the number of the scenario starting at 1.

If a connection exists then print the two lines containing zero padded timestamps and locations as shown in the sample. Use blanks to achieve the indentation. If no connection exists on the same day (i.e., arrival before midnight) then print a line containing "No connection".

After each scenario print a blank line.

Example

Input:

```
2
3
Hamburg
Frankfurt
Darmstadt
3
2
0949 Hamburg
1006 Frankfurt
2
1325 Hamburg
1550 Darmstadt
2
1205 Frankfurt
1411 Darmstadt
0800
Hamburg
Darmstadt
2
Paris
Tokyo
1
2
0100 Paris
2300 Tokyo
0800
Paris
Tokyo
```

Output:

```
Scenario 1
Departure 0949 Hamburg
Arrival   1411 Darmstadt

Scenario 2
No connection
```

Added by: Adrian Kuegel
Date: 2005-07-04
Time limit: 5s
Source limit:50000B
Languages: All
Resource: ACM Southwestern European Regional Contest 2000

SPOJ Problem Set (classical)

392. Spin

Problem code: SPIN

tex2html_wrap62 tex2html_wrap64

A disk can be rotated between horizontal and vertical only if it is positioned over the indentation marked '0' *and* the disk on its right is vertical . The right-most disk can always rotate if it is in position '0' since it has no disk on its right.

The aim is to free the slide by moving it so its left edge aligns with the 'Win' mark:

[IMAGE]

Your task is to write a program which will take several part-solved puzzles and compute the number of steps needed to move the slide to position 'Win' for each puzzle.

Input

There will be several puzzles in the input file. The first line of the file will contain an integer *n* specifying the number of puzzles. There will then be *n* lines, each of the form:

length orientations position

where *length*(length < 30) is an integer indicating the number of disks on the slide, *orientations* is a string of *length* characters from the set {h,v} giving the orientation of each disk from left to right (h stands for horizontal, and v for vertical), and *position* is an integer from 0 to *length* specifying the numbered mark which aligns with the left edge of the slide.

Output

For each puzzle, your program should output one integer on a line which counts the minimum number of steps needed to win the puzzle. A step is either a movement of the slide, one unit left or right, or the rotation of a disk.

Example

Input:

```
3
2 vv 2
7 vhhhvhh 4
29 vvvvvvvvvvvvvvvvvvvvvvvvvvvvv 29
```

Output:

```
7
357
1073741823
```

Added by: Adrian Kuegel
Date: 2005-07-04
Time limit: 5s
Source limit: 50000B
Languages: All
Resource: ACM Northwestern European Regional Contest 1993 (UVA problem 279)

SPOJ Problem Set (classical)

393. Hexagon

Problem code: HEXAGON

Consider a game board consisting of 19 hexagonal fields, as shown in the figure below. We can easily distinguish three main directions in the shape of the board: from top to bottom, from top-left to bottom-right, and from top-right to bottom-left. For each of these primary directions, the board can be viewed as a series of rows, consisting of 3, 4, 5, 4, and 3 fields, respectively.

[IMAGE]

The game board has to be completely covered using a set of hexagonal pieces. Each piece carries three numbers, one for every primary board direction. Only three different numbers are used for each direction. Every possible combination of three numbers for all three directions is assigned to a piece, leading to a set of 27 unique pieces. (The board in the above figure is still in the process of being covered.)

The score of a board is calculated as the sum of all 15 row scores (5 rows for each primary direction). The row scores are calculated as follows: if all pieces in a row carry the same number for the direction of the row, the row score is this number multiplied by the number of pieces in the row. Otherwise (the pieces carry different numbers in the row direction) the row score is zero. Note that the pieces may not be rotated. For example, the score of the leftmost row in the figure is $3 \cdot 3 = 9$, the score of the row to its right is $4 \cdot 11 = 44$.

While in the real game the pieces are chosen randomly and the set of pieces is fixed, we are interested in the highest possible score for a given set of numbers for each direction, when all pieces in a row carry the same number for the direction of the row. This means you have to choose those 19 pieces that result in the highest score under the constraint that each row has a score greater than zero.

Input

The first line of the input file contains an integer **n** which indicates the number of test cases. Each test case consists of three lines containing three different positive integers each. Each of these three lines contains the numbers for a single primary direction. From these numbers the set of pieces is generated.

Output

For each test case output a line containing the number of the case (`'Test #1'`, `'Test #2'`, etc.), followed by a line containing the highest possible score for the given numbers. Add a blank line after each test case.

Example

Input:

```
1
9 4 3
8 5 2
```

```
7 6 1
```

Output:

```
Test #1
308
```

Added by: Adrian Kuegel
Date: 2005-07-04
Time limit: 40s
Source limit: 50000B
Languages: All
Resource: ACM Southwestern European Regional Contest 1996

SPOJ Problem Set (classical)

394. Alphacode

Problem code: ACODE

Alice and Bob need to send secret messages to each other and are discussing ways to encode their messages:

Alice: "Let's just use a very simple code: We'll assign 'A' the code word 1, 'B' will be 2, and so on down to 'Z' being assigned 26."

Bob: "That's a stupid code, Alice. Suppose I send you the word 'BEAN' encoded as 25114. You could decode that in many different ways!"

Alice: "Sure you could, but what words would you get? Other than 'BEAN', you'd get 'BEAAD', 'YAAD', 'YAN', 'YKD' and 'BEKD'. I think you would be able to figure out the correct decoding. And why would you send me the word 'BEAN' anyway?"

Bob: "OK, maybe that's a bad example, but I bet you that if you got a string of length 5000 there would be tons of different decodings and with that many you would find at least two different ones that would make sense."

Alice: "How many different decodings?"

Bob: "Jillions!"

For some reason, Alice is still unconvinced by Bob's argument, so she requires a program that will determine how many decodings there can be for a given string using her code.

Input

Input will consist of multiple input sets. Each set will consist of a single line of at most 5000 digits representing a valid encryption (for example, no line will begin with a 0). There will be no spaces between the digits. An input line of '0' will terminate the input and should not be processed.

Output

For each input set, output the number of possible decodings for the input string. All answers will be within the range of a 64 bit signed integer.

Example

Input:

```
25114
1111111111
3333333333
0
```

Output:

```
6
89
1
```

Added by: Adrian Kuegel
Date: 2005-07-09
Time limit: 2s
Source limit: 50000B
Languages: All
Resource: ACM East Central North America Regional Programming Contest 2004

SPOJ Problem Set (classical)

395. Anti-prime Sequences

Problem code: APRIME

Given a sequence of consecutive integers n, n+1, n+2,..., m, an anti-prime sequence is a rearrangement of these integers so that each adjacent pair of integers sums to a composite (non-prime) number. For example, if n = 1 and m = 10, one such anti-prime sequence is 1, 3, 5, 4, 2, 6, 9, 7, 8, 10. This is also the lexicographically first such sequence.

We can extend the definition by defining a degree d anti-prime sequence as one where all consecutive subsequences of length 2, 3,..., d sum to a composite number. The sequence above is a degree 2 anti-prime sequence, but not a degree 3, since the subsequence 5, 4, 2 sums to 11. The lexicographically first degree 3 anti-prime sequence for these numbers is 1, 3, 5, 4, 6, 2, 10, 8, 7, 9.

Input

Input will consist of multiple input sets. Each set will consist of three integers, n, m, and d on a single line. The values of n, m and d will satisfy 1 <= n < m <= 1000, and 2 <= d <= 10. The line 0 0 0 will indicate end of input and should not be processed.

Output

For each input set, output a single line consisting of a comma-separated list of integers forming a degree d anti-prime sequence (do not insert any spaces and do not split the output over multiple lines). In the case where more than one anti-prime sequence exists, print the lexicographically first one (i.e., output the one with the lowest first value; in case of a tie, the lowest second value, etc.). In the case where no anti-prime sequence exists, output:

No anti-prime sequence exists.

Example

Input:

```
1 10 2
1 10 3
1 10 5
40 60 7
0 0 0
```

Output:

```
1,3,5,4,2,6,9,7,8,10
1,3,5,4,6,2,10,8,7,9
No anti-prime sequence exists.
40,41,43,42,44,46,45,47,48,50,55,53,52,60,56,49,51,59,58,57,54
```

Added by: Adrian Kuegel
Date: 2005-07-09
Time limit: 15s
Source limit: 50000B
Languages: All
Resource: ACM East Central North America Regional Programming Contest 2004

SPOJ Problem Set (classical)

396. Hit or Miss

Problem code: HITOMISS

One very simple type of solitaire game known as "Hit or Miss" (also known as "Frustration," "Harvest," "Roll-Call," "Talkative", and "Treize") is played as follows: take a standard deck of 52 playing cards -- four sets of cards numbered 1 through 13 (suits do not matter in this game) which have been shuffled -- and start counting through the deck 1, 2, 3, . . . , and so on. When your count reaches 13, start over at 1. Each time you count, look at the top card of the deck and do one of two things: if the number you count matches the value of the top card, discard it from the deck; if it does not match it, move that card to the bottom of the deck. You win the game if you are able to remove all cards from the deck (which may take a very long time).

A version of this game can be devised for two or more players. The first player starts as before with a 52 card deck, while the other players have no cards initially. As the first player removes cards from her deck, she gives them to the second player, who then starts playing the same game, starting at count 1. When that player gets a match, he passes his card to the third player, and so on. The last player discards matches rather than passing them to player 1. All players who have cards to play with perform the following 2-step cycle of moves in lockstep:

1. Each player says his or her current count value and checks for a match. If there is no match, the top card is moved to the bottom of the deck; otherwise it is passed to the next player (or discarded if this is the last player).

2. Each player except the first takes a passed card (if there is one) and places it at the bottom of his or her deck.

These rules are repeated over and over until either the game is won (all the cards are discarded by the last player) or an unwinnable position is reached. If any player ever runs out of cards, he waits until he is passed a card and resumes his count from where he left off (e.g., if player 3 passes his last card on a count of 7, he waits until he receives a card from player 2 and resumes his count with 8 at the beginning of the next 2-step cycle).

Input

Input will consist of multiple input sets. The first line of the file will contain a single positive integer n indicating the number of input sets in the file. Each input set will be a single line containing 53 integers: the first integer will indicate the number of players in the game and the remaining 52 values will be the initial layout of the cards in the deck, topmost card first. These values will all lie in the range 1 . . . 13, and the number of players will lie in the range 1 . . . 10.

Output

For each input set, output the input set number (as shown below, starting with 1) and either the phrase "unwinnable" or a list showing the last card discarded by each player. Use a single blank to separate all outputs.

Example

Input: (note that the line break is only in the sample input for displaying purposes)

```
2
4 1 2 3 4 5 6 7 8 9 10 11 12 13 1 2 3 4 5 6 7 8 9 10 11 12 13
  1 2 3 4 5 6 7 8 9 10 11 12 13 1 2 3 4 5 6 7 8 9 10 11 12 13
4 2 3 4 5 6 7 8 9 10 11 12 13 1 2 3 4 5 6 7 8 9 10 11 12 13 1
  2 3 4 5 6 7 8 9 10 11 12 13 1 2 3 4 5 6 7 8 9 10 11 12 13 1
```

Output:

```
Case 1: 13 13 13 13
Case 2: unwinnable
```

Added by: Adrian Kuegel
Date: 2005-07-09
Time limit: 2s
Source limit: 50000B
Languages: All
Resource: ACM East Central North America Regional Programming Contest 2004

SPOJ Problem Set (classical)

397. I Conduit

Problem code: CONDUIT

Irv Kenneth Diggit works for a company that excavates trenches, digs holes and generally tears up people's yards. Irv's job is to make sure that no underground pipe or cable is underneath where excavation is planned. He has several different maps, one for each utility company, showing where their conduits lie, and he needs to draw one large, consolidated map combining them all. One approach would be to simply draw each of the smaller maps one at a time onto the large map. However, this often wastes time, not to mention ink for the pen-plotter in the office, since in many cases portions of the conduits overlap with each other (albeit at different depths underground). What Irv wants is a way to determine the minimum number of line segments to draw given all the line segments from the separate maps.

Input

Input will consist of multiple input sets. Each set will start with a single line containing a positive integer n indicating the total number of line segments from all the smaller maps. Each of the next n lines will contain a description of one segment in the format

x_1 y_1 x_2 y_2

where (x_1,y_1) are the coordinates of one endpoint and (x_2,y_2) are the coordinates of the other. Coordinate values are floating point values in the range 0 ...1001 specified to at most two decimal places. The maximum number of line segments will be 10000 and all segments will have non-zero length. Following the last input set there will be a line containing a 0 indicating end of input; it should not be processed.

Output

For each input set, output on a single line the minimum number of line segments that need to be drawn on the larger, consolidated map.

Example

Input:

```
3
1.0 10.0 3.0 14.0
0.0 0.0 20.0 20.0
10.0 28.0 2.0 12.0
2
0.0 0.0 1.0 1.0
1.0 1.0 2.15 2.15
2
0.0 0.0 1.0 1.0
1.0 1.0 2.15 2.16
0
```

Output:

```
2
1
2
```

Added by: Adrian Kuegel
Date: 2005-07-09
Time limit: 3s
Source limit: 50000B
Languages: All
Resource: ACM East Central North America Regional Programming Contest 2004

SPOJ Problem Set (classical)

398. Roll Playing Games

Problem code: RPGAMES

Phil Kropotnik is a game maker, and one common problem he runs into is determining the set of dice to use in a game. In many current games, non-traditional dice are often required, that is, dice with more or fewer sides than the traditional 6-sided cube. Typically, Phil will pick random values for all but the last die, then try to determine specific values to put on the last die so that certain sums can be rolled with certain probabilities (actually, instead of dealing with probabilities, Phil just deals with the total number of different ways a given sum can be obtained by rolling all the dice). Currently he makes this determination by hand, but needless to say he would love to see this process automated. That is your task.

For example, suppose Phil starts with a 4-sided die with face values 1, 10, 15, and 20 and he wishes to determine how to label a 5-sided die so that there are a) 3 ways to obtain a sum of 2, b) 1 way to obtain a sum of 3, c) 3 ways to obtain 11, d) 4 ways to obtain 16, and e)1 way to obtain 26. To get these results he should label the faces of his 5-sided die with the values 1, 1, 1, 2, and 6. (For instance, the sum 16 may be obtained as 10 +6 or as 15 +1, with three different "1" faces to choose from on the second die, for a total of 4 different ways.) Note that he sometimes only cares about a subset of the sums reachable by rolling all the dices (like in the previous example).

Input

Input will consist of multiple input sets. Each input set will start with a single line containing an integer n indicating the number of dice that are already specified. Each of the next n lines describes one of these dice. Each of these lines will start with an integer f (indicating the number of faces on the die) followed by f integers indicating the value of each face. The last line of each problem instance will have the form

r m v_1 c_1 v_2 c_2 v_3 c_3 ... v_m c_m

where r is the number of faces required on the unspecified die, m is the number of sums of interest, v_1,...,vsm are these sums, and c_1,...,c_m are the counts of the desired number of different ways in which to achieve each of the respective sums.

Input values will satisfy the following constraints: 1 <= n <= 20, 3 <= f <= 20, 1 <= m <= 10, and 4 <= r <= 6. Values on the faces of all dice, both the specified ones and the unknown die, will be integers in the range 1 ...50, and values for the v_i's and c_i's are all non-negative and are strictly less than the maximum value of a 32-bit signed integer.

The last input set is followed by a line containing a single 0; it should not be processed.

Output

For each input set, output a single line containing either the phrase "Final die face values are" followed by the r face values in non-descending order, or the phrase "Impossible" if no die can be found meeting the specifications of the problem. If there are multiple dice which will solve the problem, choose the one whose lowest face value is the smallest; if there is still a tie, choose the one whose second-lowest face value is smallest, etc.

Example

Input:

```
1
4 1 10 15 20
5 5 2 3 3 1 11 3 16 4 26 1
1
6 1 2 3 4 5 6
6 3 7 6 2 1 13 1
4
6 1 2 3 4 5 6
4 1 2 2 3
3 3 7 9
8 1 4 5 9 23 24 30 38
4 4 48 57 51 37 56 31 63 11
0
```

Output:

```
Final die face values are 1 1 1 2 6
Impossible
Final die face values are 3 7 9 9
```

Added by: Adrian Kuegel
Date: 2005-07-09
Time limit: 15s
Source limit:50000B
Languages: All
Resource: ACM East Central North America Regional Programming Contest 2004

SPOJ Problem Set (classical)

399. Team Rankings

Problem code: TRANK

It's preseason and the local newspaper wants to publish a preseason ranking of the teams in the local amateur basketball league. The teams are the Ants, the Buckets, the Cats, the Dribblers, and the Elephants. When Scoop McGee, sports editor of the paper, gets the rankings from the selected local experts down at the hardware store, he's dismayed to find that there doesn't appear to be total agreement and so he's wondering what ranking to publish that would most accurately re ect the rankings he got from the experts. He's found that finding the median ranking from among all possible rankings is one way to go.

The median ranking is computed as follows: Given any two rankings, for instance ACDBE and ABCDE, the distance between the two rankings is defined as the total number of pairs of teams that are given different relative orderings. In our example, the pair B, C is given a different ordering by the two rankings. (The first ranking has C above B while the second ranking has the opposite.) The only other pair that the two rankings disagree on is B, D; thus, the distance between these two rankings is 2. The median ranking of a set of rankings is that ranking whose sum of distances to all the given rankings is minimal. (Note we could have more than one median ranking.) The median ranking may or may not be one of the given rankings.

Suppose there are 4 voters that have given the rankings: ABDCE, BACDE, ABCED and ACBDE. Consider two candidate median rankings ABCDE and CDEAB. The sum of distances from the ranking ABCDE to the four voted rankings is 1 + 1 + 1 + 1 = 4. We'll call this sum the value of the ranking ABCDE. The value of the ranking CDEAB is 7 + 7+ 7 + 5 = 26. It turns out that ABCDE is in fact the median ranking with a value of 4.

Input

There will be multiple input sets. Input for each set is a positive integer n on a line by itself, followed by n lines (n no more than 100), each containing a permutation of the letters A, B, C, D and E, left-justified with no spaces. The final input set is followed by a line containing a 0, indicating end of input.

Output

Output for each input set should be one line of the form:

ranking is the median ranking with value *value*.

Of course ranking should be replaced by the correct ranking and value with the correct value. If there is more than one median ranking, you should output the one which comes first alphabetically.

Example

Input:

```
4
ABDCE
BACDE
ABCED
ACBDE
0
```

Output:

```
ABCDE is the median ranking with value 4.
```

Added by: Adrian Kuegel
Date: 2005-07-09
Time limit: 3s
Source limit: 50000B
Languages: All
Resource: ACM East Central North America Regional Programming Contest 2004

SPOJ Problem Set (classical)

400. To and Fro

Problem code: TOANDFRO

Mo and Larry have devised a way of encrypting messages. They first decide secretly on the number of columns and write the message (letters only) down the columns, padding with extra random letters so as to make a rectangular array of letters. For example, if the message is "There's no place like home on a snowy night" and there are five columns, Mo would write down

```
t o i o y
h p k n n
e l e a i
r a h s g
e c o n h
s e m o t
n l e w x
```

Note that Mo includes only letters and writes them all in lower case. In this example, Mo used the character 'x' to pad the message out to make a rectangle, although he could have used any letter. Mo then sends the message to Larry by writing the letters in each row, alternating left-to-right and right-to-left. So, the above would be encrypted as

```
toioynnkpheleaigshareconhtomesnlewx
```

Your job is to recover for Larry the original message (along with any extra padding letters) from the encrypted one.

Input

There will be multiple input sets. Input for each set will consist of two lines. The first line will contain an integer in the range 2...20 indicating the number of columns used. The next line is a string of up to 200 lower case letters. The last input set is followed by a line containing a single 0, indicating end of input.

Output

Each input set should generate one line of output, giving the original plaintext message, with no spaces.

Example

Input:

```
5
toioynnkpheleaigshareconhtomesnlewx
3
ttyohhieneesiaabss
0
```

Output:

```
theresnoplacelikehomeonasnowynightx
thisistheeasyoneab
```

Added by: Adrian Kuegel
Date: 2005-07-09
Time limit: 2s
Source limit: 50000B
Languages: All
Resource: ACM East Central North America Regional Programming Contest 2004

SPOJ Problem Set (classical)

401. Translations

Problem code: TRANSL

Bob Roberts is in charge of performing translations of documents between various languages. To aid him in this endeavor his bosses have provided him with translation files. These files come in twos -- one containing sample phrases in one of the languages and the other containing their translations into the other language. However, some over-zealous underling, attempting to curry favor with the higher-ups with his initiative, decided to alphabetically sort the contents of all of the files, losing the connections between the phrases and their translations. Fortunately, the lists are comprehensive enough that the original translations can be reconstructed from these sorted lists. Bob has found this is most usually the case when the phrases all consist of two words. For example, given the following two lists:

Language 1 Phrases	Language 2 Phrases
arlo zym	bus seat
flub pleve	bus stop
pleve dourm	hot seat
pleve zym	school bus

Bob is able to determine that arlo means hot, zym means seat, ub means school, pleve means bus, and dourm means stop. After doing several of these reconstructions by hand, Bob has decided to automate the process. And if Bob can do it, then so can you.

Input

Input will consist of multiple input sets. Each input set starts with a positive integer n, n <= 250, indicating the number of two-word phrases in each language. This is followed by 2n lines, each containing one two-word phrase: the first n lines are an alphabetical list of phrases in the first language, and the remaining n lines are an alphabetical list of their translations into the second language. Only upper and lower case alphabetic characters are used in the words. No input set will involve more than 25 distinct words. No word appears as the first word in more than 10 phrases for any given language; likewise, no word appears as the last word in more than 10 phrases. A line containing a single 0 follows the last problem instance, indicating end of input.

Output

For each input set, output lines of the form

```
word1/word2
```

where word1 is a word in the first language and word2 is the translation of word1 into the second language, and a slash separates the two. The output lines should be sorted according to the first language words, and every first language word should occur exactly once. There should be no white space in the output, apart from a single blank line separating the outputs from different input sets. Imitate the format of the sample output, below. There is guaranteed to be a unique correct translation corresponding to each input instance.

Example

Input:

```
4
arlo zym
flub pleve
pleve dourm
pleve zym
bus seat
bus stop
hot seat
school bus
2
iv otas
otas re
ec t
eg ec
0
```

Output:

```
arlo/hot
dourm/stop
flub/school
pleve/bus
zym/seat

iv/eg
otas/ec
re/t
```

Added by: Adrian Kuegel
Date: 2005-07-09
Time limit: 15s
Source limit: 50000B
Languages: All
Resource: ACM East Central North America Regional Programming Contest 2004

SPOJ Problem Set (classical)

402. Hike on a Graph

Problem code: HIKE

"Hike on a Graph" is a game that is played on a board on which an undirected graph is drawn. The graph is complete and has all loops, i.e. for any two locations there is exactly one arrow between them. The arrows are coloured. There are three players, and each of them has a piece. At the beginning of the game, the three pieces are in fixed locations on the graph. In turn, the players may do a move. A move consists of moving one's own piece along an arrow to a new location on the board. The following constraint is imposed on this: the piece may only be moved along arrows of the same colour as the arrow between the two opponents' pieces.

In the sixties ("make love not war") a one-person variant of the game emerged. In this variant one person moves all the three pieces, not necessarily one after the other, but of course only one at a time. Goal of this game is to get all pieces onto the same location, using as few moves as possible. Find out the smallest number of moves that is necessary to get all three pieces onto the same location, for a given board layout and starting positions.

Input

The input file contains several test cases. Each test case starts with the number n. Input is terminated by $n=0$. Otherwise, $1<=n<=50$. Then follow three integers p_1, p_2, p_3 with $1<=p_i<=n$ denoting the starting locations of the game pieces. The colours of the arrows are given next as a $n \times n$ matrix m of whitespace-separated lower-case letters. The element m_{ij} denotes the colour of the arrow between the locations i and j. Since the graph is undirected, you can assume the matrix to be symmetrical.

Output

For each test case output on a single line the minimum number of moves required to get all three pieces onto the same location, or the word "impossible" if that is not possible for the given board and starting locations.

Example

Input:

```
3 1 2 3
r b r
b b b
r b r
2 1 2 2
y g
g y
0
```

Output:

```
2
impossible
```

Added by: Adrian Kuegel
Date: 2005-07-10
Time limit: 5s
Source limit: 50000B
Languages: All
Resource: University of Ulm Local Contest 2000

SPOJ Problem Set (classical)

403. Sort fractions

Problem code: FRACTION

You are given a positive integer N. Let us consider set A of fractions x/y where 0 <= x/y <= 1, y <= N and the maximum common divisor of x and y is 1.

For example N = 5. Set A in increasing order consists of elements 0/1; 1/5; 1/4; 1/3; 2/5; 1/2; 3/5; 2/3; 3/4; 4/5; 1/1.

Your task is to find the i-th smallest fraction in set A.

Input

The first line of input contains the number of testcases t (t <= 15). The first line of each testcase contains numbers N and M (N <= 5000, M <= 10000). The next M lines contain one question each.

Output

For each testcase, you should output M lines which are the answers to the M questions.

Example

Input:

```
1
5 4
1
3
5
8
```

Output:

```
0/1
1/4
2/5
2/3
```

Added by: Le Đôn Khue
Date: 2005-07-12
Time limit: 2s
Source limit: 10000B
Languages: All
Resource: Mr Nguyen Duc Thinh

SPOJ Problem Set (classical)

404. Scanner

Problem code: SCANNER

A body scanner works by scanning a succession of horizontal slices through the body; the slices are imaged one at a time. The image slices can be reassembled to form a three dimensional model of the object. Write a program to construct a two dimensional image slice using data captured during the scan.

epsfbox{p229.eps}

The scanner consists of four arrays of sensors arranged around a 10×15 matrix. Array 1 consists of 10 sensors pointing to the right, array 2 has 24 sensors pointing diagonally to the top right, array 3 has 15 sensors pointing to the top and array 4 has 24 sensors pointing to the top left. Each sensor records the thickness of that portion of the object directly in front of that sensor.

Readings from the arrays of sensors are recorded in counterclockwise order. Within an array of sensors, data are also recorded counterclockwise. A complete scan consists of 73 readings.

Input

The input file begins with a line with an integer indicating the number of image slices to follow. For each image slice, there are separate lines with 10, 24, 15, and 24 integers representing sensor data from sensor arrays 1 through 4 respectively. The order of the readings is indicated in the diagram. You can assume that there exist at least one image for the given sensor data.

Output

For each slice, your program should print 10 lines of 15 cells. To indicate that the cell represents a part of the object, print a hash character (#) for the cell; to indicate that the cell is not a part of the object, print a period (.). Between successive output image slices, print a blank line.

It is possible for the result of a scan to be ambiguous, in that case you can print any solution.

Example

Input:

```
1
10 10 6 4 6 8 13 15 11 6
0 1 2 2 2 2 4 5 5 6 7 6 5 6 6 5 5 6 6 3 2 2 1 0
2 4 5 5 7 6 7 10 10 10 7 3 3 5 5
0 0 1 3 4 4 4 4 3 4 5 7 8 8 9 9 6 4 4 2 0 0 0 0
```

Output:

```
.##########....
.##########....
....######.....
```

```
......####.....
.......####..##
.......########
#####..########
###############
..#########..##
....######.....
```

Added by: Adrian Kuegel
Date: 2005-07-26
Time limit: 42s
Source limit:50000B
Languages: All
Resource: ACM ICPC World Finals 1993

SPOJ Problem Set (classical)

405. Tin Cutter

Problem code: TCUTTER

In a Tin Cutting factory there is a machine for cutting parts from tin plates. It has an extraordinarily sharp knife able to make horizontal or vertical segment cuts in the tin plates. Each cutting process consists of a sequence of such cuts. Each segment cut is given by its endpoints that are always located inside the tin plate. During the cutting process some parts of tin plate can fall out and so some holes in the plate can emerge.

Factory management needs to predict the number of holes in the plate at the end of the given sequence of cuts. Write a program that answers this question. Single segment cuts are not considered to be holes.

Here there are examples of some situations that can arise after cutting:

[IMAGE]

two holes two holes one hole one hole

Input

The input file consists of blocks of lines. Each block except the last describes one cutting process. In the first line of the block there is a number tex2html_wrap_inline33 indicating the number of segment cuts in the cutting process. These cuts are defined by the following *N* lines. The line defining one segment cut has the form tex2html_wrap_inline37 where tex2html_wrap_inline39 and tex2html_wrap_inline41 are the co-ordinates of the end points of the segment cut. They are separated by one space. The co-ordinates are integers and always define horizontal or vertical segment (i.e. segment parallel with *x* or *y* axis).

The last block consists of just one line containing 0.

Output

The output file contains the lines corresponding to the blocks in the input file. Each such line contains the number of holes that remain in the tin plate after the execution of the corresponding cuts.

There is no line in the output file corresponding to the last "null" block of the input file.

Example

Input:

```
4
0 1 1 1
1 1 1 0
1 0 0 0
0 0 0 1
2
0 1 2 1
```

```
1 2 1 0
0
```

Output:

```
1
0
```

Added by: Adrian Kuegel
Date: 2005-07-26
Time limit: 5s
Source limit: 50000B
Languages: All
Resource: ACM Central European Regional Contest 1996

SPOJ Problem Set (classical)

406. Logic

Problem code: LOGIC

Consider a 10x10 grid. Cells in this grid can contain one of five logic operations (AND, OR, NOT, Input, Output). These can be joined together to form a logic circuit. Given a description of a circuit and a set of boolean values, build the logic circuit and execute the input stream against it.

Input

The first line of the input contains a single integer *n*, which specifies the number of circuits to be processed. There will then be *n* groups of circuit descriptions and test values.

A circuit is made up of a number of operations. Each line describing an operation begins with three characters: the co-ordinates for a cell, 0-9 on the *X*-axis then 0-9 on the *Y*-axis, followed by a single character to represent the operation of that cell ('`&`' for AND, '`|`' for OR, '`!`' for NOT, '`i`' for Input and '`o`' for Output). Optionally following each triple is a set of co-ordinate pairs which represent the *x* and *y* co-ordinates of cells that take the output of this cells operation as an input for theirs. This (possibly empty) output list is terminated by '`..`'. The list of operations is terminated by a line containing the word '`end`'.

Next, for each circuit, comes the set of test values. The first line contains an integer *t* which gives the number of test cases your program must run. Next, there are *t* lines, each line containing a sequence of '`0`' and '`1`' characters symbolising the input values for one test case. The number of inputs will always correspond to the number of inputs defined by the circuit description. The input values are to be applied to the inputs in the order in which the input operations were defined in the circuit description.

The next circuit description, if any, will then follow.

Output

For each circuit, your program should output one line for each test case given in the input. The line should contain one '`0`' or '`1`' character for each output defined by the circuit description in the order in which the outputs were defined.

Your program should output a blank line after each set of test cases.

Example

Input:

```
1
00i 11 13 ..
02i 11 13 ..
11& 21 ..
21o ..
13| 23 ..
23o ..
end
```

```
4
00
01
10
11
```

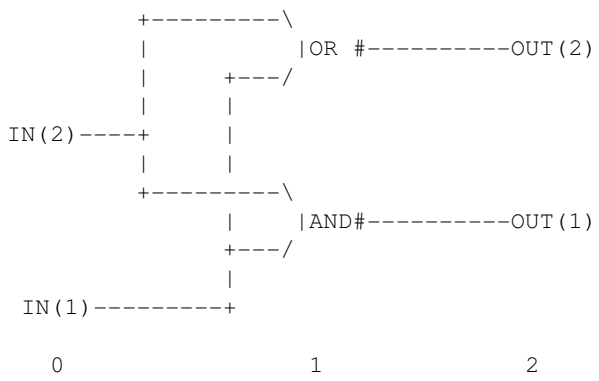
Output:

```
00
01
01
11
```

Notes:

- i, o and ! operations will always have exactly one input.
- & and | operations will always have exactly two inputs.
- Even if an operation can feed others, it does not have to.
- No recursive circuits.
- o can also be an input for another gate

Hint: Sample input specifies a circuit consiting of an 'AND' and an 'OR' operation in parallel both fed from the same two inputs:



In grid terms this is two inputs at 0,0 and 1,0. The first input feeds the AND operation at 1,1 and the OR operation at 1,3. The second input operation feeds the second input for the same AND and OR operations. The AND operation then feeds an output operation at 2,1. The OR operation also feeds an output operation, this one at 2,3.

Added by: Adrian Kuegel
Date: 2005-07-26
Time limit: 5s
Source limit:50000B
Languages: All
Resource: ACM Northwestern European Regional Contest 1993

SPOJ Problem Set (classical)

407. Random Number

Problem code: RNUMBER

A Black Box algorithm supposes that natural number sequence $u(1), u(2), dots, u(N)$ is sorted in non-descending order, $N le M$ and for each *p* ($1 le p le N$) an inequality $p le u(p) le M$ is valid.

Making tests for this algorithm we have met with the following problem. For setting a random sequence ${u(i)}$ a usual random data generator did not fit. As the sequence itself had been imposed certain restrictions, the method of choosing the next random element (in the interval defined by restrictions) did not give the random sequence as a whole.

We have come to a conclusion that the problem can be solved in the following way. If we arrange all possible sequences in certain order (for example, in lexicographical order) and assign each sequence its number, after choice of the random number it is possible to take the correspondent sequence for the random one. At the first glance it seems enough to make up a program generating all these sequences in such order. Alas! Even having not great values of *M* and *N* it would have taken any powerful modern computer centuries to enumerate all such sequences. It turned out it was possible to avoid generating all sequences if we managed to create required sequence according to its number immediately. But even this statement does not cover all. As the amount of sequences is quite large, the number can be a long one, composed of hundreds decimal digits, though our random data generator could give only normal numbers. We decided to produce a long random number from a real random number distributed in [0,1]. Namely, present the number in binary notation: $0.b(1)b(2)dots$, where all $b(i) = 0$ or 1. Let us set a regulation to associate such real number to an integer from [*A*,*B*] segment:

Formula

begin{displaymath}G(A,B,0.b_1b_2dots b_p) = left{ begin{array}{ll} A, & m... ...p, & mbox{if $b_1=1$} end{array} right. end{array} right. end{displaymath}

Here we suppose, that $A le B$, $p ge 0$, and ‘‘div 2" is an integer division by 2.

Let *M*, *N* ($1 le N le M le 200$) and a binary real number $0.b(1)b(2)dots b(p)$ ($1 le p le 400$) be given. Write a program to find out the corresponding $u(1), u(2), dots, u(N)$ sequence, i.e. to find a sequence with $G(1, T, 0.b(1)b(2)dots b(p))$ number in lexicographical order of all possible ${u(i)}$ for the given *M* and *N* (*T* is the quantity of such sequences). Numeration begins with 1. Keep in mind that in lexicographical order ${l(i)}$ proceeds ${h(i)}$ if after omitting equal beginnings, the first number of ${l(i)}$ tail is smaller than the first number or ${h(i)}$ tail. Following example illustrates the list of all possible sequences for $M = 4$ and $N = 3$ in lexicographical order.

A note (it does not concern the solution of this task):

The choice of random binary vector $0.b(1)b(2)dots b(p)$ does not give an absolute uniform random data generator if we use the Formula. However, taking into account the fact that [*A*,*B*] interval is big we shall obtain a distribution applicable in most cases.

Example

```
1, 2, 3
1, 2, 4
1, 3, 3
1, 3, 4
1, 4, 4
2, 2, 3
2, 2, 4
2, 3, 3
2, 3, 4
2, 4, 4
3, 3, 3
3, 3, 4
3, 4, 4
4, 4, 4
```

(here *T*=14)

Input

The first line of the input is an integer K <= 10, followed by K datasets.

The first line of each dataset contains *M* and *N*. The second line contains binary real number $0.b(1)b(2)dots b(p)$ (without leading, trailing and other spaces).

Output

For each dataset, write into the output data file the corresponding sequence $u(1), u(2), dots, u(N)$. The sequence numbers should be separated with spaces and end-of-line characters. There should be up to 20 numbers in each line. If neccesary, the numbers will have leading blanks to occupy 3 characters.

Example

Input:

```
1
4 3
0.01101101011110010001101010001011010
```

Output:

```
  2   2   4
```

Added by: Adrian Kuegel
Date: 2005-07-26
Time limit: 5s
Source limit:50000B
Languages: All
Resource: ACM Northeastern European Regional Contest 1996

SPOJ Problem Set (classical)

408. Jill Rides Again

Problem code: JRIDE

Jill likes to ride her bicycle, but since the pretty city of Greenhills where she lives has grown, Jill often uses the excellent public bus system for part of her journey. She has a folding bicycle which she carries with her when she uses the bus for the first part of her trip. When the bus reaches some pleasant part of the city, Jill gets off and rides her bicycle. She follows the bus route until she reaches her destination or she comes to a part of the city she does not like. In the latter event she will board the bus to finish her trip.

Through years of experience, Jill has rated each road on an integer scale of niceness. Positive niceness values indicate roads Jill likes; negative values are used for roads she does not like. There are not zero values. Jill plans where to leave the bus and start bicycling, as well as where to stop bicycling and re-join the bus, so that the sum of niceness values of the roads she bicycles on is maximized. This means that she will sometimes cycle along a road she does not like, provided that it joins up two other parts of her journey involving roads she likes enough to compensate. It may be that no part of the route is suitable for cycling so that Jill takes the bus for its entire route. Conversely, it may be that the whole route is so nice Jill will not use the bus at all.

Since there are many different bus routes, each with several stops at which Jill could leave or enter the bus, she feels that a computer program could help her identify the best part to cycle for each bus route.

Input

The input file contains information on several bus routes. The first line of the file is a single integer *b* representing the number of route descriptions in the file. The identifier for each route (*r*) is the sequence number within the data file, 1 <= r <= b. Each route description begins with the number of stops on the route: an integer *s*, 2 <= s <= 100000 on a line by itself. The number of stops is followed by *s* - 1 lines, each line *i* (1 <= i < s) is an integer n_i with absolute value <= 1000 representing Jill's assessment of the niceness of the road between the two stops *i* and *i*+1.

Output

For each route in the input file, your program should identify the beginning bus stop *i* and the ending bus stop *j* that identify the segment of the route which yields the maximal sum of niceness, $m = n_i + n_{i+1} + ... + n_{j-1}$. If more than one segment is maximally nice, choose the one with the longest cycle ride (largest *j*-*i*). To break ties in longest maximal segments, choose the segment that begins with the earliest stop (lowest *i*). For each route *r* in the input file, print a line in the form:

`The nicest part of route` *r* `is between stops` *i* `and` *j*

However, if the maximal sum is not positive, your program should print:

Route *r* has no nice parts

Example

Input:

```
3
3
-1
6
10
4
-5
4
-3
4
4
-4
4
-5
4
-2
-3
-4
```

Output:

```
The nicest part of route 1 is between stops 2 and 3
The nicest part of route 2 is between stops 3 and 9
Route 3 has no nice parts
```

Added by: Adrian Kuegel
Date: 2005-07-27
Time limit: 4s
Source limit: 50000B
Languages: All
Resource: ACM ICPC World Finals 1997

SPOJ Problem Set (classical)

409. DEL Command

Problem code: DELCOMM

It is required to find out whether it is possible to delete given files from MS-DOS directory executing the DEL command of MS-DOS operation system only once. There are no nested subdirectories.

A note

DEL command has the following format: `DEL` *wildcard*

The actual wildcard as well as a full file name can be made up either of a name containing 1 up to 8 characters or of a name and extension, containing up to 3 characters. The point character '.' separates the extension from the file name. The extension can be empty and this is equivalent to a name without any extension (in this case a wildcard ends with a point). In a wildcard the characters '?' and '*' can be used. A question mark substitutes exactly one character of the full file name excluding a point, an asterisk any sequence of characters (containing no points) even empty one. An asterisk can appear only at the last position of the name and the extension.

MS-DOS system can permit maybe other wildcards but they can not be used in this task. File names and extensions consist only of Latin capitals and digits.

Input

The first line of the input is an integer M, then a blank line followed by M datasets. There is a blank line between datasets.

Input data for each dataset contains a list of full file names without empty lines and spaces. Each name is written in a separate line of input data file and preceded with a control sign: '-' for delete or '+' for keep. Full file names are not repeated. The list comprises at least one file, and at least one file is marked to be deleted. There are no more than 1000 files.

Output

For each dataset, write to the first line of output the required DEL command (only one proposal) or `IMPOSSIBLE` if there is no solution. A space should separate "`DEL`" from wildcard. **Print a blank line between datasets.**

Example

Input:

```
1

-BP.EXE
-BPC.EXE
+TURBO.EXE
```

Output:

```
DEL ?P*.*
```

Added by: Adrian Kuegel
Date: 2005-07-27
Time limit: 5s
Source limit: 50000B
Languages: All
Resource: ACM Northeastern European Regional Contest 1996

SPOJ Problem Set (classical)

410. Variable Radix Huffman Encoding

Problem code: VHUFFM

Huffman encoding is a method of developing an optimal encoding of the symbols in a *source alphabet* using symbols from a *target alphabet* when the frequencies of each of the symbols in the source alphabet are known. Optimal means the average length of an encoded message will be minimized. In this problem you are to determine an encoding of the first *N* uppercase letters (the source alphabet, tex2html_wrap_inline87 through tex2html_wrap_inline89 , with frequencies tex2html_wrap_inline91 through tex2html_wrap_inline93) into the first *R* decimal digits (the target alphabet, tex2html_wrap_inline97 through tex2html_wrap_inline99).

Consider determining the encoding when *R*=2. Encoding proceeds in several passes. In each pass the two source symbols with the lowest frequencies, say tex2html_wrap_inline87 and tex2html_wrap_inline105 , are grouped to form a new ‘‘combination letter" whose frequency is the sum of tex2html_wrap_inline91 and tex2html_wrap_inline109 . If there is a tie for the lowest or second lowest frequency, the letter occurring earlier in the alphabet is selected. After some number of passes only two letters remain to be combined. The letters combined in each pass are assigned one of the symbols from the target alphabet.

The letter with the lower frequency is assigned the code 0, and the other letter is assigned the code 1. (If each letter in a combined group has the same frequency, then 0 is assigned to the one earliest in the alphabet. For the purpose of comparisons, the value of a ‘‘combination letter" is the value of the earliest letter in the combination.) The final code sequence for a source symbol is formed by concatenating the target alphabet symbols assigned as each combination letter using the source symbol is formed.

The target symbols are concatenated in the reverse order that they are assigned so that the first symbol in the final code sequence is the last target symbol assigned to a combination letter.

The two illustrations below demonstrate the process for *R*=2.

tabular23

When *R* is larger than 2, *R* symbols are grouped in each pass. Since each pass effectively replaces *R* letters or combination letters by 1 combination letter, and the last pass must combine *R* letters or combination letters, the source alphabet must contain $k*(R-1)+R$ letters, for some integer k.

Since *N* may not be this large, an appropriate number of fictitious letters with zero frequencies must be included. These fictitious letters are not to be included in the output. In making comparisons, the fictitious letters are later than any of the letters in the alphabet.

Now the basic process of determining the Huffman encoding is the same as for the *R*=2 case. In each pass, the *R* letters with the lowest frequencies are grouped, forming a new combination letter with a frequency equal to the sum of the letters included in the group. The letters that were grouped are assigned the target alphabet symbols 0 through *R*-1. 0 is assigned to the letter in the combination with the lowest frequency, 1 to the next lowest frequency, and so forth. If several of the letters in the group have the same frequency, the one earliest in the alphabet is assigned the smaller target symbol, and so

forth.

The illustration below demonstrates the process for R=3.

tabular63

Input

The input will contain one or more data sets, one per line. Each data set consists of an integer value for R (between 2 and 10), an integer value for N (between 2 and 26), and the integer frequencies tex2html_wrap_inline91 through tex2html_wrap_inline93 , each of which is between 1 and 999.

The end of data for the entire input is the number 0 for R; it is not considered to be a separate data set.

Output

For each data set, display its number (numbering is sequential starting with 1) and the average target symbol length (rounded to two decimal places) on one line. Then display the N letters of the source alphabet and the corresponding Huffman codes, one letter and code per line.
Print a blank line after each test case.
The examples below illustrate the required output format.

Example

Input:

```
2 5 5 10 20 25 40
2 5 4 2 2 1 1
3 7 20 5 8 5 12 6 9
4 6 10 23 18 25 9 12
0
```

Output:

```
Set 1; average length 2.10
    A: 1100
    B: 1101
    C: 111
    D: 10
    E: 0

Set 2; average length 2.20
    A: 11
    B: 00
    C: 01
    D: 100
    E: 101

Set 3; average length 1.69
    A: 1
    B: 00
    C: 20
    D: 01
    E: 22
    F: 02
    G: 21
```

```
Set 4; average length 1.32
    A: 32
    B: 1
    C: 0
    D: 2
    E: 31
    F: 33
```

Added by: Adrian Kuegel
Date: 2005-07-27
Time limit: 2s
Source limit: 50000B
Languages: All
Resource: ACM ICPC World Finals 1995

SPOJ Problem Set (classical)

411. Number of quite different words

Problem code: NUMQDW

Let's consider the alphabet consisting of the first **c** roman uppercase letters, i.e. {A, B, C, D, E, F} if **c** is 6.
We will call two words *quite different*, if there is no common subsequence of length more than one between those two words. For example ABC and CBA are quite different, but ABBA and CADDCAD aren't, because AA is a subsequence of both words.
Given a word **w** you are to find the number of words of length **n** that are quite different from **w**.

Input

The first line will contain the number of test cases (at most 20). Then there will be pairs of lines, the first one containing the numbers **n** (**n** will fit into a 32-bit signed integer and will be non-negative) and **c** (1 <= **c** <= 6), the second one the word **w**. **w** will only consist of the first **c** letters of the roman alphabet and will have at most 10000 characters.

Output

Print one line for each test case, consisting only of the number of words that are quite different from **w**. As this number can be quite large, you just have to output its remainder when dividing by 4242.

Example

Input:
```
3
3 3
ABC
4 4
CADDCAD
100 3
A
```

Output:
```
10
13
2223
```

Added by: Robin Nittka
Date: 2005-08-04
Time limit: 20s
Source limit: 50000B
Languages: All
Resource: self-invented

SPOJ Problem Set (classical)

412. K-path cover

Problem code: COVER

Problem

K-path cover of a directed graph is a set of exactly k of its edges chosen in such way that every two of them have different start vertices and every two of them have different end vertices. Assuming that for each vertex we know its cost we can define cost of the edge as a sum of costs of its start and end. We can also define cost of a k-path cover as a sum of costs of its edges. Your task is to find cheapest k-path cover for given directed graph with known costs of the vertices.

[IMAGE]
A graph and its cheapest 4-path cover.

Input

First line of input contains number of test cases c (1<=c<=20). Each test case begins with k, number of vertices n and number of edges m (1<=k<=100, 1<=n<=10000, 0<=m<=1000000). Next n lines contain costs of the vertices, each of them is an integer from [-100000,100000]. Then m lines describing edges follow, each of them containing exactly two numbers representing its start and end vertices. Vertices are numbered from 1 to n.

Output

For each test case output cost of the cheapest k-path cover. When given graph has no k-path cover output NONE.

Example

```
Input:
1
4 6 9
5
4
6
10
2
3
1 2
1 4
2 4
3 2
4 3
5 4
6 3
5 6
```

```
6 5

Output:
33
```

Added by: Pawel Gawrychowski
Date: 2005-08-08
Time limit: 10s
Source limit: 50000B
Languages: All
Resource: ONTAK 05

SPOJ Problem Set (classical)

413. Word Puzzles

Problem code: WPUZZLES

Word puzzles are usually simple and very entertaining for all ages. They are so entertaining that Pizza-Hut company started using table covers with word puzzles printed on them, possibly with the intent to minimise their client's perception of any possible delay in bringing them their order.

Even though word puzzles may be entertaining to solve by hand, they may become boring when they get very large. Computers do not yet get bored in solving tasks, therefore we thought you could devise a program to speedup (hopefully!) solution finding in such puzzles.

The following figure illustrates the PizzaHut puzzle. The names of the pizzas to be found in the puzzle are: MARGARITA, ALEMA, BARBECUE, TROPICAL, SUPREMA, LOUISIANA, CHEESEHAM, EUROPA, HAVAIANA, CAMPONESA.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
0	Q	W	S	P	I	L	A	A	T	I	R	A	G	R	A	M	Y	K	E	I
1	A	G	T	R	C	L	Q	A	X	L	P	O	I	J	L	F	V	B	U	Q
2	T	Q	T	K	A	Z	X	V	M	R	W	A	L	E	M	A	P	K	C	W
3	L	I	E	A	C	N	K	A	Z	X	K	P	O	T	P	I	Z	C	E	O
4	F	G	K	L	S	T	C	B	T	R	O	P	I	C	A	L	B	L	B	C
5	J	E	W	H	J	E	E	W	S	M	L	P	O	E	K	O	R	O	R	A
6	L	U	P	Q	W	R	N	J	O	A	A	G	J	K	M	U	S	J	A	E
7	K	R	Q	E	I	O	L	O	A	O	Q	P	R	T	V	I	L	C	B	Z
8	Q	O	P	U	C	A	J	S	P	P	O	U	T	M	T	S	L	P	S	F
9	L	P	O	U	Y	T	R	F	G	M	M	L	K	I	U	I	S	X	S	W
10	W	A	H	C	P	O	I	Y	T	G	A	K	L	M	N	A	H	B	V	A
11	E	I	A	K	H	P	L	B	G	S	M	C	L	O	G	N	G	J	M	L
12	L	D	T	I	K	E	N	V	C	S	W	Q	A	Z	U	A	O	E	A	L
13	H	O	P	L	P	G	E	J	K	M	N	U	T	I	I	O	R	M	N	C
14	L	O	I	U	F	T	G	S	Q	A	C	A	X	M	O	P	B	E	I	O
15	Q	O	A	S	D	H	O	P	E	P	N	B	U	Y	U	Y	O	B	X	B
16	I	O	N	I	A	E	L	O	J	H	S	W	A	S	M	O	U	T	R	K
17	H	P	O	I	Y	T	J	P	L	N	A	Q	W	D	R	I	B	I	T	G
18	L	P	O	I	N	U	Y	M	R	T	E	M	P	T	M	L	M	N	B	O
19	P	A	F	C	O	P	L	H	A	V	A	I	A	N	A	L	B	P	F	S

Problem

Your task is to produce a program that given the word puzzle and words to be found in the puzzle, determines, for each word, the position of the first letter and its orientation in the puzzle.

You can assume that the left upper corner of the puzzle is the origin, (0,0). Furthemore, the orientation of the word is marked clockwise starting with letter `A` for north (note: there are 8 possible directions in total).

Input

The first line of the input contains a number T <= 10 which indicates the number of test cases to follow. Each test case starts with a line consisting of three positive numbers: The number of lines of the word puzzle, 0 < L <= 1000, the number of columns, 0 < C <= 1000, and the number of words to be found, 0 < W <= 1000. The following L input lines, each consisting of C uppercase letters, contain the word puzzle. Then at last the W words are input one per line. You can assume that each word can be found exactly once in the word puzzle.

Output

For each test case your program should output W lines: For each word (using the same order as the words were input) print a triplet defining the coordinates, line and column, where the first letter of the word appears, followed by a letter indicating the orientation of the word according to the rules defined above. Each value in the triplet must be separated by one space only.
Print one blank line between test cases.

Example

Input:

```
1
20 20 10
QWSPILAATIRAGRAMYKEI
AGTRCLQAXLPOIJLFVBUQ
TQTKAZXVMRWALEMAPKCW
LIEACNKAZXKPOTPIZCEO
FGKLSTCBTROPICALBLBC
JEWHJEEWSMLPOEKORORA
LUPQWRNJOAAGJKMUSJAE
KRQEIOLOAOQPRTVILCBZ
QOPUCAJSPPOUTMTSLPSF
LPOUYTRFGMMLKIUISXSW
WAHCPOIYTGAKLMNAHBVA
EIAKHPLBGSMCLOGNGJML
LDTIKENVCSWQAZUAOEAL
HOPLPGEJKMNUTIIORMNC
LOIUFTGSQACAXMOPBEIO
QOASDHOPEPNBUYUYOBXB
IONIAELOJHSWASMOUTRK
HPOIYTJPLNAQWDRIBITG
LPOINUYMRTEMPTMLMNBO
PAFCOPLHAVAIANALBPFS
MARGARITA
ALEMA
BARBECUE
```

```
TROPICAL
SUPREMA
LOUISIANA
CHEESEHAM
EUROPA
HAVAIANA
CAMPONESA
```

Output:

```
0 15 G
2 11 C
7 18 A
4 8 C
16 13 B
4 15 E
10 3 D
5 1 E
19 7 C
11 11 H
```

Added by: Adrian Kuegel
Date: 2005-08-10
Time limit: 15s
Source limit: 50000B
Languages: All
Resource: ACM Southwestern European Regional Contest 2002

SPOJ Problem Set (classical)

414. Equatorial Bonfire

Problem code: BONFIRE

Some great ideas are never implemented. This was the case with the equatorial bonfire planned for the millenial celebration. Maybe the plan will be rediscovered for the next turn of millenia. Before it is completely forgotten we will tell you about it: The idea was to put tarred logs and gun powder contiguously along the equator and then ignite bonfires at various points of this gun powder belt at various times. The fire would spread in both directions along the equator and in the end, the whole equator would burn.

One concern of the architects of this celebration was what would be the last place to catch fire and when would it happen?

Input

The input file consists of at most 10 blocks, each specifying a separate proposal for the equatorial bonfire. The first line of each block specifies the speed of fire's advance in the degrees of longitude per hour. The next line contains the number of bonfires **N <= 5000** along the equator. Each of the next **N** lines contains two numbers specifying the time and the location of a bonfire.

The time is given in hours from 12:00am GMT (all times are positive), the location is given in the degrees of longitude (greater than -180 and less than or equal to 180). All numbers except **N** are given with precision of at most 2 decimal places. Every block is followed by an empty line. The last block is followed by an empty line and then a line containing `-1`.

Output

For every block in the input file, output a single line containing the time and coordinate of the last place to catch fire. The time and the coordinate should be in the same units and from the same range as input, but with precision of 3 decimal places. If there are multiple solutions, output the one which is the closest if you travel from the zero meridian east (i.e. in the positive direction).

Example

Input:

```
2
2
1 90
1 -90

10.0
3
1 40
2 45
6 -80

-1
```

Output:

```
46.000 0.000
15.500 -175.000
```

Note that in the second case, the fire at 45 degrees actually starts 1.5 hours after 12:00am. This of course does does not prevent anybody from igniting it again 2 hours after 12:00am.

Added by: Adrian Kuegel
Date: 2005-08-11
Time limit: 10s
Source limit: 50000B
Languages: All
Resource: IPSC 2002

SPOJ Problem Set (classical)

416. Divisibility by 15

Problem code: DIV15

There is a string containing only decimal digit characters. The length of the string is between *1* and *1000*. Using characters of the string, you have to construct the maximum number which divides by fifteen without remainder. Each character of the string may not be used more than once.

Input

First line of input contains an integer **t** (*1* <= **t** <= *90*), equal to the number of testcases. Then descriptions of **t** testcases follow.

Each testcase is described in a single line representing the source string.

Output

For each testcase output one line with the decimal representation of the maximum number. Leading zeroes should be omitted. If no number can be constructed, output a single word *"impossible"*.

Example

Input:
```
1
02041
```

Output:
```
4200
```

Added by: Ivan Metelsky
Date: 2005-08-25
Time limit: 2s
Source limit: 50000B
Languages: All
Resource: NEERC Western Subregion QF 2004

SPOJ Problem Set (classical)

417. The lazy programmer

Problem code: LAZYPROG

A new web-design studio, called SMART (Simply Masters of ART), employs two people. The first one is a web-designer and an executive director at the same time. The second one is a programmer. The director is so a nimble guy that the studio has already got **N** contracts for web site development. Each contract has a deadline $\mathbf{d_i}$.

It is known that the programmer is lazy. Usually he does not work as fast as he could. Therefore, under normal conditions the programmer needs $\mathbf{b_i}$ of time to perform the contract number **i**. Fortunately, the guy is very greedy for money. If the director pays him $\mathbf{x_i}$ dollars extra, he needs only $\mathbf{(b_i - a_i * x_i)}$ of time to do his job. But this extra payment does not influence other contracts. This means that each contract should be paid separately to be done faster. The programmer is so greedy that he can do his job almost instantly if the extra payment is $\mathbf{(b_i / a_i)}$ dollars for the contract number **i**.

The director has a difficult problem to solve. He needs to organize programmer's job and, may be, assign extra payments for some of the contracts so that all contracts are performed in time. Obviously he wishes to minimize the sum of extra payments. Help the director!

Input

First line of the input contains an integer **t** (*1* <= **t** <= *45*), equal to the number of testcases. Then descriptions of **t** testcases follow.

First line of description contains the number of contracts **N** (*1* <= **N** <= *100000*, integer). Each of the next **N** lines describes one contract and contains integer numbers $\mathbf{a_i}$, $\mathbf{b_i}$, $\mathbf{d_i}$ (*1* <= $\mathbf{a_i}$, $\mathbf{b_i}$ <= *10000*; *1* <= $\mathbf{d_i}$ <= *1000000000*) separated by spaces.

At least *90%* of testcases will have *1* <= **N** <= *10000*.

Output

For each testcase in the input your program should output one line with a single real number S. Here S is the minimum sum of money which the director needs to pay extra so that the programmer could perform all contracts in time. The number must have two digits after the decimal point.

Example

Input:
```
1
2
20 50 100
10 100 50
```

Output:
```
5.00
```

Added by: Ivan Metelsky
Date: 2005-08-25
Time limit: 10s
Source limit:50000B
Languages: All
Resource: NEERC Western Subregion QF 2004

SPOJ Problem Set (classical)

418. Necklace

Problem code: NECKLACE

There are **N** points marked on a surface, pair ($\mathbf{x_i}$, $\mathbf{y_i}$) is coordinates of a point number **i**. Let's call a *necklace* a set of **N** figures which fulfills the following rules.

- The figure **#i** consists of all such points (**x**, **y**) that $(\mathbf{x} - \mathbf{x_i})^2 + (\mathbf{y} - \mathbf{y_i})^2 <= \mathbf{r_i}^2$, where $\mathbf{r_i} >= 0$.
- Figures **#i** and **#(i+1)** intersect ($1 <= \mathbf{i} < \mathbf{N}$).
- Figures *#1* and **#N** intersect.
- All the rest pairs of figures do not intersect.

Write a program which takes points and constructs a necklace.

Input

First line of input contains an integer **t** ($1 <= \mathbf{t} <= 45$), equals to the number of testcases. Then descriptions of **t** testcases follow.

The first line of the description contains one integer number **N** ($2 <= \mathbf{N} <= 100$). Each of the next **N** lines contains two real numbers $\mathbf{x_i}$, $\mathbf{y_i}$ ($-1000 <= \mathbf{x_i}, \mathbf{y_i} <= 1000$), separated by one space. It is guaranteed that at least one necklace exists for each testcase.

Output

For each testcase your program should output exactly **N** lines. A line **#i** should contain real number $\mathbf{r_i}$ ($0 <= \mathbf{r_i} < 10000$). To avoid potential accuracy problems, a checking program uses the following rules.

- Figures **#i** and **#j** definitely do not intersect if $\mathbf{r_i} + \mathbf{r_j} <= \mathbf{d_{ij}} - 10^{-5}$.
- Figures **#i** and **#j** definitely intersect if $\mathbf{d_{ij}} + 10^{-5} <= \mathbf{r_i} + \mathbf{r_j}$.
- The case when $\mathbf{d_{ij}} - 10^{-5} < \mathbf{r_i} + \mathbf{r_j} < \mathbf{d_{ij}} + 10^{-5}$ is decided in favour of a contestant.
- $\mathbf{d_{ij}}$ equals $sqrt((\mathbf{x_i} - \mathbf{x_j})^2 + (\mathbf{y_i} - \mathbf{y_j})^2)$ in the rules above.

Example

Input:

```
1
4
0 0
10 0
10 10
0 10
```

Output:

7
7
7
7

Added by: Ivan Metelsky
Date: 2005-08-25
Time limit: 10s
Source limit: 50000B
Languages: All
Resource: NEERC Western Subregion QF 2004

SPOJ Problem Set (classical)

419. Transposing is Fun

Problem code: TRANSP

Problem

Suppose you are given a $2^a x 2^b$ array. It is stored sequentially in memory in the usual way, first values in the first row, then values in the second one and so on. You would like to transpose it, but you don't have any additional memory. The only operation that you can perform is swapping contents of two memory cells. What is the minimal number of such operations required for transpostion?

Input

The first line of input contains the number of test cases c (1<=c<=100). Each test case consists of two integers a,b (0<=a+b<=500000).

Output

For each test case output the minimal number of swaps required to transpose an $2^a x 2^b$ array. As it can be quite large, you have to output its remainder when divided by 1000003 (yes, it's a prime number :).

Example

```
Input:
3
1 1
2 2
5 7

Output:
1
6
3744
```

Added by: Pawel Gawrychowski
Date: 2005-09-03
Time limit: 30s
Source limit: 50000B
Languages: All

SPOJ Problem Set (classical)

421. Another Road Problem

Problem code: AROAD

Problem

Let's say you are given a set of cities (numbered from 1 to n) and possible bidirectional roads between them. You would like to build cheapest road network that will make getting from the capital (which has number 1) to every other city possible, where the cost of the network is just sum of its roads' costs. Seems easy? Well, it certainly would be too easy and boring, so this time you should satisfy one additional constraint: you must consider only networks in which there are at most d roads outgoing from the capital.

Input

First line of input contains number of test cases c (1<=c<=40). Each test case begins with number of cities n, number of possible roads m and maximum degree d (1<=n<=1000, 0<=m<=100000, 0<=d<=100). Then m lines describing roads follow, each of them containing road endpoints x,y and its cost c (1<=x,y<=n, 0<=c<=10000).

Output

For each test case output the cost of building cheapest road network or NONE if it is impossible.

Example

```
Input:
4
4 5 0
1 2 1
1 3 1
1 4 2
2 3 2
3 4 1000

4 5 1
1 2 1
1 3 1
1 4 2
2 3 2
3 4 1000

4 5 2
1 2 1
1 3 1
1 4 2
2 3 2
3 4 1000

4 5 3
```

```
1 2 1
1 3 1
1 4 2
2 3 2
3 4 1000

Output:
NONE
1003
5
4
```

Added by: Pawel Gawrychowski
Date: 2005-10-07
Time limit: 6s
Source limit: 50000B
Languages: All

SPOJ Problem Set (classical)

422. Transposing is Even More Fun

Problem code: TRANSP2

Problem

Suppose you are given a 2ax2b array. It is stored sequentially in memory in the usual way, first values in the first row, then values in the second one and so on. You would like to transpose it, but you don't have any additional memory. The only operation that you can perform is swapping contents of two memory cells. What is minimal number of such operations required for transpostion?

Input

First line of input contains number of test cases c (1<=c<=400000). Each test case consists of two integers a,b (0<=a+b<=1000000).

Output

For each test case output minimal number of swaps required to transpose an 2ax2b array. As it can be quite large, you have to output its remainder when divided by 1000003 (yes, it's a prime number :).

Example

```
Input:
3
1 1
2 2
5 7

Output:
1
6
3744
```

Added by: Pawel Gawrychowski
Date: 2005-10-08
Time limit: 8s
Source limit:50000B
Languages: All

SPOJ Problem Set (classical)

423. Assignments

Problem code: ASSIGN

Problem

Your task will be to calculate number of different assignments of n different topics to n students such that everybody gets exactly one topic he likes.

Input

First line of input contains number of test cases c (1<=c<=80). Each test case begins with number of students n (1<=n<=20). Each of the next n lines contains n integers describing preferences of one student. 1 at the ith position means that this student likes ith topic, 0 means that he definitely doesn't want to take it.

Output

For each test case output number of different assignments (it will fit in a signed 64-bit integer).

Example

```
Input:
3
3
1 1 1
1 1 1
1 1 1
11
1 0 0 1 0 0 0 0 0 1 1
1 1 1 1 1 0 1 0 1 0 0
1 0 0 1 0 0 1 1 0 1 0
1 0 1 1 1 0 1 1 0 1 1
0 1 1 1 0 1 0 0 1 1 1
1 1 1 0 0 1 0 0 0 0 0
0 0 0 0 1 0 1 0 0 0 1
1 0 1 1 0 0 0 0 0 0 1
0 0 1 0 1 1 0 0 0 1 1
1 1 1 0 0 0 1 0 1 0 1
1 0 0 0 1 1 1 1 0 0 0
11
0 1 1 1 0 1 0 0 0 1 0
0 0 1 1 1 1 1 1 1 1 1
1 1 0 1 0 0 0 0 0 1 0
0 1 0 1 0 1 0 1 0 1 1
1 0 0 1 0 0 0 0 1 0 1
0 0 1 0 1 1 0 0 0 0 1
1 0 1 0 1 1 1 0 1 1 0
1 0 1 1 0 1 1 0 0 1 0
0 0 1 1 0 1 1 1 1 1 1
0 1 0 0 0 0 0 0 0 1 1
```

```
0 1 1 0 0 0 0 0 1 0 1

Output:
6
7588
7426
```

Added by: Pawel Gawrychowski
Date: 2005-10-08
Time limit: 20s
Source limit: 50000B
Languages: All

SPOJ Problem Set (classical)

425. Kill evil instantly

Problem code: HAJIME

This problem tests your knowledge of the C programming language. Your task is to submit a snippet of C code that consists of two declarations defining a type called "zan", which should be a struct containing two members: first an unsigned int called "aku", then a constant pointer to char called "soku".

To make things more interesting, you can't use any whitespace in either declaration, and the two declarations must be sufficiently dissimilar (basically, you have to use two different tricks to get around the lack of whitespace).

Input

There is no input.

Output

Your submission should consist of exactly two declarations as described above, separated by whitespace.

Update: "Exactly two" means exactly two. Your code isn't allowed to define any other types; anything containing `struct foo` or `typedef unsigned int` is rejected.

"Whitespace" includes newlines. NUL ('\0') is not whitespace, but it isn't a valid token separator either.

Example

Output:

```
typedef:struct{unsigned*aku;char*soku;}zan;
typedef:struct{unsigned*aku;char*soku;}zan;
```

This example is invalid for the following reasons:

- `typedef:` is a syntax error
- `aku` and `soku` have the wrong type
- the two declarations are too similar

Added by: Lukas Mai
Date: 2005-10-17
Time limit: 20s
Source limit:512B
Languages: TEXT

SPOJ Problem Set (classical)

428. Particular Palindromes

Problem code: PARTPALI

A palindromic decimal integer reads the same forward and backward. For example, the following numbers are palindromic.

6, 55, 282, 5005, 78187, 904409, 3160613, 11111111

Palindromic integers are plentiful. In fact, any integer not divisible by 10 has an infinite number of multiples that are palindromic. (The standard representation of a nonzero multiple of 10 cannot be palindromic since its reversal would have a leading 0.)

Write a program to determine, for a given positive integer, how many of its positive multiples are palindromes of a given length.

Input

The first line of the input will specify an integer n indicating the number of problem instances to follow, one to a line. Each of the ensuing n lines will specify a pair of positive integers m,s separated by a single space, with 1 < m < 1000, s < 20. (For m,s in this range, there are fewer than 2^32 palindromes among the s-digit multiples of m.) Each line will terminate with an end-of-line.

Output

The output should indicate for each m,s, exactly how many s-digit positive palindromes are divisible by m, with one problem instance per line.

Example

Input:

```
5
3 1
25 3
12 4
30 3
81 6
```

Output:

```
3
2
7
0
0
```

Explanation: There are three positive 1-digit multiples of 3, namely, 3, 6, and 9; all 1-digit numbers are trivially palindromes. Among the 3-digit palindromes, 525 and 575 are multiples of 25. The 4-digit multiples of 12 that are palindromes are 2112, 2772, 4224, 4884, 6336, 6996, 8448. There are no

positive palindromic numbers ending in 0 (since we do not allow leading 0's). No 6-digit palindromes are divisible by 81.

Added by:	Sebastian Kanthak
Date:	2005-10-26
Time limit:	5s
Source limit:	50000B
Languages:	All
Resource:	ACM Pacific Northwest Regional Contest 2003

SPOJ Problem Set (classical)

449. Simple Numbers with Fractions Conversion

Problem code: TCNUMFL

Every integer number n is represented in positional number system of base r by a sequence of digits 0 <= d_i < r, decimal point ',' and fractional part, so the value is equal to:

$$n = d_0 + r * d_1 + r^2 * d_2 + r^3 * d_3 + ... + r^{-1} * d_{-1} + r^{-2} * d_{-2} + r^{-3} * d_{-3} + ...$$

Your task is to convert a given number in r-base represantation into s-base representation with l digits after decimal point (no rounding - use floor), for example: decimal 231,5 into binary 11100111,1 with one digit after decimal point. Assume that r <= 36 and the digits are 0,1,2,3,4,5,6,7,8,9, *A*, *B*, *C*, *D*, *E*, *F*, *G*, *H*, *I*, *J*, *K*, *L*, *M*, *N*, *O*, *P*, *Q*, *R*, *S*, *T*, *U*, *V*, *W*, *X*, *Y*, *Z*.

Input

N [the number of series <= 1000]
n r s l [$n < 36^{1000} + 1$, r,s <= 36, l <= 1000]

Output

n [*s*-base representation of number *n*]

Text grouped in [ ] does not appear in the input and output file.

Example

Input:
```
10
500,1 6 31 3
3866,DJ 22 27 1
EH75,L3 24 4 3
A73C,10B 13 27 2
6C6J,E483 22 6 2
JA,L 30 5 4
6,5A 20 31 2
1,C5 14 7 1
HD,6K 26 9 2
1001,011 2 10 3
```

Output:
```
5P,555
1M8H,H
301223231,320
14MB,25
1255211,35
4310,3222
6,8G
1,6
555,23
9,375
```

Added by: Piotr Piotrowski
Date: 2004-11-08
Time limit: 13s
Source limit: 50000B
Languages: All except: JAR

SPOJ Problem Set (classical)

515. Collatz

Problem code: CLTZ

Let N be a positive integer, Consider the following recurrence: f(1) = N and f(K) = (0.5 + 2.5 * (f(K-1) mod 2)) * f(K-1) + (f(K-1) mod 2) if K>1. For a given N you have to compute the smallest L for which f(L)=1 (such an L always exists for N's in the input).

Input

Each line contains a positive integer N in decimal notation. You can be sure that N and all intermediate results are not bigger than 10^1888. Input terminated by EOF.

Output

For each number N in the input print one line with the value of L in decimal notation.

Example

Input:

```
1
2
321
1111111111111
11111111111111111111111111111111111111111111111111111111111111111111111111111
```

Output:

```
1
2
25
261
1296
```

Added by: Csaba Noszaly
Date: 2005-04-25
Time limit: 8s
Source limit: 18000B
Languages: All except: C99 strict
Resource: Folklore

SPOJ Problem Set (classical)

518. Zig-Zag Permutation

Problem code: ZZPERM

In the following we will deal with nonempty words consists only of lower case letters ’a’,’b’,..., ’j’ and we will use the natural ’a’ < ’b’ < ... < ’j’ ordering. Your task is to write a program that generates almost all zig-zag words (zig-zag permutations) from a given collection of letters. We say that a word W=W(1)W(2)...W(n) is zig-zag iff n = 1 or W(i) > W(i+1) and W(j) < W(j+1) for all odd 0 < i < n and for all even 0 < j < n or W(i) > W(i+1) and W(j) < W(j+1) for all even 0 < i < n and for all odd 0 < j < n. For example: "aabcc" is not zig-zag, "acacb" is zig-zag, "cac" is zig-zag, "abababc" is not zig-zag. If you imagine all possible zig-zag permutations of a word in increasing lexicographic order, you can assign a serial number (rank) to each one. For example: the word "aabcc" generates the sequence: 1 <-> "acacb", 2 <-> "acbca", 3 <-> "bacac", 4 <-> "bcaca", 5 <-> "cabac", 6 <-> "cacab".

Input

The input file consists several test cases. Each case contains a word (W) not longer than 64 letters and one positive number (D). The letters of each word are in increasing order. Input terminated by EOF.

Output

For each case in the input file, the output file must contain all of the zig-zag permutations of W whose zig-zag serial is divisible by D, in increasing lexicographic order - one word per line. In the next line you have to print the total number of zig-zag permutations of W. There is no case that produces more than 365 lines of output. Print an empty line after each case.

Example

Input:

```
j 1
abc 2
aaabc 1
aaabb 2
aaaaaaaaaaaaaaaaabbbbbbbbbbbbbbbcccdd 123456
```

Output:

```
j
1

bac
cab
4

abaca
acaba
2
```

```
1

babacbcabacabadababababababababababadab
213216
```

Added by:	Csaba Noszaly
Date:	2005-05-05
Time limit:	8s
Source limit:	12345B
Languages:	All
Resource:	Folklore

SPOJ Problem Set (classical)

526. Divisors

Problem code: DIV

Let N be a positive integer. In theory it is easy to decide if d(N) (the number of positive divisors of N including 1 and N) is prime or not. Your task is just a little bit harder: compute all N in [1,10^6] for which d(N)=p*q where p and q distinct primes.

Input

There is no input for this problem.

Output

To make the problem less io related write out only every 9-th of them, one per line.

Output:

```
50
99
162
...
999524
999728
999927
```

Added by: Csaba Noszaly
Date: 2005-05-16
Time limit: 4s
Source limit:3333B
Languages: All
Resource: Folklore.

SPOJ Problem Set (classical)

530. Divisors 2

Problem code: DIV2

Let N be a positive integer and d(N) be the number of positive divisors of N including 1 and N. Your task is to compute all N in [1,10^6] for which d(N)>3 and if M divides N then d(M) divides d(N) too.

Input

None.

Output

To make the problem less output related write out only every 108-th of them, one per line.

Example

```
Output:
267
511
753
...
999579
999781
999977
```

Added by: Csaba Noszaly
Date: 2005-05-24
Time limit: 9s
Source limit:3333B
Languages: All
Resource: Folklore

SPOJ Problem Set (classical)

598. Increasing Subsequences

Problem code: INCR

A sequence **p(1)**, **p(2)**, ..., **p(N)** consisting of numbers 1, 2, ..., **N** is called a permutation if all elements in the sequence are different.

It is said that a permutation **p** contains increasing subsequence of **k** elements when there are numbers 1 <= $\mathbf{i_1} < \mathbf{i_2} < ... < \mathbf{i_k}$ <= N such that $\mathbf{p(i_1)} < \mathbf{p(i_2)} < ... < \mathbf{p(i_k)}$.

When a permutation **p** contains an increasing subsequence consisting of **B** elements and does not contain an increasing subsequence consisting of **B+1** elements then the number **B** is called the degree of increase of this permutation.

You need to write a program which being given a number **N** calculates the number of permutations whose degree of increase is **B**. Since the number of such permutations might be quite big, it is necessary to calculate its remainder of integer division by 1 000 000 000.

Input

First line of input contains integer **T** (1 <= **T** <= 60) - the number of testcases. Then descriptions of **T** testcases follow.

The description of the testcase consists of one line. The line contains two integer numbers **N** and **B** (1 <= **N** <= 40, 1 <= **B** <= 5) separated by one or more spaces.

Output

For each testcase in the input your program should output one line. This line should contain one integer number which is the remainder of integer division by 1 000 000 000 of the number of permutations whose degree of increase is **B**.

Example

Input:
```
1
3 2
```

Output:
```
4
```

Added by: Ivan Metelsky
Date: 2005-11-07
Time limit: 15s
Source limit: 50000B
Languages: All
Resource: NEERC Western Subregion QF 2005

SPOJ Problem Set (classical)

660. Dungeon of Death

Problem code: QUEST4

To reach the treasure, **Jones** has to pass through the **"Room of Death"**. The floor of this room is a square with side **120** units. It is laid with square tiles of dimensions **{1 X 1}** arranged into a grid. But, at some places in the grid tiles are missing. As soon as the door to this room is opened poisonous gas starts coming out of these missing grid locations. The only escape from this gas is to completely cover these locations with planks lying outside the room. Each plank has dimensions **{120 X 1}** and can only be placed parallel to either sides of the floor. Now **Jones** wants to minimize the damage to his health so that he has enough of it left for the treasure. He figures out that in order to achieve this he has to use the minimum number of planks possible. He also realises that even if the planks overlap, poisonous gas from the missing tiles can still be successfully blocked. Please help **Jones** in this task.

Dungeon of Death: Tiles Uncovered

Dungeon of Death: Tiles Covered

Input

- The first line of the input is a positive integer **t <= 20**, denoting the number of rooms.
- The descriptions for the t rooms follow one after the other.
- **Room Description:**
  - The first line of the room description is a positive integer **n** (**n <= 10010**), denoting the number of missing tile locations.
  - This is followed by the **n** lines, one for each missing tile location.
  - Each line contains two integers **x y** (**0 <= x, y < 120**), separated by a single space, representing the co-ordinates of the missing tile location.

Output

The output should consist of **t** lines, one for each room. The $\mathbf{k}^{th}$ line in the output should be an integer $\mathbf{m_k}$, the minimum number of planks needed for the $\mathbf{k}^{th}$ room.

Example

Input:

```
2
3
1 0
2 0
3 0
4
1 1
2 2
```

```
3 3
4 4
```

Output:

```
1
4
```

Added by: Kashyap KBR
Date: 2005-12-08
Time limit: 8s
Source limit:50000B
Languages: All

SPOJ Problem Set (classical)

661. Nail Them

Problem code: QUEST5

To get to the treasure, **Jones** must complete one more task. He comes across a table, where there are a number of wooden planks lying along the length of the table. He notices that the width of the table is exactly equal to the width of every plank on it. The planks are so heavy that they cannot be manually moved in any way. Some of these wooden planks are overlapping. **Jones** has a hammer and the Gods grant him infinite nails. The planks have to be joined to the table with nails such that every plank is connected to the table through at least one nail. The nails are of sufficient length, and have to be hammered vertically into the table. One or more planks can be joined to the table through a single nail provided they have a common overlap. Find out the minimum number of nails he needs to nail all planks to the table.

Planks

Input

- The first line of the input is a positive integer **t <= 20**, denoting the number of tables.
- The descriptions of the table follow one after the other.
- **Table description:**
  - The first line of the description of the $\mathbf{k^{th}}$ table contains a positive integer **n (n <= 10010)**, the number of planks on it.
  - This is followed by **n** lines containing the description of the planks.
  - The description of each plank is a pair of integers **a** and **b (0 <= a <= b <= 10000010)**, denoting the distance of the left end and right end of the plank from the left end of the table.

Output

The output must contain **t** lines , the $\mathbf{k^{th}}$ line corresponding to the $\mathbf{k^{th}}$ table. The output on the $\mathbf{k^{th}}$ line must be an integer $\mathbf{i_k}$, the minimum number of nails required.

Example

Input:

```
2
3
1 5
3 5
2 4
2
1 4
4 5
```

Output:

```
1
1
```

Added by: Kashyap KBR
Date: 2005-12-08
Time limit: 2s
Source limit: 50000B
Languages: All

SPOJ Problem Set (classical)

665. String it out

Problem code: SUBS

Let **A** and **B** be two strings made up of alphabets such that $\mathbf{A = A_{[1\text{-}n]}}$, $\mathbf{B = B_{[1\text{-}m]}}$. We say **B** is a subsequence of **A** if there exists a sequence of indices $\mathbf{i_1 < i_2 <..}m$ of **A** such that $\mathbf{A[i_k] = B[k]}$.

Given **B[1-m]**, a string of characters from some alphabets, **B^i** is defined as string with the characters of **B** each repeating **i** times. For example, **(abbacc)^3 = aaabbbbbbaaaccccccc**. Also, **B^0** is the empty string.

Given strings **X**, **Y** made up of characters from **'a' - 'z'** find the maximum value of **M** such that **X^M** is a subsequence of **Y**.

Input

- The first line of the input contains a positive integer **t <= 20**, denoting the no. of test cases.
- The following **2t** lines contain the value of **X** and **Y** for the cases.
- The description of the test cases follow one after the other.
  - Line **2k** contains the value of **X** for case **k**; **(1 <= k <= t)**
  - Line **2k+1** contains the value of **Y** for case **k**; **(1 <= k <= t)**.
  - The no. of characters in **X** , **Y** will be **<= 500010**.

Output

The output must contain **t** lines, each line corresponding to a test case. The value on the $\mathbf{k^{th}}$ line should be the value of **M** for the $\mathbf{k^{th}}$ pair of **X** and **Y**.

Example

Input:

```
3
abc
aabbcc
abc
bbccc
abcdef
abc
```

Output:

```
2
0
0
```

Added by: Kashyap KBR
Date: 2005-12-12
Time limit: 8s
Source limit:50000B
Languages: All

SPOJ Problem Set (classical)

666. Con-Junctions

Problem code: VOCV

The city of **Y-O** is a network of two-way streets and junctions with the following properties:

1. There is no more than one street between each pair of junctions.
2. Every junction is connected to every other junction either directly via a street or through other junctions by a unique path.
3. When a light is placed at a junction, all the streets meeting at this junction are also lit.

A valid lighting is a set of junctions such that if lights were placed at these, all the streets would be lit. An optimal lighting is a valid lighting such that it contains the least number of junctions.

The task is divided into two subtasks:

1. Find the number of lights in an optimal lighting.
2. Find the total number of such optimal lightings in the city.

Input

- The first line of the input contains a positive integer **t <= 20**, denoting the number of test cases.
- The description of the test cases follows one after the other.
- **Network Description:**
  - The first line of description of a network consists of a positive integer **n <= 100010** denoting the number of junctions in the network.
  - Each junction is numbered with a unique integer between **1** and **n**.
  - The following **n-1** lines contain a pair of integers **u v (1 <= u,v <= n)** separated by a single space denoting that there is a street between junction **u** and junction **v**.

Output

The output must consist of **t** lines, the **k th** line corresponding to the **k th** network; **(1 <= k <= t)**. The **k th** line must contain two integers separated by a single space. The first integer on the **k th** line must be the number of junctions in an optimal lighting of network **k**. The second integer must be **N%10007**, which is the remainder left by the number of optimal lightings when divided by **10007**.

Example

Input:

```
2
4
1 2
2 3
3 4
3
```

```
1 2
1 3
```

Output:

2 3
1 1

Added by: Kashyap KBR
Date: 2005-12-12
Time limit: 10s
Source limit: 50000B
Languages: All except: NICE JAR NEM ST SCM qobi

SPOJ Problem Set (classical)

676. Sorting is not easy

Problem code: LSORT

An N-element permutation is an N-element sequence of distinct numbers from the set {1, 2, ...,n}. For example the sequence 2,1,4,5,3 is a 5-element permutation. P is an N-element permutation. Your task is to sort P in ascending order. But because it is very simple, I have a new rule for you. You have two sequences P and Q. P is an N-element permutation and Q is initially empty and formed by sorting P (i.e. finally Q = 1, 2, 3,... , N). You have to implement N steps to sort P. In the i-th step, P has N-i+1 remaining elements, Q has i-1 elements and you have to choose some x-th element (from the N-i+1 available elements) of P and put it to the left or to the right of Q. The cost of this step is equal to $x * i$. The total cost is the sum of costs of individual steps. After N steps, Q must be an ascending sequence. Your task is to minimize the total cost.

Input

The first line of the input file is T (T <= 10), the number of test cases. Then descriptions of T test cases follow. The description of each test case consists of two lines. The first line contains a single integer N (1 <= N <= 1000). The second line contains N distinct integers from the set {1, 2, .., N}, the N-element permutation P.

Output

For each test case your program should write one line, containing a single integer - the minimum total cost of sorting.

Example

N = 4
P = {4,1,3,2}
Step 1, Choose 3-rd, P={4,1,2}, Q={3} , Cost=3
Step 2, Choose 1-st, P={1,2}, Q={3,4} , Cost=2
Step 3, Choose 2-nd, P={1}, Q={2,3,4} , Cost=6
Step 4, Choose 1-st, P={}, Q={1,2,3,4}, Cost=4
The total cost is 15.
Another way to sort:
Step 1, Choose 4-th, P={4,1,3}, Q={2} , Cost=4
Step 2, Choose 2-nd, P={4,3}, Q={1,2} , Cost=4
Step 3, Choose 2-nd, P={4}, Q={1,2,3} , Cost=6
Step 4, Choose 1-st, P={}, Q={1,2,3,4}, Cost=4
The total cost is 18.

Input:
```
1
4
4 1 3 2
```
Output:
```
15
```

Added by: Nguyen Minh Hieu
Date: 2005-12-20
Time limit: 2s
Source limit: 10000B
Languages: All except: C99 strict
Resource: Romanian National Contest

SPOJ Problem Set (classical)

677. A place for the brewery

Problem code: BROW

The dwellers of the island Abstinence are very fond of alkoholfree beer. Hitherto alkohol-free beer was imported from Poland, but this year one of the cities on Abstinence is going to build a brewery. All the cities of this island lie on the coast and are connected by a highway running around the island along its shore. The investor building the brewery collected information about the demand for beer, i.e. how many tanks of beer are needed daily in each city. He has also a table of distances between cities. The cost of transporting one tank is 1 thaler per mile. A daily cost of transport is the amount of money, which has to be spent on transporting a necessary number of tanks of beer from the brewery to each city. The daily cost depends on the location of the brewery. The investor wants to find a location that minimizes the daily cost.

Task

Write a program which

- reads the number of cities, distances between them and daily requests for beer,
- computes the minimal daily cost of transport,
- writes the result.

Input

There are multiple test cases. Their number is given in the first line of input. In the first line of each test case there is one integer n - the number of cities, 5 <= n <= 10 000. (We assume that cities are numbered along the highway, so that the neighbouring cities have subsequent numbers. Cities 1 and n are neighbours too.) In each of the following n lines there are two non-negative numbers separated by a single space. Numbers zi di written in the line (i+1) are respectively the demand for beer in the city i and the distance (in miles) from city i to the next city on the highway. The entire length of the highway is not greater than 1 000 000 miles. The demand for beer in each city is not greater than 1 000 tanks.

Output

For each test case your program should write only one line - exactly one integer equal to the minimal daily cost of transport.

Example

Input:

```
1
6
1 2
2 3
1 2
5 2
1 10
```

```
2 3
```

Output:

```
41
```

Added by:	Paweł Dobrzycki
Date:	2005-12-21
Time limit:	1s
Source limit:	50000B
Languages:	All
Resource:	VII Polish Olympiad in Informatics, Ist Stage

SPOJ Problem Set (classical)

681. Building the Tower

Problem code: HANOI07

There are N cubes in a toy box which has 1-unit height, the width is double the height. The teacher organizes a tower-building game. The tower is built by the cubes. The height of the tower is H (h levels). The bottom of the tower contains M cubes; and for all above level, each must contains a number of cubes which is exactly 1 less than or greater than the number of cubes of the level right below it. Your task is to determine how many different towers can be there. Two towers are considered different if there is at least one number i (1< i <=H) so that the i'th level of one tower contains a different number of cubes to the i'th level of the other tower.

Input

The first line of input file is the integer number t (0 < t < 1002) , the number of test cases . Each test case in one line , the line contains three positive number N, H and M (N <= 32767, H<=60, M<=10).

Output

With each test case , write in one line , the total of different towers that can be founded.

Example

Input:

```
2
7 3 2
22 5 4
```

Output:

```
2
10
(* In the first test case , all the towers are : 2-1-2 , 2-3-2 . *)
```

Added by: Nguyen Minh Hieu
Date: 2005-12-30
Time limit: 7s
Source limit:7777B
Languages: All
Resource: ACM

SPOJ Problem Set (classical)

682. Pairs of Integers

Problem code: PAIRINT

You are to find all pairs of integers such that their sum is equal to the given integer number N and the second number results from the first one by striking out one of its digits. The first integer always has at least two digits and starts with a non-zero digit. The second integer always has one digit less than the first integer and may start with a zero digit.

Input

The first line of the input file is the integer number t (1 <= t <= 20), the number of test cases. Then t lines follow, each test case in one line; the line consists of a single integer N (10 <= N <= 10^9).

Output

For each test case:

On the first line write the total number of different pairs of integers that satisfy the problem statement. On the following lines write all those pairs. Write one pair on a line in ascending order of the first integer in the pair. Each pair must be written in the following format

X + Y = N

Here X, Y, and N, must be replaced with the corresponding integer numbers. There should be exactly one space on both sides of '+' and '=' characters.

Example

Input:

```
2
302
11
```

Output:

```
5
251 + 51 = 302
275 + 27 = 302
276 + 26 = 302
281 + 21 = 302
301 + 01 = 302
1
10 + 1 = 11
```

Added by: Nguyen Minh Hieu
Date: 2006-01-01
Time limit: 1s
Source limit: 10000B
Languages: All except: C99 strict
Resource: 2001-2002 ACM Northeastern European Regional Programming Contest

SPOJ Problem Set (classical)

684. Another Assignment Problem

Problem code: ASSIGN4

Assume that you are a manager and there are m types of worker (numbered from 1 to m) and n types of task (numbered from 1 to n). There are a(i) workers of type #i and b(j) postitions for task #j. C(i, j) is the cost of hiring a worker of type #i to do the task of type #j. Your job is to minimize the cost of hiring workers to fill all the positions given that the total number of workers is equal to the total number of positions.

Input

The first line of input contains the number of test cases nTest (1<= nTest <= 10). Each test case contains:

- The first line contains the number of worker types - m and number of task types - n.
- The second line contains m positive integers: a(1), a(2), ..., a(m).
- The third line contains n positive integers: b(1), b(2), ..., b(n).
- Each of the next m lines contains n integers describing matrix C(i, j).

Notes:

1 <= m, n <= 200;

1 <= a(i), b(i) <= 30000;

1 <= C(i, j) <= 10000.

Sum of a(i) equals to sum of b(j).

Output

For each test case write the minimum cost in a separate line (it will fit in a signed 32-bit integer).

Example

Input:

```
2
3 4
3 6 7
2 5 1 8
1 2 3 4
8 7 6 5
9 12 10 11
4 4
1 3 5 7
2 4 2 8
1 4 7 3
4 7 5 3
```

```
5 7 8 3
5 3 6 8
```

Output:

```
110
54
```

Added by: Nguyen Dinh Tu
Date: 2006-01-02
Time limit: 5s
Source limit:50000B
Languages: All
Resource: Tran Quang Khai

SPOJ Problem Set (classical)

685. Partition the sequence

Problem code: SEQPAR

Given an integer sequence containing n elements (numbered from 1 to n), your task is to find the minimum value M so that we can find k + 1 integers 0 = p(0) < p(1) < p(2) < ... < p(k-1) < p(k) = n, such that for any i from 0 to k - 1, the sum of elements from postition p(i)+1 to postition p(i+1) is not greater than M.

Input

The first line of input contains the number of test cases nTest (1 <= nTest <= 10).

Each test case contains:

The first line contains n, k. (1 <= k <= n <= 15000)

Each of the next n lines contains an integer of the sequence with value range from -30000 to 30000.

Output

For each test case write the minimum number M in a separate line.

Example

Input:

```
1
9 4
1
1
1
3
2
2
1
3
1
```

Output:

```
5
```

Added by: Nguyen Dinh Tu
Date: 2006-01-02
Time limit: 44s
Source limit:50000B
Languages: All
Resource: Viet Nam Olympiad in Informatic 2005, Day I

SPOJ Problem Set (classical)

687. Repeats

Problem code: REPEATS

A string s is called an (k,l)-repeat if s is obtained by concatenating k>=1 times some seed string t with length l>=1. For example, the string

s = abaabaabaaba

is a (4,3)-repeat with t = aba as its seed string. That is, the seed string t is 3 characters long, and the whole string s is obtained by repeating t 4 times.

Write a program for the following task: Your program is given a long string u consisting of characters ‘a’ and/or ‘b’ as input. Your program must find some (k,l)-repeat that occurs as substring within u with k as large as possible. For example, the input string

u = babbabaabaabaabab

contains the underlined (4,3)-repeat s starting at position 5. Since u contains no other contiguous substring with more than 4 repeats, your program must output the maximum k.

Input

In the first line of the input contains H- the number of test cases (H <= 20). H test cases follow. First line of each test cases is n - length of the input string (n <= 50000), The next n lines contain the input string, one character (either ‘a’ or ‘b’) per line, in order.

Output

For each test cases, you should write exactly one interger k in a line - the repeat count that is maximized.

Example

Input:

```
1
17
b
a
b
b
a
b
a
a
b
a
a
b
a
```

```
a
b
a
b
```

Output:

```
4
```

since a (4, 3)-repeat is found starting at the 5th character of the input string.

Added by:	Hoang Hong Quan
Date:	2006-01-05
Time limit:	18s
Source limit:	50000B
Languages:	All
Resource:	BOI 2004

SPOJ Problem Set (classical)

688. Toy Cars

Problem code: SAM

Jasio is a little boy - he is only three years old and enjoys playing with toy cars very much. Jasio has n different cars. They are kept on a shelf so high, that Jasio cannot reach it by himself. As there is little space in his room, at no moment may there be more than k toy cars on the floor. Jasio plays with one of the cars on the floor. Jasio's mother remains in the room with her son all the time. When Jasio wants to play with another car that is on the floor, he reaches it by himself. But when the toy is on the shelf, his mummy has to hand it to him. When she gives Jasio one car, she can at the same time pick any car from the floor and put it back on the shelf (so that there remains sufficient space on the floor). The mother knows her child very well and therefore can predict perfectly which cars Jasio will want to play with. Having this knowledge, she wants to minimize the number of times she has to hand Jasio a toy from the shelf. Keeping that in mind, she has to put the toys off on the shelf extremely thoughtfully.

Task

Write a programme that:

1.reads from the standard input the sequence of toy cars in order in which Jasio will want to play with them,

2.calculates the minimal number of times the mother has to pick cars from the shelf,

3.writes the result to the standard output.

Input

In the first line of the standard input is H- the number of test case (H <= 16). For each test case follow contains some lines, start with three integers: n, k, p (1 <= k <= n <= 100000, 1 <= p <= 500000), separated by single spaces. These denote respectively: the total number of cars, the number of cars that can remain on the floor at once and the length of the sequence of cars which Jasio will want to play with. Each of the following p lines contains one integer. These integers are the numbers of cars Jasio will want to play with (the cars are numbered from 1 to n).

Output

For each test case, you should write only one integer - the minimal number of times his mother has to pick a car from the shelf.

Example

Input:

```
For the following input data:
3 2 7
1
2
```

```
3
1
3
1
2
```

Output:

```
the correct answer is:
4
```

Added by: Hoang Hong Quan
Date: 2006-01-08
Time limit: 8s
Source limit:50000B
Languages: All
Resource: 12th Polish Olympiad in Informatics, stage 1

SPOJ Problem Set (classical)

693. Lethal Warfare

Problem code: LWAR

A major cosmic battle was getting over. The InterGalactic SuperPower had been under attack, but it had defended itself quite well. It was about to launch its final retaliatory assault. But the number of enemy ships was quite large and they could scatter very easily. Their only hope, or so their Space Warfare expert said, was to bomb the enemies (who happened to be lined up in a long line!) using the strategy described below.

Because the number of ships will be a power of 2, to bomb all the ships (numbered 0 to 2^N -1), the strategy to be used, which we will call **BombStrat**, goes like this:
1. Bomb it's first half, [0 to 2^{N-1} -1], in the left to right direction.
2. Of the remaining half, bomb its latter half part in reverse direction, i.e., bomb ships 2^N-1, 2^N-2,...., $2^{N-1}+2^{N-2}$ in that order.
3. Then use **BombStrat** on the remaining ships: [2^{N-1} to $2^{N-1} + 2^{N-2}$ -1]

For example, when N=3, i.e., with ships numbered from 0 to 2^3 -1, this is what happens:
Step 1: Ships 0,1,2,3 get bombed in that order.
Step 2: Ships 7, 6 go down next.
Step 3: Now, the remaining ships [4, 5] are destroyed using the same strategy.

So the bombing is done in the order 0 -> 1 -> 2 -> 3 -> 7 -> 6 -> 4 -> 5. To make the job easier for the InterGalactic SuperPower's ships' pilots, they want to find which ship should be bombed when. This is your task. Given N, and the description of a ship, return the 0-based serial number of the bomb will blast it.

Input

T - the number of test cases, T<=50.
For each test case:

One line containing a binary number, describing the number of the place. The length of this string will equal N (it will be padded with leading zeroes if necessary). N<=30000.

Output

For each test case, output the index of a bomb, represented in the same format, as binary digits, whose length is exactly N.

Example

```
Sample Input:
3
111
100
1100
```

```
Sample Output:
100
110
1011
```

Added by:	Prasanna
Date:	2006-01-13
Time limit:	1s
Source limit:	50000B
Languages:	All
Resource:	ByteCode '06

SPOJ Problem Set (classical)

694. Distinct Substrings

Problem code: DISUBSTR

Given a string, we need to find the total number of its distinct substrings.

Input

T- number of test cases. T<=20;
Each test case consists of one string, whose length is <= 1000

Output

For each test case output one number saying the number of distinct substrings.

Example

Sample Input:

```
2
CCCCC
ABABA
```

Sample Output:

```
5
9
```

Explanation for the testcase with string ABABA:
len=1 : A,B
len=2 : AB,BA
len=3 : ABA,BAB
len=4 : ABAB,BABA
len=5 : ABABA
Thus, total number of distinct substrings is 9.

Added by: Prasanna
Date: 2006-01-13
Time limit: 1s
Source limit:50000B
Languages: All
Resource: ByteCode '06

SPOJ Problem Set (classical)

695. Unite Fast

Problem code: UFAST

The Agents need to unite. They are on a road and each of them possess a special device which can both send and receive signals, in both directions upto a maximal distance of D units. Apart from this small limitation, the devices work very efficiently so that the time taken for interdevice communication is practically zero. Now that the agents are at different points on the road, for them to be able to communicate with each other at will, every agent should be connected to every other agent through one or more intermediate devices. For example: agent A may communicate to agent C via agent B's device, when A and C are not close enough. This happens when **dist**(A,C)>D, but **dist**(A,B)<=D and **dist**(B,C)<=D.

Getting the line ready, is the process of agents moving from their current positions in order to get the network fully connected. That is, from every agent to every other agent, there is a communication path.

The agent's final positions (in two cases that are going to follow) are decided by a programmer, who watches the scene from above and instructs each agent of the time to move and the final position to move to. Each agent moves a unit distance in unit time.

We need to find the minimal time taken for the programmer to "get the line ready" if he moves the agents:
1. Independently: In other words, every agent moves to their final position without waiting for any other agent; all agents are told of their final positions at time zero.
2. Sequentially: In this the agents form a definite sequence of movement. No two agents are moving at the same time.

Input

T - number of test cases. For each test case :
N D - where N is the number of agents, D is the maximal communication distance
The i-th line, of the N-lines that follow gives the position of the i-th agent on the road currently.

Output

For each test case, output two integers;
1st - minimal time taken to unite if they move independently
2nd - minimal time taken to unite if they move sequentially

Constraints:
T<=20
1<=N,D<=100 ;
Each agent's initial position is between 0 and 1000.

Example

Sample Input:

```
2
4 3
10 20 30 35
5 3
1 2 3 4 30
```

Sample Output:

```
8 23
12 23
```

Added by:	Prasanna
Date:	2006-01-13
Time limit:	3s
Source limit:	50000B
Languages:	All
Resource:	ByteCode '06

SPOJ Problem Set (classical)

696. Liar Liar

Problem code: LIAR

Professor Millman hates us, and worse, characterizes us as liars. We don't care if he means it or not, but we (more professional that him!) planned to give the lower and upper bound on the number of liars in the class (so that you know what happens the next time he scolds us!).

To start with we took a survey of all students in the class. Each student gave a reply about every student saying whether that student is a liar or not. These answers are in the form of a Matrix A, where A[i][j] represents the reply given by the i-th student about the j-th student. If that character is 'L' - it means he/she is a liar; if it's 'T' - then it means that, that student is a truth speaker.

We take the following as our definition of the terms Truth-Speaker, and Liar:
Truth-Speaker ('T'): All his/her replies are true.
Liar ('L') : (S)he has made at least one false reply.

Input

T - the number of test cases; For each test case :
N - total number of students in the class
Matrix A of NxN characters, without space separation;

Output

For i-th test case output one line of the form "Class Room#i contains atleast A and atmost B liars", where A and B are the lower and the upper bounds on the number of liars respectively. If there is a paradoxical class room, instead of the above line, print "Class Room#i is paradoxical".

Constraints:
T<=50; Our class rooms contain atmost 70 students.

Example

Sample Input:

```
4
2
LL
TT
3
TTT
TTT
TTT
4
TLLL
LTLL
```

```
LLTL
LLLT
5
TLTLT
TTTTT
LLTLL
LLLLL
TLTLT
```

Sample Output:

```
Class Room#1 is paradoxical
Class Room#2 contains atleast 0 and atmost 3 liars
Class Room#3 contains atleast 3 and atmost 4 liars
Class Room#4 contains atleast 4 and atmost 4 liars
```

Here a paradox occurs if a person can't be classified as a liar or a truth-speaker.

Added by: Prasanna
Date: 2006-01-13
Time limit: 1s
Source limit: 50000B
Languages: All
Resource: ByteCode '06

SPOJ Problem Set (classical)

697. Matrix Words

Problem code: MWORDS

Given an NxN matrix of characters. We start at position (1,1) and want to reach (N,N) in exactly 2N-1 moves. Each move consists of movement in one of the four standard directions. As we move, we collect the characters found in our positions forming a string. We now constrain our attention to all paths that do not cross the diagonal of the matrix. However the parts of the path can be on the diagonal line. These paths can be classified into two partitions; the paths that lie above and paths that lie below the diagonal. Each path is represented by a string of characters formed by the ordered concatenation of characters found on the way. If we consider the set of all valid paths, (both upper and lower) get their corresponding strings, sort them all in alphabetical order, we obtain the (ordered) master set. Note that the master set might contain duplicates, and all strings in the master set consist of exactly 2N-1 characters. Let M be the total number of strings in the master set, given an integer I, we need to find the string with index = I (modulo M) within the master set.
If Master Set = { "A","B","B","C" } (although this set can never be a master set!)
I=0 produces "A", while I=2 and I=5, produces "B".

Constraints:
N<=30.
I<=10^{18}. 'I' will fit into a 64-bit integer.

Input

T-number of test cases
N I
Next is the NxN matrix of characters, N characters per line.
All characters are between 'A'-'Z' (only uppercase).

Output

For each test case output the corresponding string sought for in the master set.

Example

Sample Input:

```
2
3 18
DAA
BDA
BBD
3 18
DAA
ADA
AAD
```

Sample Output:

```
DBBBD
DADAD
```

Explanation:

Test case I: Master Set = { "DAAAD", "DADAD","DBBBD","DBDBD"}
Test case II: Master Set = { "DAAAD","DAAAD","DADAD","DADAD"}

Added by: Prasanna
Date: 2006-01-13
Time limit: 1s
Source limit:50000B
Languages: All
Resource: ByteCode '06

SPOJ Problem Set (classical)

698. Plane Hopping

Problem code: PLHOP

This man has grown so rich that, when he travels between any two locations he always takes atleast K flights. In a region of N cities, we need to find the minimal cost required for the man to travel between every pair of cities. There are provisions (especially for this type of rich men,) to fly from i-th city to the i-th city itself!

Input

T - The number of test cases.
In each test case :
K N
NxN matrix representing the costs of the tickets. The i-th line, j-th column's entry represents the cost of a ticket from city i to city j. The numbers are of course space separated.

Constraints :
T<=20
N<=50
K<=10^9
The cost of each ticket <= 100
Each element of the output matrix will fit into a 64-bit integer.

Output

For the i-th test case , 1st line is of the form "Region #i:".
In the following N lines, output an NxN matrix where the j-th element of the i-th line represents the minimal cost to travel from city i to city j with taking atleast K flights. The numbers on a line must be separated by atleast one space. Output a blank line after each testcase (including the last one).

Example

Sample Input:

```
2
3 4
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16
10999 4
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16
```

Sample Output:

```
Region #1:
3 4 5 6
7 8 9 10
11 12 13 14
15 16 17 18

Region #2:
10999 11000 11001 11002
11003 11004 11005 11006
11007 11008 11009 11010
11011 11012 11013 11014
```

Added by: Prasanna
Date: 2006-01-13
Time limit: 2.5s
Source limit: 50000B
Languages: All
Resource: ByteCode '06

SPOJ Problem Set (classical)

699. Huge Knap Sack

Problem code: HKNAP

Our King has won the brutal battle and this whole land is now ours. The special thing about this land is that it has many beautiful golden statues. Our King wants to take back as much gold as possible to his palace. We have found that there are **N** types of statues and -- almost unbelievably -- that there is an unlimited number of each type of statue. Each statue of type **i** has a weight of **W[i]** units and occupies **V[i]** units of volume. Our King wants to maximize the amount of gold he carries back to his palace. We may use **S** sacks for this purpose, each of volume **Y**. All sacks are filled up independently by golden statues. However, there is a provision to stitch two sacks together, at the cost of **C** units of gold. Stitching three sacks costs 2*C because it requires two stitchings, and so on. Your task is to find how much gold our King can possibly gain, i.e. the total weight of the statues brought back, minus the stitching charges.

Input

T - The number of test cases.
For each test case :
N S Y C // 1st line
Next N lines two numbers W[i] and V[i] each.

Output

One integer, the maximum gain in gold for our King.
This gain is the total amount of gold transported minus stitching charges.

Constraints :
1<= S <= 1000
1<= Y <= 1000 000 000
1<= N <= 1000
1<= W[i] <= 100; (for all i)
1<= V[i] <= 18;
The Output will fit into a 64-Bit integer.
1<=T<=20
All W[i] & V[i] are guaranteed to be either prime or equal to 1.

Example

Sample Input:

```
2
2 5 3 1
1 2
5 7
2 5 3 1
1 2
```

```
7 5
```

Sample Output:

```
6
17
```

Added by:	Prasanna
Date:	2006-01-13
Time limit:	2s
Source limit:	50000B
Languages:	All
Resource:	ByteCode '06

SPOJ Problem Set (classical)

700. Branch Prediction

Problem code: BPRED

As most of you might already know, the Intel-class hi-tech processors of today do a series of parallel tasks to help speedup instruction execution. The most complicated of those tasks is branch prediction. Since the instruction chunks on a modern processor are broken down into independent chunks and executed for a speed up, there is always a requirement to predict what branch an execution path will take (before the actual operands required for the condition to be evaluated to select the branch, are available). This complex task, is not addressed to fullest level today, but heuristics (as always) have helped.

The task we are going to consider now is much more simple compared to the actual branch prediction task. For our modelling, let us suppose that every instruction has the following syntax:
All labels are strings of alphabets only. Labels are case-sensitive.

Moreover the probability that a certain branch will be taken is **P** (it is equal for all branches). If a branch is taken, the point of execution (control) goes to the branched-label. Otherwise the next statement in that order is executed. Control starts at the "**start**" (lowercase) label and control ends at the "**end**" (lowercase) label. The branch-label of start and end are themselves, and when start is executed, the control goes to the next instruction, and when end is executed, processing ends, with 100% probability. The last statement in the program is always an "**end**".

It is required to find the expected number of times a statement executes.

Input

T - the number of test cases;
For each test case:
1st line contains one integer N (the number of lines to follow), one real P and one label L.
Each of the N lines that follow consist of instructions (two labels).

Output

For each test case, output one line containing:
"Expected number of times label L is executed is R",
where L - is the label given in the input
R - is the number of times the label is expected to be executed. It must be printed with exactly five decimal places.

Constraints:
T<=20
3<=N<=120.
P lies between 0.01 and 0.99, i.e. no jump is 100% sure.
Also you can assume no label occurs on the jump side, without being defined throughout the program.
Each label is less than 10 characters in length.
Also each line has a distinct label associated with it.

Example

Sample Input:

```
3
5 .5 B
C start
start start
B C
D C
end end
5 .99 C
start start
A B
B A
C end
end end
3 .5 label
start start
label label
end end
```

Sample Output:

```
Expected number of times label B is executed is 4.00000
Expected number of times label C is executed is 1.00000
Expected number of times label label is executed is 2.00000
```

Added by: Prasanna
Date: 2006-01-13
Time limit: 1s
Source limit: 50000B
Languages: All
Resource: ByteCode '06

SPOJ Problem Set (classical)

702. Barn Expansion

Problem code: EXPAND

Farmer John has N (1 <= N <= 25,000) rectangular barns on his farm, all with sides parallel to the X and Y axes and integer corner coordinates in the range 0..1,000,000. These barns do not overlap although they may share corners and/or sides with other barns. Since he has extra cows to milk this year, FJ would like to expand some of his barns. A barn has room to expand if it does not share a corner or a wall with any other barn. That is, FJ can expand a barn if all four of its walls can be pushed outward by at least some amount without bumping into another barn. If two barns meet at a corner, neither barn can expand. Please determine how many barns have room to expand.

Input

t - the number of test cases, then t test cases follow.

Each test case takes the following form:

The first line contains the number of rectangular barns - n.

Each of the next n lines contains:

Four space-separated integers A, B, C, and D, describing one barn. The lower-left corner of the barn is at (A,B) and the upper right corner is at (C,D).

Output

For each test case write a single integer that is the number of barns that can be expanded in a separate line

Example

Input:

```
1
5
0 2 2 7
3 5 5 8
4 2 6 4
6 1 8 6
0 0 8 1
```

Output:

```
2
```

Input/Output details:

There are 5 barns. The first barn has its lower-left corner at (0,2) and its upper-right corner at (2,7), and so on.

Only two barns can be expanded --- the first two listed in the input. All other barns are each in contact with at least one other barn.

Added by: Nguyen Dinh Tu
Date: 2006-01-17
Time limit: 4s
Source limit:50000B
Languages: All
Resource: USACO December 2005 Gold Division

SPOJ Problem Set (classical)

703. Mobile Service

Problem code: SERVICE

A company provides service for its partners that are located in different towns. The company has three mobile service staff employees. If a request occurs at some location, an employee of the service staff must move from his current location to the location of the request (if no employee is there) in order to satisfy the request. Only one employee can move at any moment. They can move only on request and are not allowed to be at the same location. Moving an employee from location p to location q incurs a given cost C(p,q). The cost function is not necessarily symmetric, but the cost of not moving is 0, i.e. C(p,p)=0. The company must satisfy the received requests in a strict first-come, first-serve basis. The goal is to minimize the total cost of serving a given sequence of requests.

Task

You are to write a program that decides which employee of the service staff is to move for each request such that the total cost of serving the given sequence of requests is as small as possible.

Input

The first line of input contains the number of test cases - nTest. Each test case contains:

The first line of each test cases contains two integers, L and N. L (3 <= L <= 200) is the number of locations and N (1 <= N <= 1000) is the number of requests. Locations are identified by the integers from 1 to L. Each of the next L lines contains L non-negative integers. The jth number in the line i+1 is the cost C(i,j), and it is less than 2000.

The last of each test cases contains N integers, the list of the requests. A request is identified by the identifier of the location where the request occurs. Initially, the three service staff employees are located at location 1, 2 and 3, respectively.

Output

For each test case write the minimal total cost in a separate line.

Example

Input:

```
1
5 9
0 1 1 1 1
1 0 2 3 2
1 1 0 4 1
2 1 5 0 1
4 2 3 4 0
4 2 4 1 5 4 3 2 1
```

Output:

```
5
```

Added by:	Nguyen Dinh Tu
Date:	2006-01-17
Time limit:	10s
Source limit:	50000B
Languages:	All
Resource:	CEOI 2005, Day 1

SPOJ Problem Set (classical)

704. Remove The String

Problem code: PSTRING

Given two strings X and Y, your task is find the minimum number of characters to be removed from X in order to obtain a string X' that does not contain Y as a substring.

Input

Input contains some test cases. Each test cases contains two lines, First is X and second is Y. Length of X <= 10000, Length of Y <= 1000.

Output

For each test cases, You should output exactly one integer is the minimum number of characters to be remove

Example

Input:
```
ababaa
aba
```

Output:
```
1
```

Added by: Hoang Hong Quan
Date: 2006-01-17
Time limit: 3s
Source limit:50000B
Languages: All
Resource: A contest of Romanian

SPOJ Problem Set (classical)

705. New Distinct Substrings

Problem code: SUBST1

Given a string, we need to find the total number of its distinct substrings.

Input

T- number of test cases. T<=20; Each test case consists of one string, whose length is <= 50000

Output

For each test case output one number saying the number of distinct substrings.

Example

Input:
```
2
CCCCC
ABABA
```

Output:
```
5
9
```

Added by: Hoang Hong Quan
Date: 2006-01-18
Time limit: 2s
Source limit:50000B
Languages: All
Resource: Base on a problem in ByteCode06

SPOJ Problem Set (classical)

707. Triple-Free Sets

Problem code: TFSETS

A set **S** of positive integers is called *strongly triple-free* if, for any integer **x**, the sets {**x**, 2**x**} and {**x**, 3**x**} are not subsets of **S**. Let's define **F(n)** as a number of strongly triple-free subsets of {1, 2, ..., **n**}, where **n** is a natural number.

You need to write a program which being given a number **n** calculates the number **F(n)** modulo 1 000 000 001.

Input

The first line of input contains integer **T** (1 <= **T** <= 500) - the number of testcases. Then descriptions of **T** testcases follow.

The description of the testcase consists of one line. The line contains an integer number **n** (1 <= **n** <= 100 000).

Output

For each testcase in the input your program should output one line. This line should contain one integer number which is the number **F(n)** modulo 1 000 000 001.

Example

Input:
```
5
3
1
10
20
39
```

Output:
```
5
2
198
43776
971827200
```

Added by: Ivan Metelsky
Date: 2006-01-19
Time limit: 15s
Source limit:50000B
Languages: All
Resource: Based on a problem from www.test-the-best.by

SPOJ Problem Set (classical)

709. The day of the competitors

Problem code: NICEDAY

The International Olympiad in Informatics is coming and the leaders of the Vietnamese Team have to choose the best contestants all over the country. Fortunately, the leaders could choose the members of the team among N very good contestants, numbered from 1 to N (3 <= N <= 100000). In order to select the best contestants the leaders organized three competitions. Each of the N contestants took part in all three competitions and there were no two contestants with equal results on any of the competitions. We say that contestant A is better than another contestant V when A is ranked before V in all of the competitions. A contestant A is said to be excellent if no other contestant is better than A. The leaders of the Vietnamese Team would like to know the number of excellent contestants.

Input

First line of the input contains an integer t (1 <= t <= 10), equal to the number of testcases. Then descriptions of t testcases follow. First line of description contains the number of competitors N . Each of the next N lines describes one competitor and contains integer numbers ai, bi, ci (1 <= ai, bi , ci <= N) separated by spaces, the order of i-th competitor's ranking in the first competition , the second competition and the third competition.

Output

For each test case in the input your program should output the number of excellent contestants in one line.

Note : Because the input is too large so we have 4 input files and the total time limit is 4s (not 1s).

Example

Input:

```
1
3
1 2 3
2 3 1
3 1 2
```

Output:

```
3
```

Added by:	Nguyen Minh Hieu
Date:	2006-01-20
Time limit:	1s
Source limit:	10000B
Languages:	All
Resource:	Base on a problem from BOI

SPOJ Problem Set (classical)

726. Promotion

Problem code: PRO

A large Bytelandian supermarket chain has asked you to write a program for the simulating costs of a promotion being prepared.

The promotion has to follow the following rules:

- A customer who wants to participate in the promotion, writes on the receipt, paid by himself, his personal details and throws it into a special ballot box.
- At the end of every day of the promotion, two bills are taken out from the ballot box:
  - first, the receipt amounting to the largest sum is chosen,
  - then the receipt amounting to the smallest sum is chosen;

  The customer who has paid the largest sum gets a money prize equal to the difference between the sum on his bill and the sum on the bill amounting to the smallest sum.
- To avoid multiple prizes for one purchase, both bills selected according to the above rules are not returned to the ballot box, but all remaining bills still participate in the promotion.

The turnover of the supermarket is very big, thus an assumption can be made, that at the end of every day, before taking out receipts amounting to the largest and the smallest sum, there are at least 2 receipts in the ballot box.

Your task is to compute (on the basis of information about prices on receipts thrown into the ballot box on each day of promotion) what the total cost of prizes during the whole promotion will be.

Write a program, which: reads from the standard input a list of prices on receipts thrown into the ballot box on each day of the promotion, computes the total cost of prizes paid in consecutive days of promotion, then writes the result to the standard output.

Input

The first line of the input contains one positive integer n (1 <= n <= 5000), which is the duration of promotion in days. Each of the next n lines consists of a sequence of non-negative integers separated by single spaces. Numbers in the (i+1)-th line of the file represent prices on receipts thrown into the ballot box on the i-th day of promotion. The first integer in the line is k, 0 <= k <= 10^5, the number of receipts on the day, and the next k numbers are positive integers standing for the sums on receipts; none of these numbers is larger than 10^6.

The total number of bills thrown into the ballot box during the whole promotion does not exceed 10^6.

Output

The output should contain exactly one integer, equal to the total cost of prizes paid during the whole promotion.

Example

Input:
```
5
3 1 2 3
2 1 1
4 10 5 5 1
0
1 2
```

Output:
```
19
```

Added by: Walrus
Date: 2006-01-24
Time limit: 1s-4s
Source limit: 50000B
Languages: All
Resource: VII Polish Olympiad In Informatics 2000, stage III

SPOJ Problem Set (classical)

729. Move your armies

Problem code: MAXIMUS

Commodus has discovered with your help that the traitor is Maximus. Commodus has gathered N prestigious armies A1 A2 ... AN and asked you to lead them to kill Maximus. A brave warrior like you must now act intelligently to lead the armies to victory.

There are three countries which are considered here, for simplicity lets name them C_0, C_1 and C_2. You have moved the armies to C_0 and you know that Maximus is in C_2. You are wise enough to know that without all your N armies you stand no chance against great Maximus. The problem is that your armies are too egoistic in nature (after all they were organized by Commodus). Only the biggest army can leave any country C_y (Army A_x can leave C_y, if there is no army A_i in C_y with $i > x$.). Also, the army A_x will go into C_y only if it is the biggest army to get there, i. e. there is no army A_i in C_y with $i > x$.

There is another confusion here, all the armies A_m have been trained by a different commander and they march differently. Each army A_m where m is either 1 or prime can only move from C_i to $C_{(i+1)\%3}$, while your armies A_m where $m > 1$ is composite will march only from C_i to $C_{(i+2)\%3}$.

Commodus is impatient and he is asking you to find the number of moves you need to reach Maximus. You are planning to reach there with the shortest possible number of moves; tell your answer to Commodus.

Example for N = 2:
The required number of steps would be 7
initially
C0 - A1, A2
C1 -
C2 -
after step 1
C0 - A1
C1 - A2
C2 -
after step 2
C0 - A1
C1 -
C2 - A2
after step 3
C0 -
C1 - A1
C2 - A2
after step 4
C0 - A2
C1 - A1
C2 -

after step 5
C0 - A2
C1 -
C2 - A1
after step 6
C0 -
C1 - A2
C2 - A1
after step 7
C0 -
C1 -
C2 - A1, A2

Input

The input will consist of at most 100 test cases. Each test case consists of a number N (the number of armies, 1 <= N <= 5000). The last test case is followed by a line containing 0.

Output

For each number N, you have to output the number of moves needed to move the armies to C_2 with the minimum number of steps.

Example

Input:

```
1
2
3
4
100
0
```

Output:

```
2
7
21
49
1299510268586153115889930564780511199231
```

Added by: Adrian Kuegel
Date: 2006-01-29
Time limit: 10s
Source limit: 10000B
Languages: All
Resource: Codearena 2006

SPOJ Problem Set (classical)

734. Ivan and his interesting game

Problem code: IVAN

Little Ivan likes to play games in his spare time. Unfortunately, he cannot always enjoy the company of his friends and sometimes he is a little bored when he is alone. Therefore, he makes up games, where he is the only player. He is especially proud of his last game and likes to tell you about it.
You are given two finite sequences of positive integers. The game consists of making consecutive moves. You are allowed to make the following move. You remove the last K1 numbers (K1>=1) from the first sequence (possibly the whole sequence) and find their sum S1 and the last K2 numbers (K2>=1) from the second sequence (again you can remove the whole sequence) and find their sum S2. Then you calculate the cost of the move to be (S1 - K1)*(S2 - K2). You continue to make moves until you remove all the numbers in both sequences. The total cost of the game is the sum of the costs of all moves. Your goal is to minimize this total cost. You are not allowed to leave one of the sequences empty, while the other is not.
As Ivan has told you the rules of the game, you realize that it is easily solvable with the help of a computer, so you decide to write a program GAME, that computes the minimum total cost of the game.

Input

Input data is read from the standard input and consists of three lines. The first line contains two space-separated integers, L1 and L2 (1 <= L1, L2 <= 2000), which denote the lengths of the two sequences. The second line contains L1 space-separated integers, which are the elements of the first sequence. The third line contains L2 space-separated integers, which are the elements of the second sequence. The elements of the sequences do not exceed 1000.

Output

Your program has to output one line on the standard output that contains only one number - the minimum total cost of the game as described above.

Example

Input:
```
3 2
1 2 3
1 2
```
Output:
```
2
```

Added by:	VOJ problem setters
Date:	2006-02-08
Time limit:	1s-2s
Source limit:	10000B
Languages:	All
Resource:	unknown

SPOJ Problem Set (classical)

735. Minimum Diameter Spanning Tree

Problem code: MDST

Solve the minimum diameter spanning tree problem for the simple graphs.

For a given list of adjacent vertices of a graph G find the minimum diameter spanning tree T and write down the diameter of this tree diam(T).

Each graph has only one connected component, so there is at least one spanning tree, which connects all the vertices.

Input

t [the number of test graphs]
> Graph:
> *n* [1 <= *n* <= 1000 the number of graph vertices]
> *i m* $v_{1<sub>} v_2 \ldots v_m$ *[the list of m adjacent vertices to vertex i]*

Output

For each test case output:
> d<i> [diameter of the minimum diameter spanning tree]

Example

Input:

```
6

10
1 3 2 3 4
2 3 1 5 7
3 3 1 5 6
4 3 1 6 8
5 3 2 3 9
6 3 3 4 10
7 1 2
8 1 4
9 1 5
10 1 6

10
1 4 4 5 7 9
2 1 8
3 4 4 7 8 10
4 3 1 3 9
5 2 1 9
6 2 8 9
7 4 1 3 8 9
8 5 2 3 6 7 9
9 7 1 4 5 6 7 8 10
```

```
10 2 3 9

1
1 0

2
1 1 2
2 1 1

3
1 1 2
2 2 1 3
3 1 2

5
1 2 2 4
2 3 1 3 4
3 1 2
4 3 2 5 1
5 1 4
```

Output:

```
5
3
0
1
2
3
```

Added by: *Bartłomiej Kowalski*
Date: *2006-02-09*
Time limit: *1s-25s*
Source limit: *50000B*
Languages: *All*
Resource:

SPOJ Problem Set (classical)

738. Another Counting Problem

Problem code: TREE

Tree is an important data structure in Computer Science. Of all trees we work with, Binary Tree is probably the most popular one. A Binary Tree is called a **Strictly Binary Tree** if every nonleaf node in a binary tree has nonempty left and right subtrees. Let us define **a Strictly Binary Tree of depth d**, as a Strictly Binary Tree that has at least one root to leaf path of length **d**, and no root to leaf path in that tree is longer than **d**. So let us use a similar reasoning to define a generalized structure.

An n-ary Tree is called a **Strictly n-ary Tree** if every nonleaf node in an n-ary tree has n children each. A **Strictly n-ary Tree of depth d** can now be defined as a Strictly n-ary Tree that has at least one root to leaf path of length d, and no root to leaf path in that tree is longer than d.

Given the value of n and depth d, your task is to find the number of different strictly n-ary trees of depth d.

The figure below shows the 3 different strictly binary trees of depth 2.

[IMAGE]

Input

Input consists of several test cases. Each test case consists of two integers **n (0 < n <= 32), d (0 <= d <= 16)**. Input is terminated a test case where n=0 and d=0, you must not process this test case.

Output

For each test case, print three integers, n, d and the number of different strictly n-ary trees of level d, in a single line. There will be a single space in between two integers of a line. You can assume that you would not be asked about cases where you had to consider trees that may have more than 2^{10} nodes in a level of the tree. You may also find it useful to know that the answer for each test case will always fit in a 200 digit integer.

Example

Input:
```
2 0
2 1
2 2
2 3
3 5
0 0
```

Output:
```
2 0 1
```

```
2 1 1
2 2 3
2 3 21
3 5 588715871622705926450340001
```

Added by: Nguyen Van Quang Huy
Date: 2006-02-14
Time limit: 1s
Source limit:50000B
Languages: All
Resource: acm.uva.es

SPOJ Problem Set (classical)

739. The Moronic Cowmpouter

Problem code: NEG2

Inexperienced in the digital arts, the cows tried to build a calculating engine (yes, it's a cowmpouter) using binary numbers (base 2) but instead built one based on base negative 2! They were quite pleased since numbers expressed in base -2 do not have a sign bit.

You know number bases have place values that start at 1 (base to the 0 power) and proceed right-to-left to base^1, base^2, and so on. In base -2, the place values are 1, -2, 4, -8, 16, -32, ... (reading from right to left). Thus, counting from 1 goes like this: 1, 110, 111, 100, 101, 11010, 11011, 11000, 11001, and so on.

Eerily, negative numbers are also represented with 1's and 0's but no sign. Consider counting from -1 downward: 11, 10, 1101, 1100, 1111, and so on.

Please help the cows convert ordinary decimal integers (range -2,000,000,000 .. 2,000,000,000) to their counterpart representation in base -2.

Input

A single integer to be converted to base -2

Output

A single integer with no leading zeroes that is the input integer converted to base -2. The value 0 is expressed as 0, with exactly one 0.

Example

Input:
```
-13
```

Output:
```
110111
```

Added by: Nguyen Van Quang Huy
Date: 2006-02-15
Time limit: 1s
Source limit:50000B
Languages: All
Resource: USACO FEB06 Bronze Division

SPOJ Problem Set (classical)

740. Treats for the Cows

Problem code: TRT

FJ has purchased N (1 <= N <= 2000) yummy treats for the cows who get money for giving vast amounts of milk. FJ sells one treat per day and wants to maximize the money he receives over a given period time. The treats are interesting for many reasons:

- The treats are numbered 1..N and stored sequentially in single file in a long box that is open at both ends. On any day, FJ can retrieve one treat from either end of his stash of treats.
- Like fine wines and delicious cheeses, the treats improve with age and command greater prices.
- The treats are not uniform: some are better and have higher intrinsic value. Treat i has value v(i) (1 <= v(i) <= 1000).
- Cows pay more for treats that have aged longer: a cow will pay v(i)*a for a treat of age a.

Given the values v(i) of each of the treats lined up in order of the index i in their box, what is the greatest value FJ can receive for them if he orders their sale optimally?

The first treat is sold on day 1 and has age a=1. Each subsequent day increases the age by 1.

Input

Line 1: A single integer, N

Lines 2..N+1: Line i+1 contains the value of treat v(i)

Output

The maximum revenue FJ can achieve by selling the treats

Example

Input:
```
5
1
3
1
5
2
```

Output:
```
43
```

Added by: Nguyen Van Quang Huy
Date: 2006-02-15
Time limit: 1s
Source limit:50000B
Languages: All
Resource: USACO FEB06 Gold Division

SPOJ Problem Set (classical)

741. Steady Cow Assignment

Problem code: STEAD

Farmer John's N (1 <= N <= 1000) cows each reside in one of B (1 <= B <= 20) barns which, of course, have limited capacity. Some cows really like their current barn, and some are not so happy.

FJ would like to rearrange the cows such that the cows are as equally happy as possible, even if that means all the cows hate their assigned barn.

Each cow gives FJ the order in which she prefers the barns. A cow's happiness with a particular assignment is her ranking of her barn. Your job is to find an assignment of cows to barns such that no barn's capacity is exceeded and the size of the range (i.e., one more than the positive difference between the the highest-ranked barn chosen and that lowest-ranked barn chosen) of barn rankings the cows give their assigned barns is as small as possible.

Input

Line 1: Two space-separated integers, N and B

Lines 2..N+1: Each line contains B space-separated integers which are exactly 1..B sorted into some order. The first integer on line i+1 is the number of the cow i's top-choice barn, the second integer on that line is the number of the i'th cow's second-choice barn, and so on.

Line N+2: B space-separated integers, respectively the capacity of the first barn, then the capacity of the second, and so on. The sum of these numbers is guaranteed to be at least N.

Output

One integer, the size of the minimum range of barn rankings the cows give their assigned barns, including the endpoints

Example

Input:
```
6 4
1 2 3 4
2 3 1 4
4 2 3 1
3 1 2 4
1 3 4 2
1 4 2 3
2 1 3 2
```

Output:
```
2
```

Added by: Nguyen Van Quang Huy
Date: 2006-02-16
Time limit: 2s
Source limit:50000B
Languages: All
Resource: USACO FEB06 Gold Division

SPOJ Problem Set (classical)

744. Longest Permutation

Problem code: LPERMUT

You are given a sequence A of n integer numbers ($1<=A_i<=n$). A subsequence of A has the form A_u, A_{u+1} ... , A_v ($1<=u<=v<=n$). We are interested in subsequences that are permutations of 1, 2, .., k (k is the length of the subsequence).

Your task is to find the longest subsequence of this type.

Input

- Line 1: n ($1<=n<=100000$)
- Line 2: n numbers A_1, A_2, ... ,A_n ($1<=A_i<=n$)

Output

A single integer that is the length of the longest permutation

Example

Input:
```
5
4 1 3 1 2
```

Output:
```
3
```

Added by: Walrus
Date: 2006-02-20
Time limit: 1s
Source limit:50000B
Languages: All
Resource: A problem put forward by Mr Mircea Pasoi

SPOJ Problem Set (classical)

757. Thermal Luminescence

Problem code: TEM

After many years of hard work a group of scientists developed a shiny new state-of-the-art processor with a 3D configuration. Due to the high clock frequency at which this processor works, the silicon cube uses up too much energy. Even with its powerful cooling system, the processor is unable to cope with the heat discharge in some of its cubical blocks. With the help of special analysis methods, scientists have developed the overheat rate for each of the cubical blocks of the system. As it conveniently happens, this overheat rate is an integer value, either positive or negative depending on many factors (such as the proximity of ventilators, refrigerators, etc.).

Science can do no more, so now the developers of the processor need your support. For a given three-dimensional matrix representing the overheat rate of elements of the processor, you have to find a submatrix for which the sum of overheat rates coming from all its elements is maximal.

Input

t - number of test cases [*t* <= 99], then t tests follow.
Each test begins with 3 integers: x, y, z - the width, length and height of matrix [5 <= x, y, z <= 50]. Then there follows the description of x rectangular 2D matrixes of height y and width z. In total there are x*y*z integers, which absolute value does not exceed 10000.

Output

For each test case you should output 6 integers: x1, y1, z1, x2, y2, z2, where each triple (xi, yi, zi) defines one of the two opposite corners of submatrix, resulting in the maximum overheat. [1 <= x1 <= x2 <= x] [1 <= y1 <= y2 <= y] [1 <= z1 <= z2 <= z]

Example

Input:

```
1
5 5 5
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
-1 -1 -10 -1 -1
-1 -1 -10 -1 -1
-1 -1 -10 -1 -1
-1 -1 -10 -1 -1
-1 -1 -10 -1 -1
20 2 2 2 20
20 2 2 2 20
20 2 2 2 20
20 2 2 2 20
20 2 2 2 20
5 5 5 5 5
```

```
5 5 5 5 5
1 1 1 1 1
1 1 1 1 1
1 1 1 1 1
-10 10 -10 10 -10
-10 10 -10 10 -10
-10 10 -10 10 -10
-10 10 -10 10 -10
-10 10 -10 10 -10
```

Output:

```
3 1 1 4 5 5
```

Note:

The maximum overheat for the example is equal to 295.

Added by:	Roman Sol
Date:	2005-04-11
Time limit:	15s
Source limit:	50000B
Languages:	All
Resource:	ZCon 2006

SPOJ Problem Set (classical)

760. Convex Hull 3D

Problem code: CH3D

Bytelandian scientists have developed a brand new method for determining the volume of a person's lungs. The idea is simple: the patient is asked to inhale a sufficiently large number of nanobots, which then transmit their exact 3D-coordinates to an external sensor. Early clinical tests proved rather fun (especially for the scientists who were watching the process of nanobot inhalation), but gave rise to several problems of an algorithmic nature. In other words, nobody had any idea of how the volume of the lungs should be determined afterwards. A lung consists of a large number of disjoint alveoli (which can for our purposes be regarded as little hollows), and inhaled nanobots tend to float around aimlessly within the alveolus they happened to fall into. Whereas it is relatively simple to distinguish between different alveoli, establishing the volume of a single alveolus is a tough task.

One way to estimate the shape and volume of an alveolus is to smear all nanobots with a little liquid glue and see what they end up stuck to. Another (arguably more humane) method is to calculate the *convex hull* of the set of points representing nanobot coordinates, its volume and surface area. A convex hull of given set of points in 3D is the convex set of minimum volume which contains all these points.

Lung 1
arrow
Lung 2
arrow
Lung 3

Input

t - number of test cases [*t* <= 100], then *t* tests follow.
Each test starts with integer *N* - the number of given points [10 <= N <= 1000]. Then exactly N lines follow with 3 real numbers Xi, Yi, Zi in each of them, where [-10.0 <= Xi, Yi, Zi <= 10.0].

Output

For each test case you should output 2 real numbers: the surface area and volume of the hull with precision 0.01.

Example

Input:

```
1
10
0.00000 0.00000 0.00000
1.00000 0.00000 0.00000
0.00000 1.00000 0.00000
0.00000 0.00000 1.00000
1.00000 1.00000 0.00000
1.00000 0.00000 1.00000
```

```
0.00000 1.00000 1.00000
1.00000 1.00000 1.00000
0.50000 0.50000 0.50000
0.66666 0.77777 0.88888
```

Output:

```
6.0000 1.0000
```

Added by: Roman Sol
Date: 2005-11-28
Time limit: 8s
Source limit: 50000B
Languages: All
Resource: ZCon 2006

SPOJ Problem Set (main)

764. Delay-noise Analysis

Problem code: MIS

During the development of delay-noise analysis theory, scientists have come upon the following problem. After they had conducted experiments they found out that some of the nodes of the circuit couldn't switch at the same time. For example, if we know that node *N* switches from 0 to 1, then node *K* can't switch at the same moment because of logical restrictions in circuit. Each node of the circuit injects some noise into neighboring nodes while switching, and this noise can be measured. Scientists now need some fast method to calculate the maximum delay-noise that can be injected by switching aggressor-nodes.

Scientists formalize the problem in the following way. We consider a graph *G = (V, E, w)* consisting of vertex set *V*, edges se [IMAGE] , and weight function *W*, such that [IMAGE] and [IMAGE] . For [IMAGE] and [IMAGE] , *N(u)* and *N(K)* denotes the set of neighboring vertices of *u* and *K* accordingly, formally defined as:
[IMAGE]
The set of vertices [IMAGE] satisfying [IMAGE] is called *internally stable*. In this problem you should find a set of internally stable vertices *B* such that *w(B) = max{w(S)}*, taken over all internally stable sets *S* of the given graph *G*.

Experiments have shown that the density of edges in considered graphs is between 20% and 90%.

Input

t - number of test cases [*t* <= 60]
n k - [*n* - number of vertices (2 <= *n* <= 250), *k* - number of edges (1 <= *k* <= *n*(n-1)/2*)]
then *n* integers follow (*wi* - weight of vertex *i*) [0 <= *wi* <= 2^31-1]
then *k* pairs of integers follow, representing the edges between vertices (*si sj* denotes an edge between vertices *i* and *j*) [1 <= *si, sj* <= *n*]. It is known that the total weight of all vertices in the graph doesn't exceed 10^9.

Output

For each test case output *MaxWeight* - the weight of a maximum internally stable set of the given graph [0 <= *MaxWeight* <= 10^9].

Example

Input :

```
2
5 6
10 20 30 40 50
1 2
1 5
2 3
3 4
3 5
```

```
4 5

4 4
10 4 10 14
1 2
2 3
3 4
4 1
```

Output :

```
70
20
```

Example illustrations:[IMAGE]

Added by: Roman Sol
Date: 2005-03-22
Time limit: 35s
Source limit:50000B
Languages: All
Resource: ZCon 2006

SPOJ Problem Set (classical)

780. The Archipelago

Problem code: ARCHPLG

Byteland is a country located in the Archipelago of Rectangular Islands. The archipelago consists of `1<=n<=1000` islands. A fact that each island has a rectangular shape is very nice for Bytelandian cartographers.

Bytelandian islands are rather small and none are very fertile, so each of (rectangular of course) pieces of cultivated land is under special control, simply speaking: 'never enter there to save your life'. Other areas are guaranteed to be free accessible for the people.

The communication between islands is possible by ferries. On each island there is `0<=b<=10` terminals, from where crossings to another terminals on other islands are possible. It is known that total number of crossing connections is `0<=m<=100000`. Other infrastructure is practically unknown. Specifically the only possible way of traveling through the island is to do it on foot.

Well, now we are close to a task you are requested to solve. John - one of the Bytelandian citizens is working as a sales manager. Simply speaking he is often requested to travel from one place to another, what he rather dislike and preferably (like other Bytelandian people use to do) he would like to spent more time in one of the beach clubs playing Puto (a kind of strategic game, very popular in Byteland). Please help him to find a way to spare his time.

Task

Find one of the fastest ways for John using ferries and foot paths on islands. Assume that while walking John is always moving one BM (Bytelandian unit of length) per BH (Bytelandian unit of time). You can also assume that the ferry departures nearly immediately after John arrives the terminal, it will be enough to round up the walking time to the nearest integer.

Input

In the first line `t` - the number of tests, then for each test: in next line `n` - the number of islands. Description of each island is as follows:

```
name
w h [island dimensions]
b – [number of terminals]
[description of each terminal in a form:]
name x y [name of a terminal and its coordinates]
F [number of restricted areas F<20]
xl, yd, xr, yu [coordinates of each restricted area,
0 <=xl < xr<=250 0<=yd < yu<=250.]
```

All coordinates are nonnegative integers measured in BM according to upper left corner of an island.

You can assume that any two restricted areas are disjoint. After the description of all islands all ferry connections are given (each connection is bi-directional).

```
m  [number of connections]
[description of each connection]
NB1 NW1 NB2 NW2 time [name of a first terminal, its island, the second respectively
and communiaction time]
...
[description follows]
...
NBS NWS NBC NWC [start and goal terminal for John]
```

Output

For each test describe the shortest route for John from terminal NBS on NWS island to terminal NBC on NWC island in the following format:

```
case nr Y [nr - test number]
T [travel time in BH]
NBS  NWS
...
[consecutive terminals]
...
NBC NWC
[empty line]
[consecutive tests]
```

If two consecutive terminals are located on the same island and John must take some walk you must give all middle point like in an example.

[IMAGE]

Example

Input:

```
1
3
W1
8 7
2
Lindos 4 0
Kamejros 4 7
3
2 1 6 2
2 3 6 4
2 5 6 6
W2
14 12
2
Malia 14 1
Knossos 1 12
5
2 6 10 10
11 1 12 6
8 1 10 5
11 7 12 9
3 2 5 4
W3
1 1
```

```
1
Korkyra 0 0
0
2
Kamejros W1 Knossos W2 100
Malia W2 Korkyra W3 100
Korkyra W3 Lindos W1
```

An example of a correct answer:

Output:

```
case 1 Y
230
Korkyra W3
Malia W2
12 6
11 7
10 10
Knossos W2
Kamejros W1
2 6
2 1
Lindos W1
```

Added by: Łukasz Kuszner
Date: 2006-03-15
Time limit: 40s
Source limit: 50000B
Languages: All
Resource: GUT Algorithm Analysis 2005

SPOJ Problem Set (classical)

827. Trigonometric optimization

Problem code: TRIOPT

Many problems arising in practical applications may be stated as *optimization problems*. Usually it is necessary to maximize or minimize so called *criterion function* taking into account some *constraints*.

Let's consider a trigonometric optimization problem. It is necessary to maximize or to minimize criterion function $\mathbf{F_1(x) + F_2(y) + F_3(z)}$ with constraint $\mathbf{x + y + z = S}$, where **x**, **y**, **z** - variables, **S** - parameter, **x**, **y**, **z**, **S** - natural numbers. Each of the functions $\mathbf{F_1}$, $\mathbf{F_2}$ and $\mathbf{F_3}$ is a trigonometric function ***sin*** or ***cos***.

You need to write a program which solves the *trigonometric optimization* problem.

Input

The first line of the input data contains integer **T** (1 <= **T** <= 65) - the number of testcases. Then the descriptions of **T** testcases follow.

The description of each testcase consists of *5* lines. The first line describes function $\mathbf{F_1}$ and contains either **sin** or **cos**. The second and the third lines describe functions $\mathbf{F_2}$ and $\mathbf{F_3}$ respectively and have the same format as the first line. Next, the fourth line contains either **min** or **max**. If the line contains **min** than it is necessary to minimize *criterion function*, otherwise it is necessary to maximize *criterion function*. Finally, the fifth line contains parameter **S** value (3 <= **S** <= 1 000 000).

Output

For each testcase you should output one line into the output data. This line should contain one real number - the found value of the *criterion function*. Absolute error of your answer must not exceed $\mathbf{10^{-10}}$.

Example

Input:
```
1
sin
cos
sin
max
10
```

Output:
```
2.7787651403
```

Added by: Ivan Metelsky
Date: 2006-04-27
Time limit: 90s
Source limit: 50000B
Languages: All
Resource: NEERC Western Subregion QF 2005

SPOJ Problem Set (classical)

839. Optimal Marks

Problem code: OPTM

You are given an undirected graph G(V, E). Each vertex has a mark which is an integer from the range $[0..2^{31} - 1]$. Different vertexes may have the same mark.

For an edge (u, v), we define Cost(u, v) = mark[u] xor mark[v].

Now we know the marks of some certain nodes. You have to determine the marks of other nodes so that the total cost of edges is as small as possible.

Input

The first line of the input data contains integer **T** (1 <= **T** <= 10) - the number of testcases. Then the descriptions of T testcases follow.

First line of each testcase contains 2 integers **N** and **M** (0 < **N** <= 500, 0 <= **M** <= 3000). **N** is the number of vertexes and **M** is the number of edges. Then **M** lines describing edges follow, each of them contains two integers u, v representing an edge connecting u and v.

Then an integer **K**, representing the number of nodes whose mark is known. The next **K** lines contain 2 integers u and p each, meaning that node u has a mark p. It's guaranteed that nodes won't duplicate in this part.

Output

For each testcase you should print **N** lines integer the output. The **K**th line contains an integer number representing the mark of node **K**. If there are several solutions, you have to output the one which minimize the sum of marks. If there are several solutions, just output any of them.

Example

Input:

```
1
3 2
1 2
2 3
2
1 5
3 100
```

Output:

```
5
4
100
```

Added by:	Thanh-Vy Hua
Date:	2006-05-05
Time limit:	6s
Source limit:	10000B
Languages:	All
Resource:	Guo HuaYang

SPOJ Problem Set (classical)

850. Soccer Choreography

Problem code: WM06

Mr. Bitmann, the coach of the national soccer team of Bitland, is a perfectionist. He taught his players optimal tactics and improved their endurance and shape. So they qualified for the soccer woldcup this year. Due to his perfectionism the coach attaches importance not only to the performance in the game but also before the game. So he told the team captain in what formation the team should assemble before the national anthem is played. Since each of the 11 team members has a unique number between 1 and 11 on his shirt, he can represent the formation as a permutation of numbers.

Before the first game the coach told the captain that the team should line up in increasing order (picture (d)). But some players forgot the ordering and the orientation of the formation like in picture (a). Only player 1 has the right orientation. The coach went nearly mad when he saw this desaster! How could he solve the problem?

scenario example

"Hmmm... I'll let my players dance!". A great idea! He took his notebook and started to create a choreography which leads to his expected formation. Due to the fact that no one of the players took dancing lessons he restricts his dance to one basic move: One player or more players who stand side by side can turn 180 degrees around the center of the move. Picture (b) contains an example: The players

```
-11 -10 -9 -2
```

(we mark players which stand in the wrong direction with a minus) can do one move to

```
2 9 10 11
```

As perfect as he is he calculated a dance with a minimum number of moves. It works perfectly and now he's planning to do dancing performances with teams with more than 11 members. So he needs your help to find optimal dancing moves...

Input

Each testcase starts with the number of team members n (0<=n< 2200). The next lines represent the formation at the beginning and the expected formation at the end of the choreography.

Output

For each testcase output m, the minimal number of moves which are necessary to reach the expected formation. The next m+1 lines should represent one possible scenario of moves.

Example

Input:

```
11
-5 -4 -3 -8 -7 -6 1 -11 -10 -9 -2
1 2 3 4 5 6 7 8 9 10 11
11
1 2 3 -4 -5 -6 -7 -8 -9 10 11
11 9 8 7 6 5 4 3 2 10 1
0
```

Output:

```
3 Steps
-5 -4 -3 -8 -7 -6 +1 -11 -10 -9 -2
-5 -4 -3 -8 -7 -6 +1 +2 +9 +10 +11
-5 -4 -3 -2 -1 +6 +7 +8 +9 +10 +11
+1 +2 +3 +4 +5 +6 +7 +8 +9 +10 +11
5 Steps
+1 +2 +3 -4 -5 -6 -7 -8 -9 +10 +11
+1 +2 -3 -4 -5 -6 -7 -8 -9 +10 +11
+1 -2 -3 -4 -5 -6 -7 -8 -9 +10 +11
+1 -2 -3 -4 -5 -6 -7 -8 -9 -11 -10
+11 +9 +8 +7 +6 +5 +4 +3 +2 -1 -10
+11 +9 +8 +7 +6 +5 +4 +3 +2 +10 +1
```

Added by: Simon Gog
Date: 2006-05-11
Time limit: 1s-2s
Source limit:50000B
Languages: All

SPOJ Problem Set (classical)

861. Counting inversions

Problem code: SWAPS

You are given a sequence A of **N** (**N**<=250000) integers between 1 and 50000. On this sequence you have to apply **M** (**M**<=10000) operations of the form: modify the i-th element in the sequence and then say how many inversions are there in the sequence. The number of inversions in a sequence is given by the number of pairs (i,j) with i < j and Ai > Aj.

Input

The first line of input contains the number **N** and the next line contains the numbers that form the sequence. After that follows the number **M** and then **M** lines, each containig 2 integers X and Y, meaning that new value of the X-th element of the sequence is Y and that you should count the number of inversions in the modified sequence.

Output

Output must contain **M** lines, the i-th line of output containg the number of inversions in the sequence after the first i operations.

Example

Input:

```
10
2 6 6 4 7 6 3 5 9 1
7
8 8
5 1
5 6
10 5
7 1
10 10
4 6
```

Output:

```
17
18
16
13
14
8
6
```

Added by: Gogu Marian
Date: 2006-05-31
Time limit: 1s-5s
Source limit:50000B
Languages: All
Resource:

SPOJ Problem Set (classical)

866. DNA Translation

Problem code: DNA

Deoxyribonucleic acid (DNA) is composed of a sequence of nucleotide bases paired together to form a double-stranded helix structure. Through a series of complex biochemical processes the nucleotide sequences in an organism's DNA are translated into the proteins it requires for life. The object of this problem is to write a computer program which accepts a DNA strand and reports the protein generated, if any, from the DNA strand.

The nucleotide bases from which DNA is built are adenine, cytosine, guanine, and thymine (hereafter referred to as A, C, G, and T, respectively). These bases bond together in a chain to form half of a DNA strand. The other half of the DNA strand is a similar chain, but each nucleotide is replaced by its complementary base. The bases A and T are complementary, as are the bases C and G. These two "half-strands" of DNA are then bonded by the pairing of the complementary bases to form a strand of DNA.

Typically a DNA strand is listed by simply writing down the bases which form the primary strand (the complementary strand can always be created by writing the complements of the bases in the primary strand). For example, the sequence TACTCGTAATTCACT represents a DNA strand whose complement would be ATGAGCATTAAGTGA. Note that A is always paired with T, and C is always paired with G.

From a primary strand of DNA, a strand of ribonucleic acid (RNA) known as messenger RNA (mRNA for short) is produced in a process known as transcription. The transcribed mRNA is identical to the complementary DNA strand with the exception that thymine is replaced by a nucleotide known as uracil (hereafter referred to as U). For example, the mRNA strand for the DNA in the previous paragraph would be AUGAGCAUUAAGUGA.

It is the sequence of bases in the mRNA which determines the protein that will be synthesized. The bases in the mRNA can be viewed as a collection of codons, each codon having exactly three bases. The codon AUG marks the start of a protein sequence, and any of the codons UAA, UAG, or UGA marks the end of the sequence. The one or more codons between the start and termination codons represent the sequence of amino acids to be synthesized to form a protein. For example, the mRNA codon AGC corresponds to the amino acid serine (Ser), AUU corresponds to isoleucine (Ile), and AAG corresponds to lysine (Lys). So, the protein formed from the example mRNA in the previous paragraph is, in its abbreviated form, Ser-Ile-Lys.

The complete genetic code from which codons are translated into amino acids is shown in the table below (note that only the amino acid abbreviations are shown). It should also be noted that the sequence AUG, which has already been identified as the start sequence, can also correspond to the amino acid methionine (Met). So, the first AUG in a mRNA strand is the start sequence, but subsequent AUG codons are translated normally into the Met amino acid.

First base in codon	*Second base in codon*				*Third base in codon*
	U	**C**	**A**	**G**	
U	Phe	Ser	Tyr	Cys	**U**
	Phe	Ser	Tyr	Cys	**C**
	Leu	Ser	---	---	**A**
	Leu	Ser	---	Trp	**G**
C	Leu	Pro	His	Arg	**U**
	Leu	Pro	His	Arg	**C**
	Leu	Pro	Gln	Arg	**A**
	Leu	Pro	Gln	Arg	**G**
A	Ile	Thr	Asn	Ser	**U**
	Ile	Thr	Asn	Ser	**C**
	Ile	Thr	Lys	Arg	**A**
	Met	Thr	Lys	Arg	**G**
G	Val	Ala	Asp	Gly	**U**
	Val	Ala	Asp	Gly	**C**
	Val	Ala	Glu	Gly	**A**
	Val	Ala	Glu	Gly	**G**

The input for this program consists of strands of DNA sequences, one strand per line, from which the protein it generates, if any, should be determined and output. The given DNA strand may be either the primary or the complementary DNA strand, and it may appear in either forward or reverse order, and the start and termination sequences do not necessarily appear at the ends of the strand. For example, a given input DNA strand to form the protein Ser-Ile-Lys could be any of ATACTCGTAATTCACTCC, CCTCACTTAATGCTCATA, TATGAGCATTAAGTGAGG, or GGAGTGAATTACGAGTAT. The input file will be terminated by a line containing a single asterisk character.

You may assume the input to contain only valid, upper-case, DNA nucleotide base letters (A, C, G, and T). No input line will exceed 255 characters in length. There will be no blank lines or spaces in the input. Some sequences, though valid DNA strands, do not produce valid protein sequences; the string "*** No translatable DNA found ***" should be output when an input DNA strand does not translate into a valid protein.

Input:

```
ATACTCGTAATTCACTCC
CACCTGTACACAGAGGTAACTTAG
TTAATACGACATAATTAT
GCCTTGATATGGAGAACTCATTAGATA
AAGTGTATGTTGAATTATATAAAACGGGCATGA
ATGATGATGGCTTGA
*
```

Output:

```
Ser-Ile-Lys
Cys-Leu-His
Ser-Tyr
*** No translatable DNA found ***
Leu-Asn-Tyr-Ile-Lys-Arg-Ala
Met-Met-Ala
```

Added by: Wanderley Guimaraes
Date: 2006-06-01
Time limit: 1s
Source limit: 50000B
Languages: All
Resource: ACM Mid Central Regionals 1995

SPOJ Problem Set (classical)

867. Perfect Cubes

Problem code: CUBES

For hundreds of years Fermat's Last Theorem, which stated simply that for $n > 2$ there exist no integers a, b, $c > 1$ such that $a^n = b^n + c^n$, has remained elusively unproven. (A recent proof is believed to be correct, though it is still undergoing scrutiny.) It *is* possible, however, to find integers greater than 1 that satisfy the "perfect cube" equation $a^3 = b^3 + c^3 + d^3$ (e.g. a quick calculation will show that the equation $12^3 = 6^3 + 8^3 + 10^3$ is indeed true). This problem requires that you write a program to find all sets of numbers $\{a,b,c,d\}$ which satisfy this equation for $a <= 100$.

The output should be listed as shown below, one perfect cube per line, in non-decreasing order of a (i.e. the lines should be sorted by their a values). The values of b, c, and d should also be listed in non-decreasing order on the line itself. There do exist several values of a which can be produced from multiple distinct sets of b, c, and d triples. In these cases, the triples with the smaller b values should be listed first.

Note that the programmer will need to be concerned with an efficient implementation. The official time limit for this problem is 2 minutes, and it is indeed possible to write a solution to this problem which executes in under 2 minutes on a 33 MHz 80386 machine. Due to the distributed nature of the contest in this region, judges have been instructed to make the official time limit at their site the greater of 2 minutes or twice the time taken by the judge's solution on the machine being used to judge this problem.

The first part of the output is shown here:

```
Cube = 6, Triple = (3,4,5)
Cube = 12, Triple = (6,8,10)
Cube = 18, Triple = (2,12,16)
Cube = 18, Triple = (9,12,15)
Cube = 19, Triple = (3,10,18)
Cube = 20, Triple = (7,14,17)
Cube = 24, Triple = (12,16,20)
```

Added by: Wanderley Guimaraes
Date: 2006-06-01
Time limit: 1s
Source limit:50000B
Languages: All
Resource: ACM Mid Central Regionals 1995

SPOJ Problem Set (classical)

869. Galactic Import

Problem code: IMPORT

With the introduction of the new ThrustoZoom gigadimensional drive, it has become possible for HyperCommodities, the import/export conglomerate from New Jersey, to begin trading with even the most remote galaxies in the universe. HyperCommodities wants to import goods from some of the galaxies in the Plural Z sector. Planets within these galaxies export valuable products and raw materials like vacuuseal, transparent aluminum, digraphite, and quantum steel. Preliminary reports have revealed the following facts:

- Each galaxy contains at least one and at most 26 planets. Each planet within a galaxy is identified by a unique letter from A to Z.
- Each planet specializes in the production and export of one good. Different planets within the same galaxy export different goods.
- Some pairs of planets are connected by hyperspace shipping lines. If planets A and B are connected, they can trade goods freely. If planet C is connected to B but not to A, then A and C can still trade goods with each other through B, but B keeps 5% of the shipment as a shipping fee. (Thus A only receives 95% of what C shipped, and C receives only 95% of what A shipped.) In general, any two planets can trade goods as long as they are connected by some set of shipping lines, but each intermediate planet along the shipping route keeps 5% of what it shipped (which is not necessarily equal to 5% of the original shipment).
- At least one planet in each galaxy is willing to open a ThrustoZoom shipping line to Earth. A ThrustoZoom line is the same as any other shipping line within the galaxy, as far as business is concerned. For example, if planet K opens a ThrustoZoom line to Earth, then the Earth can trade goods freely with K, or it can trade goods with any planet connected to K, subject to the usual shipping fees.

HyperCommodities has assigned a relative value (a positive real number less than 10) to each planet's chief export. The higher the number, the more valuable the product. More valuable products can be resold with a higher profit margin in domestic markets. The problem is to determine which planet has the most valuable export when shipping fees are taken into account.

The input consists of one or more galaxy descriptions. Each galaxy description begins with a line containing an integer *N* which specifies the number of planets in the galaxy. The next *N* lines contain descriptions of each planet, which consist of:

1. The letter used to represent the planet.
2. A space.
3. The relative value of the planet's export, in the form *d.dd*.

4. A space.

5. A string containing letters and/or the character '*'; a letter indicates a shipping line to that planet, and a '*' indicates a willingness to open a ThrustoZoom shipping line to Earth.

For each galaxy description, output a single line which reads "Import from P" where P is the letter of the planet with the most valuable export, once shipping fees have been taken into account. (If more than one planet have the same most valuable export value then output the plant which is alphabetically first).

A sample input file is shown here:

```
1
F 0.81 *
5
E 0.01 *A
D 0.01 A*
C 0.01 *A
A 1.00 EDCB
B 0.01 A*
10
S 2.23 Q*
A 9.76 C
K 5.88 MI
E 7.54 GC
M 5.01 OK
G 7.43 IE
I 6.09 KG
C 8.42 EA
O 4.55 QM
Q 3.21 SO
```

The following output file should be produced from the above sample input:

```
Import from F
Import from A
Import from A
```

Added by: Wanderley Guimaraes
Date: 2006-06-01
Time limit: 1s
Source limit:50000B
Languages: All
Resource: ACM Mid Central Regionals 1995

SPOJ Problem Set (classical)

870. Basically Speaking

Problem code: BASE

The Really Neato Calculator Company, Inc. has recently hired your team to help design their Super Neato Model I calculator. As a computer scientist you suggested to the company that it would be neato if this new calculator could convert among number bases. The company thought this was a stupendous idea and has asked your team to come up with the prototype program for doing base conversion. The project manager of the Super Neato Model I calculator has informed you that the calculator will have the following neato features:

- It will have a 7-digital display.
- Its buttons will include the capital letters A through F in addition to the digits 0 through 9.
- It will support bases 2 through 16.

The input for your prototype program will consist of one base conversion per line. There will be three numbers per line. The first number will be the number in the base you are converting from. The second number is the base you are converting from. The third number is the base you are converting to. There will be one or more blanks surrounding (on either side of) the numbers. There are several lines of input and your program should continue to read until the end of file is reached.

The output will only be the converted number as it would appear on the display of the calculator. The number should be right justified in the 7-digit display. If the number is to large to appear on the display, then print ‘‘ERROR’’ (without the quotes) right justified in the display.

A sample input file is shown here:

```
1111000  2 10
1111000  2 16
2102101  3 10
2102101  3 15
  12312  4  2
     1A 15  2
1234567 10 16
   ABCD 16 15
```

The following output file should be produced from the above sample input:

```
    120
     78
   1765
    7CA
  ERROR
  11001
 12D687
   D071
```

Added by: Wanderley Guimaraes
Date: 2006-06-01
Time limit: 1s
Source limit:50000B
Languages: All
Resource: ACM Mid Central Regionals 1995

SPOJ Problem Set (classical)

871. Letter Sequence Analysis

Problem code: SEQUENCE

Cryptographic analysis makes extensive use of the frequency with which letters and letter sequences occur in a language. If an encrypted text is known to be in english, for example, a great deal can be learned from the fact that the letters E, L, N, R, S, and T are the most common ones used in written english. Even more can be learned if common letter pairs, triplets, etc. are known.

For this problem you are to write a program which accepts as input a text file of unspecified length and performs letter sequence analysis on the text. The program will report the five most frequent letter sequences for each set of sequences from one to five letters. That is it will report the individual characters which occur with the five highest frequencies, the pairs of characters which occur with the five highest frequencies, and so on up to the letter sequences of five characters which occur with the five highest frequencies.

The program should consider contiguous sequences of alphabetic characters only, and case should be ignored (e.g. an ‘a’ is the same as an ‘A’). A report should be produced using the format shown in the example at the end of this problem description. For each sequence length from one to five, the report should list the sequences in descending order of frequency. If there are several sequences with the same frequency then all sequences should be listed in alphabetical order as shown (list all sequences in upper case). Finally, if there are less than five distinct frequencies for a particular sequence length, simply report as many distinct frequency lists as possible.

When a text file containing simply the line ‘‘Peter Piper Picks Pickles!’’ is used as input, the output should appear as shown here:

```
Analysis for Letter Sequences of Length 1
----------------------------------------
Frequency = 5, Sequence(s) = (P)
Frequency = 4, Sequence(s) = (E)
Frequency = 3, Sequence(s) = (I)
Frequency = 2, Sequence(s) = (C,K,R,S)
Frequency = 1, Sequence(s) = (L,T)

Analysis for Letter Sequences of Length 2
----------------------------------------
Frequency = 3, Sequence(s) = (PI)
Frequency = 2, Sequence(s) = (CK,ER,IC,PE)
Frequency = 1, Sequence(s) = (ES,ET,IP,KL,KS,LE,TE)

Analysis for Letter Sequences of Length 3
----------------------------------------
Frequency = 2, Sequence(s) = (ICK,PIC)
Frequency = 1, Sequence(s) = (CKL,CKS,ETE,IPE,KLE,LES,PER,PET,PIP,TER)

Analysis for Letter Sequences of Length 4
----------------------------------------
Frequency = 2, Sequence(s) = (PICK)
Frequency = 1, Sequence(s) = (CKLE,ETER,ICKL,ICKS,IPER,KLES,PETE,PIPE)
```

```
Analysis for Letter Sequences of Length 5
----------------------------------------
Frequency = 1, Sequence(s) = (CKLES,ICKLE,PETER,PICKL,PICKS,PIPER)
```

When the first three paragraphs of this problem description are used as input, the output should appear as shown here:

```
Analysis for Letter Sequences of Length 1
----------------------------------------
Frequency = 201, Sequence(s) = (E)
Frequency = 112, Sequence(s) = (T)
Frequency = 96, Sequence(s) = (S)
Frequency = 90, Sequence(s) = (R)
Frequency = 84, Sequence(s) = (N)

Analysis for Letter Sequences of Length 2
----------------------------------------
Frequency = 37, Sequence(s) = (TH)
Frequency = 33, Sequence(s) = (EN)
Frequency = 27, Sequence(s) = (HE)
Frequency = 24, Sequence(s) = (RE)
Frequency = 23, Sequence(s) = (NC)

Analysis for Letter Sequences of Length 3
----------------------------------------
Frequency = 24, Sequence(s) = (THE)
Frequency = 21, Sequence(s) = (ENC,EQU,QUE,UEN)
Frequency = 12, Sequence(s) = (NCE,SEQ,TER)
Frequency = 9, Sequence(s) = (CES,FRE,IVE,LET,REQ,TTE)
Frequency = 8, Sequence(s) = (ETT,FIV)

Analysis for Letter Sequences of Length 4
----------------------------------------
Frequency = 21, Sequence(s) = (EQUE,QUEN)
Frequency = 20, Sequence(s) = (UENC)
Frequency = 12, Sequence(s) = (ENCE,SEQU)
Frequency = 9, Sequence(s) = (FREQ,NCES,REQU)
Frequency = 8, Sequence(s) = (ETTE,FIVE,LETT,TTER)

Analysis for Letter Sequences of Length 5
----------------------------------------
Frequency = 21, Sequence(s) = (EQUEN)
Frequency = 20, Sequence(s) = (QUENC)
Frequency = 12, Sequence(s) = (SEQUE,UENCE)
Frequency = 9, Sequence(s) = (ENCES,FREQU,REQUE)
Frequency = 8, Sequence(s) = (ETTER,LETTE)
```

Added by: Wanderley Guimaraes
Date: 2006-06-01
Time limit: 1s
Source limit:50000B
Languages: All
Resource: ACM Mid Central Regionals 1995

SPOJ Problem Set (classical)

872. Mark-up

Problem code: MARKUP

Mark-up languages are computer languages that assist in the formatting of text files. Special keywords are used to mark the text to allow control of fonts, page styles, paragraph styles, etc. TeX, troff, and HTML are examples of mark-up languages.

Spell checking can be difficult to adapt to these special texts. In general, special processors or spell checkers must be created in order to accommodate mark-up languages. A special processor would recognize the mark-up language and strip it from the text so that the ‘‘plain’’ text could then be processed by a spell checker. For this problem, you are to write such a processor for a small mark-up language so that the output of your program will be the plain text without the mark-ups.

The mark-up language to consider is one which allows the modification of fonts within the text. Each markup command will be preceded by a \ character. If the letter following the \ character is not a recognized command from the table below then the character following the \ is printed as part of the plain text. For instance, the mark-up \\ can be used to print a single \.

- `\b` toggle bold font on/off (default state is off)
- `\i` toggle italics font on/off (default state is off)
- `\s` set font size; the s is immediately followed by an optional number; if the number is missing then the command will restore the previous size
- `\*` toggle processing of mark-ups on/off; if processing is toggle off then mark-ups are considered to be literal text (default state is on)

The number following the `\s` command can have a decimal point so 12, 9.5, 11., and .5 should all be recognized as valid numbers.

The input file will be plain text containing mark-ups from the language above. At the start, processing of mark-ups should be on. The file should be processed until the end-of-file is encountered.

A sample input file is shown here:

```
\s18.\bMARKUP sample\b\s

\*For bold statements use the \b command.\*

If you wish to \iemphasize\i something use the \\i command.

For titles use \s14BIG\s font sizes, 14 points usually works well.

Remember that all of the commands toggle except for the \\s command.
```

The following output file should be produced from the above sample input:

```
MARKUP sample

For bold statements use the \b command.

If you wish to emphasize something use the \i command.

For titles use BIG font sizes, 14 points usually works well.

Remember that all of the commands toggle except for the \s command.
```

Added by: Wanderley Guimaraes
Date: 2006-06-01
Time limit: 1s
Source limit: 50000B
Languages: All
Resource: 1995 ACM Mid-Central Programming Contest

SPOJ Problem Set (classical)

898. Transmitters

Problem code: TRANSMIT

In a wireless network with multiple transmitters sending on the same frequencies, it is often a requirement that signals don't overlap, or at least that they don't conflict. One way of accomplishing this is to restrict a transmitter's coverage area. This problem uses a shielded transmitter that only broadcasts in a semicircle.

A transmitter *T* is located somewhere on a 1,000 square meter grid. It broadcasts in a semicircular area of radius *r*. The transmitter may be rotated any amount, but not moved. Given *N* points anywhere on the grid, compute the maximum number of points that can be simultaneously reached by the transmitter's signal. Figure 1 shows the same data points with two different transmitter rotations.

[IMAGE]

All input coordinates are integers (0-1000). The radius is a positive real number greater than 0. Points on the boundary of a semicircle are considered within that semicircle. There are 1-150 unique points to examine per transmitter. No points are at the same location as the transmitter.

Input consists of information for one or more independent transmitter problems. Each problem begins with one line containing the (x,y) coordinates of the transmitter followed by the broadcast radius, *r*. The next line contains the number of points *N* on the grid, followed by *N* sets of (x,y) coordinates, one set per line. The end of the input is signalled by a line with a negative radius; the (x,y) values will be present but indeterminate. Figures 1 and 2 represent the data in the first two example data sets below, though they are on different scales. Figures 1a and 2 show transmitter rotations that result in maximal coverage.

For each transmitter, the output contains a single line with the maximum number of points that can be contained in some semicircle.

Input:

```
25 25 3.5
7
25 28
23 27
27 27
24 23
26 23
24 29
26 29
350 200 2.0
5
350 202
350 199
350 198
348 200
352 200
995 995 10.0
4
1000 1000
```

```
999 998
990 992
1000 999
100 100 -2.5
```

Output:

```
3
4
4
```

Added by: Wanderley Guimaraes
Date: 2006-06-09
Time limit: 1s
Source limit: 50000B
Languages: All
Resource: ACM Mid Central Regionals 2001

SPOJ Problem Set (classical)

899. Ws Cipher

Problem code: WSCIPHER

Weird Wally's Wireless Widgets, Inc. manufactures an eclectic assortment of small, wireless, network capable devices, ranging from dog collars, to pencils, to fishing bobbers. All these devices have very small memories. Encryption algorithms like Rijndael, the candidate for the Advanced Encryption Standard (AES) are demonstrably secure but they don't fit in such a tiny memory. In order to provide some security for transmissions to and from the devices, WWWW uses the following algorithm, which you are to implement.

Encrypting a message requires three integer keys, k_1, k_2, and k_3. The letters [a-i] form one group, [j-r] a second group, and everything else ([s-z] and underscore) the third group. Within each group the letters are rotated *left* by k_i positions in the message. Each group is rotated independently of the other two. Decrypting the message means doing a *right* rotation by k_i positions within each group.

Consider the message `the_quick_brown_fox` encrypted with k_i values of 2, 3 and 1. The encrypted string is `_icuo_bfnwhoq_kxert`. The figure below shows the decrypting right rotations for one character in each of the three character groups.

[IMAGE]

Looking at all the letters in the group [a-i] we see {`i`,`c`,`b`,`f`,`h`,`e`} appear at positions {2,3,7,8,11,17} within the encrypted message. After a right rotation of k_1=2, these positions contain the letters {`h`,`e`,`i`,`c`,`b`,`f`}. The table below shows the intermediate strings that come from doing all the rotations in the first group, then all rotations in the second group, then all the rotations in the third group. Rotating letters in one group will not change any letters in any of the other groups.

	[a-i], k_1 = 2	**[j-r], k_2 = 3**	**[s-z] and _, k_3 = 1**
Encrypted:	`_icuo_bfnwhoq_kxert`	`_heuo_icnwboq_kxfrt`	`_heuq_ickwbro_nxfot`
Decrypted:	`_heuo_icnwboq_kxfrt`	`_heuq_ickwbro_nxfot`	`the_quick_brown_fox`
Changes:	`^^  ^^ ^  ^`	`^  ^ ^^ ^ ^`	`^ ^ ^  ^  ^ ^ ^`

All input strings contain only lowercase letters and underscores(_). Each string will be at most 80 characters long. The k_i are all positive integers in the range 1-100.

Input consists of information for one or more encrypted messages. Each problem begins with one line containing k_1, k_2, and k_3 followed by a line containing the encrypted message. The end of the input is signalled by a line with all key values of 0.

For each encrypted message, the output is a single line containing the decrypted string.

Input:

```
2 3 1
_icuo_bfnwhoq_kxert
1 1 1
bcalmkyzx
3 7 4
wcb_mxfep_dorul_eov_qtkrhe_ozany_dgtoh_u_eji
2 4 3
cjvdksaltbmu
0 0 0
```

Output:

```
the_quick_brown_fox
abcklmxyz
the_quick_brown_fox_jumped_over_the_lazy_dog
ajsbktcludmv
```

Added by: Wanderley Guimaraes
Date: 2006-06-09
Time limit: 1s
Source limit: 50000B
Languages: All
Resource: ACM Mid Central Regionals 2001

SPOJ Problem Set (classical)

900. Split Windows

Problem code: SPLIT

The Dotty Software Company makes software that is displayed on inexpensive text based terminals. One application for this system has a main window that can be subdivided into further subwindows. Your task is to take a description of the screen layout after a sequence of window splits and draw the minimum sized window grid that is consistent with the description.

In this problem we will concentrate on the boundaries of windows, so all the characters inside of windows will be left blank. Each window that is not further subdivided has a label. Each label is a distinct uppercase letter. For a text terminal the boundaries of windows must be drawn with characters, chosen as follows: A capital letter label is placed in the upper left-hand corner of each undivided window. Asterisks,'*', appear in corners of windows where there is not a label. Dashes, '−', appear on upper and lower boundaries where there are not corners. Vertical bars, '|', appear on side boundaries where there are not corners.

For example, the sequence of splits below would generate Window 1: Initially there could be an application window labeled `M`, that is split next into left and right subwindows, adding label `R`, and the left subwindow is split into top and bottom subwindows, adding the label `C`.

[IMAGE]

For each pattern of splits there is a binary tree of characters that can describe it. The window splitting and tree structures are described together, building up from the simplest cases.

1. A window may be an undivided rectangle. Such a window has a capital letter as label. The tree for the window contains just the label.
2. A window may either be split into left and right subwindows or into top and bottom subwindows, and the corresponding trees have as root the boundary character for the split: a vertical line '|' or a horizontal dash '−' respectively. The root has left and right subtrees corresponding to the top and bottom or left and right subwindows respectively.

Tree 1, above, and Trees 2-4, below, would be consistent with Windows 1-4. Note that Tree 4 contains Trees 2 and 3.

[IMAGE]

[IMAGE]

The trees may be more succinctly expressed via a preorder traversal:

1. The preorder traversal of a tree with just one node (containing a letter) is that letter.
2. The preorder traversal of a tree with a left and a right subtree is the character from the root of the tree ('−' or '|') followed by the preorder traversal of the left subtree, and then the preorder traversal of the right subtree.

The preorder traversals for Trees 1 through 4 are

```
|-MCR     -|-ABC-D|E-FG    -P-|Q|RST     |-|-ABC-D|E-FG-P-|Q|RST
```

Each undivided window must have space for at least one character inside. Hence each tree of splits will be associated with a minimum window size. Windows 1-4 are minimum sized windows for Trees 1-4. Each window illustrates the fact that even in a minimum sized window, not all undivided windows contain only one character.

Consider Tree 4 and Window 4. The main window is split into a left window with Tree 2 and right window with Tree 3. The left window is like Window 2, but the right window is not just like Window 3. The heights of left and right subwindows must match, so the right window must be stretched.

The stretching rule depends on a definition of the size of windows. For dimension calculations it is easiest to imagine that a window contains its interior and a half character wide boundary on all sides, so the total dimensions of a window are one more than the dimensions of the interior. Hence the minimum dimensions of a window are 2 by 2, since a window must contain one character inside, and we add one for the boundary. This definition also means that the sum of the widths of left and right subwindows is the width of their enclosing window. The sum of the heights of top and bottom subwindows is the height of their enclosing window.

The right window in Window 4 must be stretched to match the height 10 of the left window. The right window is split into a top with tree `P` having minimum height 2 and a bottom with tree `-|Q|RST` having minimum height 4. The rule for the dimensions in the stretched window is that the heights of the subwindows expand in proportion to their minimum heights, if possible. Some symbols may help here: Let $D = 10$ be the height of the combined stretched window. We want to determine D_1 and D_2, the stretched heights of the top and bottom subwindow. Call the corresponding minimum dimensions $d = 6$, $d_1 = 2$, and $d_2 = 4$. If the window were expanded from a total height d to D in proportion, we would have $D_1 = d_1*(D/d) = 2*(10/6) = 3.333...$and $D_2 = d_2*(D/d) = 6.666....$ Since the results are not integers we increase D_1 to 4 and decrease D_2 to 6.

There is a similar calculation for the bottom window with tree `-|Q|RST`. It is further subdivided into a top with tree `|Q|RS` and a bottom with tree `T`, each having minimum height $2 = d_1 = d_2$. The heights need to add up to $D = 6$, so they are increased proportionally to $D_1 = D_2 = 2*(6/4) = 3$ (exact integers).

The final dimensions of an enclosing window are always determined before the final dimensions of its subwindows. In this example only heights needed to be apportioned. If all horizontal and vertical splits were interchanged in this example, producing a tree `-|-|ABC|D-E|FG|P|-Q-RST`, then widths would be apportioned correspondingly, as shown in the third part of the sample output below. If the proportion calculations do not work out to integers, it is always the top or left subwindow whose dimension is increased to the next integer.

The first line of input contains one integer, which is the total number of preorder traversals describing window structures. This line is followed by one line for each preorder traversal. Each preorder traversal will contain appropriate dividers '|' and '–' and from 1 to 26 uppercase letters.

For each preorder traversal, print the number of the preorder traversal on one line followed by the minimum sized window grid that the traversal could represent. Contrary to the general contest output conventions, there may be more than one consecutive blank in this output, but the other general rules for contest output are followed. The total number of rows or columns in output grids will be no more

than 53.

Input:

```
3
|-MCR
|-|-ABC-D|E-FG-P-|Q|RST
-|-|ABC|D-E|FG|P|-Q-RST
```

Output:

```
1
M-R-*
| | |
C-* |
| | |
*-*-*
2
A-C-P-----*
| | |     |
B-* |     |
| | |     |
D-*-Q-R-S-*
|   | | | |
E-F-* | | |
| | T-*-*-*
| G-*     |
| | |     |
*-*-*-----*
3
A-B-D-E---*
| | | |   |
C-*-* F-G-*
|   | | | |
P---Q-*T*-*
|   |  |  |
|   R--*  |
|   |  |  |
|   S--*  |
|   |  |  |
*---*--*--*
```

Added by: Wanderley Guimaraes
Date: 2006-06-09
Time limit: 1s
Source limit: 50000B
Languages: All
Resource: ACM Mid Central Regionals 2001

SPOJ Problem Set (classical)

901. Index Generation

Problem code: INDEXGEN

Most nonfiction and reference books have an *index* to help readers find references to specific terms or concepts in the text. Here is a sample index.

```
larch, 4, 237, 238, 414
+ Monty Python and, 64, 65, 66
+ planting of, 17
Lenny Kravitz, 50
+ going his way, 53
lumbago, 107
mango
+ Chris Kattan, 380
+ storage of, 87, 90
+ use in Nethack, 500, 501
+ Vitamin C content, 192
```

Each index entry contains a ***primary entry*** followed by zero or more ***secondary entries***, which begin with a '+'. Entries will normally be followed by a list of page references, but a primary entry might not be if at least one secondary entry is present (as is the case with *mango*, above). Primary entries are sorted, and secondary entries following a primary entry are also sorted. Sorting is case-insensitive. Page references for an entry are in ascending order and do not include duplicates. (A duplicate could occur if there are two or more identical entries on the same page.)

Your task is to read a document that has index information embedded within it and produce the index. Documents consist of one or more lines of ASCII text. The page number starts at 1, and the character '`&`' indicates the start of a new page (which adds 1 to the current page number). Index entries are indicated by a ***marker***, which in its most elaborate form has the following syntax:

{*text*`%`*primary*`$`*secondary*}

Here *text* is the text to be indexed, *primary* is an alternative primary entry, and *secondary* is a secondary entry. Both '`%`*primary*' and '`$`*secondary*' are optional, but if both are present they must appear in the order given. If *primary* is present then it is used as the primary entry, and if not then *text* is used as the primary entry. If *secondary* is present then the marker adds a page reference for that secondary entry; otherwise it adds a page reference for the primary entry. A single marker cannot add a page reference for both a primary and secondary entry. Here are examples of each of the four possible types of marker, which correspond to four of the entries in the sample index above.

```
... his {lumbago} was acting up, so ...
... {Lenny%Lenny Kravitz} lit up the crowd with his version of ...
... Monty Python often used the {larch$Monty Python and} in ...
... when storing {mangos%mango$storage of}, be sure to ...
```

The input consists of one or more documents, followed by a line containing only '`**`' that signals the end of the input. Documents are implictly numbered starting with 1. Each document consists of one or more lines of text followed by a line containing only '`*`'. Each line of text will be at most 79 characters long, not counting end-of-line characters. For document *i*, output the line '`DOCUMENT` *i*'

followed by the sorted index using the exact output format shown in the examples.

Be sure to read *Notes to Teams*, which has general formatting guidelines that pertain to all problem input files, including this one. Also note:

- A document will contain at most 100 markers, with at most 20 primary entries.
- A primary entry will have at most 5 secondary entries.
- An entry will have at most 10 unique page references (not including duplicates).
- The character '`&`' will not appear anywhere within a marker, and will appear at most 500 times within a document.
- The character '`*`' is used only to signal the end of a document or the end of the input.
- The characters '`{`', '`}`', '`%`', and '`$`' will only be used to define markers, and will not appear in any text or entries.
- A marker may span one or more lines. Every end-of-line within a marker must be converted to a single space.
- A space within a marker (including a converted end-of-line) is normally included in the text/entry, just like any other character. However, any space that immediately follows '`{`', immediately precedes '`}`', or is immediately adjacent to '`%`' or '`$`' must be ignored.
- The total length of a marker, measured from the opening '`{`' to the closing '`}`', and in which all embedded end-of-lines are converted to spaces, will be at most 79 characters.

Input:

```
Call me Ishmael.
*
One {fish $unary}, two {fish$ binary},&red {fish $ scarlet}, blue {fish$
azure}. & By { Dr. Seuss }.
*
This is a {simple } & & { document} that &{
simply %simple
$adverb
} & {illustrates %vision} &&&&& one {simple-minded% simple} {Judge}'s {vision}
for what a {document } might { look % vision} like.
*
**
```

Output:

```
DOCUMENT 1
DOCUMENT 2
Dr. Seuss, 3
fish
+ azure, 2
+ binary, 1
+ scarlet, 2
+ unary, 1
DOCUMENT 3
document, 3, 10
Judge, 10
simple, 1, 10
+ adverb, 4
vision, 5, 10
```

Added by: Wanderley Guimaraes
Date: 2006-06-09
Time limit: 1s
Source limit:50000B
Languages: All
Resource: ACM Mid Central Regionals 2001

SPOJ Problem Set (classical)

902. Hangover

Problem code: HANGOVER

How far can you make a stack of cards overhang a table? If you have one card, you can create a maximum overhang of half a card length. (We're assuming that the cards must be perpendicular to the table.) With two cards you can make the top card overhang the bottom one by half a card length, and the bottom one overhang the table by a third of a card length, for a total maximum overhang of 1/2 + 1/3 = 5/6 card lengths. In general you can make n cards overhang by 1/2 + 1/3 + 1/4 + ... + 1/(n + 1) card lengths, where the top card overhangs the second by 1/2, the second overhangs tha third by 1/3, the third overhangs the fourth by 1/4, etc., and the bottom card overhangs the table by 1/(n + 1). This is illustrated in the figure below.

[IMAGE]

Input

The input consists of one or more test cases, followed by a line containing the number 0.00 that signals the end of the input. Each test case is a single line containing a positive floating-point number c whose value is at least 0.01 and at most 5.20; c will contain exactly three digits.

Output

For each test case, output the minimum number of cards necessary to achieve an overhang of at least c card lengths. Use the exact output format shown in the examples.

Input:
```
1.00
3.71
0.04
5.19
0.00
```

Output:
```
3 card(s)
61 card(s)
1 card(s)
273 card(s)
```

Added by: Wanderley Guimaraes
Date: 2006-06-09
Time limit: 1s
Source limit:50000B
Languages: All
Resource: ACM Mid Central Regionals 2001

SPOJ Problem Set (classical)

903. Double Vision

Problem code: DOUBLEVI

The DoubleVision company designs inks and fonts that can be easily read by both humans and machines. They design their fonts on a rectangular grid. Shown below is a very simple 5x3 design for the first five digits.

```
.o. .o. oo. oo. o.o
o.o .o. ..o ..o o.o
o.o .o. .o. oo. ooo
o.o .o. o.. ..o ..o
.o. .o. ooo oo. ..o
```

The ink appears to be normal black ink, but just underneath the surface DoubleVision adds a special polymer that can be detected by an infrared scanner. A human sees the black ink but not the polymer, and a machine sees the polymer but not the black ink. The only problem is that the polymer is much more expensive than the ink, so DoubleVision wants to use as little of it as possible. They have discovered that with many fonts, each symbol can be uniquely identified by at most two pixels. By only adding the polymer to one or two pixels per symbol, they drastically lower costs while still ensuring 100% accuracy in their scanners. The font shown above has this property; pixels that uniquely identify each letter are highlighted with '`#`'. (There are other choices that would work as well.)

```
.#. .o. #o. oo. o.#
#.o .#. ..o ..o o.o
o.o .o. .o. #o. ooo
o.o .o. #.. ..o ..o
.o. .o. ooo #o. ..o
```

Your job is to write a program to determine if a given font has this property, and if so highlight the pixels.

The input consists of one or more test cases, followed by a line containing '`0 0 0`' (three zeros) that signals the end of the input. Each test case begins with a line containing three positive integers *n*, *r*, and *c*, separated by a space: *n* is the number of symbols in the font, *r* is the number of rows in each grid, and *c* is the number of columns in each grid. The next *r* lines contain the image of each symbol, using the exact format shown in the examples: a dot '`.`' represents an empty part of the grid, a lowercase '`o`' represents a pixel, and adjacent grids are separated by a space. The total width of each line will be at most 79 characters (not counting end-of-line characters), and *r* will be at most 10. The test cases are implicitly numbered starting with 1.

For test case *i*, first output a line that says '`Test` *i*'. Then determine if each symbol can be uniquely identified with one or two pixels. If not, output a line with the word '`impossible`'. Otherwise, output the font in the same format except that the identifying pixels for each symbol are replaced with '`#`'.

In general there may be several different pixels or pixel pairs that uniquely identify a symbol. To ensure that the output is unique, we add the following definition and rules. When comparing two pixels, the *topmost-leftmost* pixel is the one closest to the top of the grid. If both pixels are on the same row, then the topmost-leftmost is the one closest to the left of the grid.

If one pixel will work, highlight the topmost-leftmost pixel that works. Never highlight a two-pixel solution if a one-pixel solution is possible. If two pixels are needed, highlight the pair with the topmost-leftmost pixel. If two or more pairs have the same topmost-leftmost pixel, highlight the one with the topmost-leftmost *other* pixel.

Input:

```
3 2 2
oo oo .o
o. .o o.
3 2 2
oo oo .o
o. .o oo
5 5 3
.o. .o. oo. oo. o.o
o.o .o. ..o ..o o.o
o.o .o. .o. oo. ooo
o.o .o. o.. ..o ..o
.o. .o. ooo oo. ..o
1 2 4
.o..
...o
0 0 0
```

Output:

```
Test 1
impossible
Test 2
#o #o .o
#. .# ##
Test 3
.#. .o. #o. oo. o.#
#.o .#. ..o ..o o.o
o.o .o. .o. #o. ooo
o.o .o. #.. ..o ..o
.o. .o. ooo #o. ..o
Test 4
.#..
...o
```

Added by: Wanderley Guimaraes
Date: 2006-06-09
Time limit: 1s
Source limit: 50000B
Languages: All
Resource: ACM Mid Central Regionals 2001

SPOJ Problem Set (classical)

904. Image Perimeters

Problem code: IMAGE

Technicians in a pathology lab analyze digitized images of slides. Objects on a slide are selected for analysis by a mouse click on the object. The perimeter of the boundary of an object is one useful measure. Your task is to determine this perimeter for selected objects.

The digitized slides will be represented by a rectangular grid of periods, '`.`', indicating empty space, and the capital letter '`X`', indicating part of an object. Simple examples are

```
XX Grid 1  .XXX Grid 2
XX         .XXX
.XXX
...X
..X.
X...
```

An `X` in a grid square indicates that the entire grid square, including its boundaries, lies in some object. The `X` in the center of the grid below is *adjacent* to the `X` in any of the 8 positions around it. The grid squares for any two adjacent `X`'s overlap on an edge or corner, so they are connected.

```
XXX
X X  Central X and adjacent X's
XXX
```

An object consists of the grid squares of all `X`'s that can be linked to one another through a sequence of adjacent `X`'s. In Grid 1, the whole grid is filled by one object. In Grid 2 there are two objects. One object contains only the lower left grid square. The remaining `X`'s belong to the other object.

The technician will always click on an `X`, selecting the object containing that `X`. The coordinates of the click are recorded. Rows and columns are numbered starting from 1 in the upper left hand corner. The technician could select the object in Grid 1 by clicking on row 2 and column 2. The larger object in Grid 2 could be selected by clicking on row 2, column 3. The click could not be on row 4, column 3.

[IMAGE] One useful statistic is the perimeter of the object. Assume each `X` corresponds to a square one unit on each side. Hence the object in Grid 1 has perimeter 8 (2 on each of four sides). The perimeter for the larger object in Grid 2 is illustrated in the figure at the left. The length is 18.

Objects will not contain any totally enclosed holes, so the leftmost grid patterns shown below could *NOT* appear. The variations on the right could appear:

```
Impossible Possible

XXXX XXXX XXXX XXXX
X..X XXXX X... X...
XX.X XXXX XX.X XX.X
XXXX XXXX XXXX XX.X
```

```
.....  .....  .....  .....
..X..  ..X..  ..X..  ..X..
.X.X.  .XXX.  .X...  .....
..X..  ..X..  ..X..  ..X..
.....  .....  .....  .....
```

The input will contain one or more grids. Each grid is preceded by a line containing the number of rows and columns in the grid and the row and column of the mouse click. All numbers are in the range 1-20. The rows of the grid follow, starting on the next line, consisting of '.' and 'X' characters.

The end of the input is indicated by a line containing four zeros. The numbers on any one line are separated by blanks. The grid rows contain no blanks.

For each grid in the input, the output contains a single line with the perimeter of the specified object.

Input:

```
2 2 2 2
XX
XX
6 4 2 3
.XXX
.XXX
.XXX
...X
..X.
X...
5 6 1 3
.XXXX.
X....X
..XX.X
.X...X
..XXX.
7 7 2 6
XXXXXXX
XX...XX
X..X..X
X..X...
X..X..X
X.....X
XXXXXXX
7 7 4 4
XXXXXXX
XX...XX
X..X..X
X..X...
X..X..X
X.....X
XXXXXXX
0 0 0 0
```

Output:

```
8
18
40
48
8
```

Added by: Wanderley Guimaraes
Date: 2006-06-09
Time limit: 1s
Source limit:50000B
Languages: All
Resource: ACM Mid Central Regionals 2001

SPOJ Problem Set (classical)

912. Submatrix of submatrix

Problem code: MATRIX2

You are given a matrix P of N rows and M columns. It consists of integer numbers in the range [1..100]. We define the sum of a matrix is the sum of its elements. Your task is to find a submatrix Q (of A rows and B columns) of P and a submatrix K (of C rows and D columns) of Q so that the difference between the sum of Q and the sum of K is maximal, and submatrix K is absolutely inside matrix Q (i.e no element on matrix Q's sides is also in matrix K).
Because the tests are large, we suggest a method to define matrix P:
P[i][j] = (P[i][j-1] * 71 + 17) mod 100 + 1 . (1 <= i <= N , 1 <= j <= M)
With this method we only care about P[i][1].
Constraints
1 <= N , M <= 1000
1 <= A <= N
1 <= B <= M
0 <= C <= A - 2
0 <= D <= B - 2

Input

The first line of the input contains an integer t (1 <= t <= 10), equal to the number of testcases. Then descriptions of t testcases follow. The first line of the description contains 6 integer numbers N, M, A, B, C, D. Then N lines follow, line i contains one integer number P[i][1].

Output

For each test case, your program should output the maximal difference between two matrices (in a separate line).

Example

Input:
```
1
3 3 3 3 1 1
1
2
3
```
Output:
```
260
```

Added by:	Nguyen Dinh Tu
Date:	2006-08-18
Time limit:	5s
Source limit:	50000B
Languages:	All
Resource:	Base on a problem in IOI 2006 .

SPOJ Problem Set (classical)

913. Query on a tree II

Problem code: QTREE2

You are given a tree (an undirected acyclic connected graph) with **N** nodes, and edges numbered 1, 2, 3...**N**-1. Each edge has an integer value assigned to it, representing its length.

We will ask you to perfrom some instructions of the following form:

- **DIST a b** : ask for the distance between node **a** and node **b**
  or
- **KTH a b k** : ask for the **k**-th node on the path from node **a** to node **b**

Example:
N = 6
1 2 1 // edge connects node 1 and node 2 has cost 1
2 4 1
2 5 2
1 3 1
3 6 2
Path from node 4 to node 6 is 4 -> 2 -> 1 -> 3 -> 6
DIST 4 6 : answer is 5 (1 + 1 + 1 + 2 = 5)
KTH 4 6 4 : answer is 3 (the 4-th node on the path from node 4 to node 6 is 3)

Input

The first line of input contains an integer **t**, the number of test cases (**t** <= 25). **t** test cases follow.

For each test case:

- In the first line there is an integer **N** (**N** <= 10000)
- In the next **N**-1 lines, the i-th line describes the i-th edge: a line with three integers **a b c** denotes an edge between **a**, **b** of cost **c** (**c** <= 100000)
- The next lines contain instructions **"DIST a b"** or **"KTH a b k"**
- The end of each test case is signified by the string "**DONE**".

There is one blank line between successive tests.

Output

For each **"DIST"** or **"KTH"** operation, write one integer representing its result.

Print one blank line after each test.

Example

Input:

```
1

6
1 2 1
2 4 1
2 5 2
1 3 1
3 6 2
DIST 4 6
KTH 4 6 4
DONE
```

Output:

```
5
3
```

Added by: Thanh-Vy Hua
Date: 2006-08-27
Time limit: 2s
Source limit: 15000B
Languages: All
Resource: Special thanks to Ivan Krasilnikov for his alternative solution

SPOJ Problem Set (classical)

944. Free Tour

Problem code: FTOUR

In order to celebrate the 2nd anniversary of Travel Agent SPOJ (Safe - Professional - hOspitable - Joyful), the management intend to hold free tours around cities for clients to make them more satisfied with SPOJ.

A tour is a simple cycle, starting at any city (called a source-city) visits some other cities (each city must be visited at most once) and then returns to the source-city. The number of roads in the tour should be an even number because we are celebrating a 2nd anniversary, and 2 is even! Since many tours in different areas of the country are planned, the cost of organising them could turn out quite high. Hence, the management of SPOJ hope to find at least one ’reasonable’ tour, which should have as small a number of roads as possible.

You’re given maps of the areas where SPOJ wants to hold free tours. For each map, help them figure out a reasonable tour.

Input

The first line of input contains an integer **t**, the number of maps (**t** <= 5). **t** maps follow.

For each map:

- In the first line there are 2 integers **N** - number of cities in that area, **M** - number of roads (1 <= **N** <= 8000, 0 <= **M** <= 10000)
- In the next **M** lines, the i-th line describes the i-th road: a line with two integers **a b** denotes a bidirectional road between city **a** and city **b**

There is one blank line between successive tests.

Output

For each map, if there is no tour satisfying the conditions, write "-1" (without quotes). Otherwise, write one integer representing the number of roads in a reasonable tour, and in the next line show out the tour with form "source-city a b c .. source-city", that means the tour is source-city -> city a -> city b -> ... -> source-city. If there are many tours satisfy in each map, any of them will be accepted.

Example

Input:

```
2

3 3
1 2
2 3
3 1
```

```
4 4
1 2
2 3
3 4
4 1
```

Output:

```
-1
4
1 2 3 4 1
```

Added by: Thanh-Vy Hua
Date: 2006-09-16
Time limit: 1s-4s
Source limit: 15000B
Languages: All

SPOJ Problem Set (classical)

962. Intergalactic Map

Problem code: IM

Map Jedi knights, Qui-Gon Jinn and his young apprentice Obi-Wan Kenobi, are entrusted by Queen Padmé Amidala to save **Naboo** from an invasion by the Trade Federation. They must leave Naboo immediately and go to **Tatooine** to pick up the proof of the Federation's evil design. They then must proceed on to the Republic's capital planet **Coruscant** to produce it in front of the Republic's Senate. To help them in this endeavor, the queen's captain provides them with an intergalactic map. This map shows connections between planets not yet blockaded by the Trade Federation. Any pair of planets has at most one connection between them, and all the connections are two-way. To avoid detection by enemy spies, the knights must embark on this adventure without visiting any planet more than once. Can you help them by determining if such a path exists?
Note - In the attached map, the desired path is shown in bold.

Input Description

The first line of the input is a positive integer t <= 20, which is the number of test cases. The descriptions of the test cases follow one after the other. The first line of each test case is a pair of positive integers n, m (separated by a single space). 2 <= n <= 30011 is the number of planets and m <= 50011 is the number of connections between planets. The planets are indexed with integers from 1 to n. The indices of Naboo, Tatooine and Coruscant are 1, 2, 3 respectively. The next m lines contain two integers each, giving pairs of planets that have a connection between them.

Output Description

The output should contain t lines. The i^{th} line corresponds to the i^{th} test case. The output for each test case should be **YES** if the required path exists and **NO** otherwise.

Example

Input
2
3 3
1 2
2 3
1 3
3 1
1 3
Output
YES
NO

Added by: Kashyap KBR
Date: 2006-10-10
Time limit: 6s
Source limit:50000B
Languages: All
Resource: Fair Isaac Programming Challenge - prelims 2006

SPOJ Problem Set (classical)

964. Entrapment

Problem code: EN

Example The separatist leader General Grievous, the second in command of Count Dooku, comes to know that Chancellor Palpatine's convoy, escorted by Obi-wan and Anakin, is scheduled to depart from **Kashyyyk** in the Middle Rim of the Universe to **Alderaan**. General Grievous is aware that there are multiple paths going via different sets of planets from Kashyyyk to Alderaan. To make his abduction attempt successful, he decides to send his robots to the planet closest to Kashyyyk, other than itself, which lies on all the possible paths from Kashyyyk to Alderaan. Since you have pledged your allegiance to Count Dooku, you need to help him identify this planet. The planetary map which is given to you for this purpose consists of a set of one-way connections between planets. You also know that a pair of planets can have at most one connection between them in each direction and there is always a path from Kashyyyk to Alderaan.
Note: In the given example, the planet with index 5 is the required planet.

Input Description

The first line of the input is a positive integer t <= 20, which is the number of test cases. The descriptions of the test cases follow one after the other. The first line of each test case is a pair of positive integers n, m (separated by a single space). 2 <= n <= 30011 is the number of planets and m <= 100011 is the number of connections between planets. The planets are indexed with integers from 1 to n and the indices of Kashyyyk and Alderaan are 1, n respectively. Each of the next m lines contains two integers p,q, meaning that there is a one-way connection from planet p to planet q.

Output Description

The output should contain t lines. The i^{th} line corresponds to the i^{th} test case. The output for the i^{th} test case should be the index of the planet with the required property.

Example

Input
2
3 2
1 3
3 2
4 4
1 3
3 4
1 2
2 4
Output
3
4

Added by: Kashyap KBR
Date: 2006-10-10
Time limit: 5s
Source limit:50000B
Languages: All
Resource: Fair Isaac Programming Challenge - prelims 2006

SPOJ Problem Set (classical)

967. Parking Bay

Problem code: PB

Illustration Luke Skywalker successfully leads the rebel starship fleet to break the Emperor's siege of the planet Endor. The rebels, jubilant in their victory, return to their base on the moon of Endor to pay their homage to Princess Leia. They fly towards the parking bay where there are n parking slots in a row. One by one n starships numbered S_1 to S_n enter the parking bay. Each rebel R_i heads to his favorite parking slot P_i, and if it is free, he docks his starship there. Otherwise, he continues further to the next free slot and occupies it. But if all succeeding slots are occupied, he leaves for good. How many sequences P_i are such that every rebel can dock his starship?

Input Description

The first line of the input is a positive integer t <= 20, which is the number of test cases. The descriptions of the test cases follow one after the other. The description of each test case contains exactly one line (a positive integer), containing the value of n <= 1000000.

Output Description

The output consists of exactly t lines. The ith line should be A_i%10007, where A_i is the number of sequences in the i^{th} case, and 'x%y' represents the remainder when x is divided by y.

Example

Input
2
1
2
Output
1
3
Explanation: In the given example, 11, 12 and 21 are the possible sequences.

Added by: Kashyap KBR
Date: 2006-10-10
Time limit: 2s
Source limit:50000B
Languages: All
Resource: Fair Isaac Programming Challenge - prelims 2006

SPOJ Problem Set (classical)

972. Birthday

Problem code: BIRTHDAY

It is Byteman's birthday today. There are n children at his birthday party (including Byteman). The children are numbered from 1 to n. Byteman's parents have prepared a big round table and they have placed n chairs around the table. When the children arrive, they take seats. The child number 1 takes one of the seats. Then the child number 2 takes the seat on the left. Then the child number 3 takes the next seat on the left, and so on. Finally the child number n takes the last free seat, between the children number 1 and n-1. Byteman's parents know the children very well and they know that some of the children will be noisy, if they sit too close to each other. Therefore the parents are going to reseat the children in a specific order. Such an order can be described by a permutation p1, p2, . . . , pn (p1, p2, . . . , pn are distinct integers from 1 to n) -- child pi (for i = 2,3, . . . ,n) should sit on child pi-1's left, and child p1 should sit on child pn's left. To seat all the children in the given order, the parents must move each child around the table to the left or to the right some number of seats. For each child, they must decide how the child will move -- that is, they must choose a direction of movement (left or right) and distance (number of seats). On the given signal, all the children stand up at once, move to the proper places and sit down. The reseating procedure throws the birthday party into a mess. The mess is equal to the total distance any child moves. The children can be reseated in many ways. The parents choose one with minimum mess. Help them to find such a way to reseat the children.

Input

The first line of standard input contains one integer n (1 <= n <= 50000). The second line contains n integers p1, p2, . . . , pn, separated by single spaces. Numbers p1, p2, . . . , pn form a permutation of the set {1,2, . . . ,n} describing the desired order of the children.

Output

The first and the only line of standard output should contain one integer: the minimum possible mess.

Example

Input:
```
5
1 5 4 3 2
```

Output:
```
6
```

Added by: Le Đôn Khue
Date: 2006-10-11
Time limit: 3s
Source limit:50000B
Languages: All
Resource: 2nd round of Hanoi University of Science - Based on a problem from IOI 2005

SPOJ Problem Set (classical)

987. Mobile

Problem code: MOBILE

Mobile
Manfred loves to build mobiles out of old CDs. For each one, he has an exact plan how it should look like: The CDs are all hanging exactly on the same height. For each pair of CDs, he writes down the height of the lowest bar such that both CDs are hanging somewhere under this bar. For example, the following mobile and distance matrix fit together:

[IMAGE]

After a while, Manfred realizes that he does not succeed to build every mobile he planned to. For example, there is no solution for the following distance matrix:
0 1 2
1 0 3
2 3 0
So, he decides to write a computer program that checks the distance matrices and tells him if there is a solution.

Input

Several matrices to check. The first row contains the size of the matrix (n), the next n rows contain the distances in the matrix. Then, the data of the next matrix comes, and so on. The input is terminated by a zero as matrix size.

Output

For each matrix, write true if Manfred can build a mobile, false otherwise.

Example

Input:

```
5
0 1 1 1 3
1 0 1 1 3
1 1 0 1 3
1 1 1 0 3
3 3 3 3 0
3
0 1 2
1 0 3
2 3 0
3
1 1 1
1 0 2
1 1 0
0
```

Output:

```
true
false
false
```

Added by: Martin Bader
Date: 2006-10-17
Time limit: 15s
Source limit: 50000B
Languages: All
Resource: Ulm Bioinformatics Course WS 06/07

SPOJ Problem Set (classical)

999. Generalized Matrioshkas

Problem code: MATRIOSH

Vladimir worked for years making matrioshkas, those nesting dolls that certainly represent truly Russian craft. A matrioshka is a doll that may be opened in two halves, so that one finds another doll inside. Then this doll may be opened to find another one inside it. This can be repeated several times, till a final doll -that cannot be opened- is reached.

Recently, Vladimir realized that the idea of nesting dolls might be generalized to nesting toys. Indeed, he has designed toys that contain toys but in a more general sense. One of these toys may be opened in two halves and it may have more than one toy inside it. That is the new feature that Vladimir wants to introduce in his new line of toys.

Vladimir has developed a notation to describe how nesting toys should be constructed. A toy is represented with a positive integer, according to its size. More precisely: if when opening the toy represented by m we find the toys represented by n1, n2, . . ., nr, it must be true that n1 +n2 +. . .+nr < m. And if this is the case, we say that toy m contains directly the toys n1, n2, . . ., nr . It should be clear that toys that may be contained in any of the toys n1, n2, . . ., nr are not considered as directly contained in the toy m.

A generalized matrioshka is denoted with a non-empty sequence of non zero integers of the form: a1 a2 . . . aN such that toy k is represented in the sequence with two integers -k and k, with the negative one occurring in the sequence first that the positive one.

For example, the sequence

-9 -7 -2 2 -3 -2 -1 1 2 3 7 9

represents a generalized matrioshka conformed by six toys, namely, 1, 2 (twice), 3, 7 and 9.
Note that toy 7 contains directly toys 2 and 3. Note that the first copy of toy 2 occurs left from the second one and that the second copy contains directly a toy 1. It would be wrong to understand that the first -2 and the last 2 should be paired.

On the other hand, the following sequences do not describe generalized matrioshkas:

-9 -7 -2 2 -3 -1 -2 2 1 3 7 9

because toy 2 is bigger than toy 1 and cannot be allocated inside it.

-9 -7 -2 2 -3 -2 -1 1 2 3 7 -2 2 9

because 7 and 2 may not be allocated together inside 9.

Input

The input file contains several test cases, each one of them in a separate line. Each test case is a sequence of non zero integers, each one with an absolute value less than 10^7.

Output

Output texts for each input case are presented in the same order that input is read. For each test case the answer must be a line of the form

:-) Matrioshka!

if the design describes a generalized matrioshka. In other case, the answer should be of the form

:-(Try again.

Example

Input:

```
-9 -7 -2 2 -3 -2 -1 1 2 3 7 9
-9 -7 -2 2 -3 -1 -2 2 1 3 7 9
-9 -7 -2 2 -3 -1 -2 3 2 1 7 9
-100 -50 -6 6 50 100
-100 -50 -6 6 45 100
-10 -5 -2 2 5 -4 -3 3 4 10
-9 -5 -2 2 5 -4 -3 3 4 9
```

Output:

```
:-) Matrioshka!
:-( Try again.
:-( Try again.
:-) Matrioshka!
:-( Try again.
:-) Matrioshka!
:-( Try again.
```

Added by: Camilo Andrés Varela León
Date: 2006-10-24
Time limit: 5s
Source limit:50000B
Languages: All
Resource: XX Colombian National Programming ACM 2006

SPOJ Problem Set (classical)

1000. Equidivisions

Problem code: EQDIV

An equidivision of an n × n square array of cells is a partition of the n^2 cells in the array in exactly n sets, each one with n contiguous cells. Two cells are contiguous when they have a common side.

A good equidivision is composed of contiguous regions. The figures show a good and a wrong equidivision for a 5×5 square:

[IMAGE]

Note that in the second example the cells labeled with 4 describe three non-contiguous regions and cells labeled with 5 describe two non-contiguous regions. You must write a program that evaluates if an equidivision of the cells in a square array is good or not.

Input

It is understood that a cell in an n×n square array is denoted by a pair (i, j), with 1 <= i, j <= n. The input file contains several test cases. Each test case begins with a line indicating n, 0 < n < 100, the side of the square array to be partitioned. Next, there are n - 1 lines, each one corresponding to one partition of the cells of the square, with some non-negative integer numbers.

Consecutive integers in a line are separated with a single blank character. A line of the form

a1 a2 a3 a4 ...

means that cells denoted with the pairs (a1, a2), (a3, a4), ... belong to one of the areas in the partition. The last area in the partition is defined by those cells not mentioned in the n - 1 given lines. If a case begins with n = 0 it means that there are no more cases to analyze.

Output

For each test case good must be printed if the equidivision is good, in other case, wrong must be printed. The answers for the different cases must preserve the order of the input.

Example

Input:

```
2
1 2 2 1
5
1 1 1 2 1 3 3 2 2 2
2 1 4 2 4 1 5 1 3 1
4 5 5 2 5 3 5 5 5 4
2 5 3 4 3 5 4 3 4 4
5
1 1 1 2 1 3 3 2 2 2
2 1 3 1 4 1 5 1 4 2
```

```
4 5 5 2 5 3 5 5 5 4
2 4 1 4 3 5 4 3 4 4
0
```

Output:

```
wrong
good
wrong
```

Added by: Camilo Andrés Varela León
Date: 2006-10-25
Time limit: 3s
Source limit: 50000B
Languages: All
Resource: XX Colombian National Programming ACM 2006

SPOJ Problem Set (classical)

1001. Babylonian Roulette

Problem code: BROUL

People of Babylon were devoted to chance games and one of the most popular was a special kind of roulette. Recently, some old Babylonian tablets were found. They described details of the roulette game.

In modern terms, the rules of the game were as follows:

- Roulette's compartments had only six labels: -1, -2, -3, 1, 2, 3.
- The game was played by turns, during a day. Turns were numerated 0, 1, 2, ...
- Players could win or lose a multiple of the bet, a quantity of money that was constant along the day.
- At turn t there was an amount of money Pt, called the pot.
- At the start, there was an initial amount of money P0 in the pot.
- P0 and the bet were positive numbers arbitrarily defined by the King.
- In a turn, a player turned the roulette. A player could not play more than once in a day. Depending on the compartment where the ball came to rest, the player won (or lose, if the value was negative) an amount wt = L * bet of money, where L corresponded to the compartment's label.
- The won money was taken from the pot (or put in it if the player lose), i.e. the value of the pot in a given turn was determined by Pt+1 = Pt + wt.
- If as a result of the last rule Pt+1 was a negative number the winner won only the maximum multiple of the bet that he could win without making a negative pot.
- If at some turn the pot was less than the bet, the game was ended for that day. If that was not the case the game continued till sunset.

Beside the tablets that explained the rules some other tablets were found. These had lines with three numbers. Archeologists conjecture that each of these lines were part of a kind of accountability system for the game, where numbers represented, for a given day, the value of the pot at the beginning, the bet and the value of the pot at the end.

For example, a line with the numbers

10000 1500 11500

could mean that there was only one turn where the player won with label 1. Another possibility is that there were three turns with results 2, 1 and -2.

On the other hand, there were found other tablets with triplets of numbers that seem like the above described that, however, cannot represent results of a game day. There is no hypothesis of what they are.

Archeologists want to validate their hypothesis analyzing batches of tablets with triplets. They want to estimate the number of people that played in a day. To begin, they want to establish, for each triplet of numbers in a tablet that could represent a result of a game day, the minimal number of players that

played that day. In the above example the answer to this question is 1. Tablets that cannot represent results should be identified. You are hired to help with this task.

Input

The input file contains several test cases, each one of them in a separate line. Each test case is a triplet of non negative integers, indicating the initial pot, the bet and the final pot for a day.

Each of the input numbers is less than 10^8. The initial pot and the bet are greater than 0.

A line with a triplet of 0's denotes the end of the input.

Output

Output texts for each input case are presented in the same order that input is read. For each test case the answer must be a printed line.

If the test case cannot represent the result of a game day, the output line has the words No accounting tablet. In other case, the printed answer is one positive integer number telling the minimal number of players that could turn the roulette for the day corresponding to the annotations.

Example

Input:

```
10000 1000 22000
24 13 2
5100 700 200
54 16 158
360 6 72
25 10 5
0 0 0
```

Output:

```
4
No accounting tablet
3
No accounting tablet
16
1
```

Added by: Camilo Andrés Varela León
Date: 2006-10-25
Time limit: 2s
Source limit: 50000B
Languages: All
Resource: XX Colombian National Programming ACM 2006

SPOJ Problem Set (classical)

1002. Uncle Jack

Problem code: UJ

Dear Uncle Jack is willing to give away some of his collectable CDs to his nephews. Among the titles you can find very rare albums of Hard Rock, Classical Music, Reggae and much more; each title is considered to be unique. Last week he was listening to one of his favorite songs, Nobody's fool, and realized that it would be prudent to be aware of the many ways he can give away the CDs among some of his nephews.

So far he has not made up his mind about the total amount of CDs and the number of nephews. Indeed, a given nephew may receive no CDs at all.

Please help dear Uncle Jack, given the total number of CDs and the number of nephews, to calculate the number of different ways to distribute the CDs among the nephews.

Input

The input consists of several test cases. Each test case is given in a single line of the input by, space separated, integers N (1 <= N <= 1000) and D (0 <= D <= 2500), corresponding to the number of nephews and the number of CDs respectively. The end of the test cases is indicated with N = D = 0.

Output

The output consists of several lines, one per test case, following the order given by the input. Each line has the number of all possible ways to distribute D CDs among N nephews.

Example

Input:

```
1 20
3 10
0 0
```

Output:

```
1
59049
```

Added by: Camilo Andrés Varela León
Date: 2006-10-25
Time limit: 1s
Source limit:512B
Languages: All
Resource: XX Colombian National Programming ACM 2006

SPOJ Problem Set (classical)

1003. Little Quilt

Problem code: QUILT

Little Quilt is a small language introduced by Ravi Sethi in his book 'Programming Languages'.

Here, a restricted version of Little Quilt is presented. The language is defined by the following BNF grammar:

< QUILT > ::= A | B | turn(< QUILT >) | sew(< QUILT >,< QUILT >)

A and B represent the two primitive quilts. Each primitive quilt corresponds to a matricial arrangement of 2 × 2 characters. turn() and sew() are operations over quilts.

The instruction turn(x) turns the quilt x 90 degrees clockwise. The following table illustrates the primitive quilts as well as examples of the effect of the turn() operation:

[IMAGE]

Accordingly, the instruction sew(x,y) sews quilt x to the left of quilt y. Both x and y must have the same height, otherwise an error will be generated. The following figure represents the result of sew(A,turn(B)):

[IMAGE]

while the sew(turn(sew(B,turn(B))),A) generates an error message.

Your job is to build an interpreter of the Little Quilt language.

Input

The input file will be a text file containing different Little Quilt expressions, each one ended by a semicolon character (;). Space and new line characters must be ignored; this means that an expression may span several lines.

Output

The output file contains the quilts produced as a result of interpreting the input expressions.

Each quilt must be preceded by a line, left aligned, with the format

Quilt i:
where i is the quilt number, starting at 1. If the expression interpretation generates and error, the word

error
must be printed.

Example

Input:

```
sew(turn(sew(B,turn(B))),
turn(sew(turn(B),B))) ;
sew(turn(sew(B,turn(B))),A);
sew(turn(sew(A,turn(A))),
turn(turn(
turn(sew(A,turn(A))))))
;
```

Output:

```
Quilt 1:
||--
||--
--||
--||
Quilt 2:
error
Quilt 3:
\\//
+\/+
+/\+
//\\
```

Added by: Camilo Andrés Varela León
Date: 2006-10-25
Time limit: 1s-2s
Source limit:50000B
Languages: All
Resource: XX Colombian National Programming ACM 2006

SPOJ Problem Set (classical)

1004. Polygon Encoder

Problem code: POLYCODE

Imagine an infinite table with rows and columns numbered using the natural numbers. The following figure shows a procedure to traverse such a table assigning a consecutive natural number to each table cell:

[IMAGE]

This enumeration of cells can be used to represent complex data types using natural numbers:

- A pair of natural numbers (i, j) is represented by the number corresponding to the cell in row i and column j. For instance, the pair (3,2) is represented by the natural number 17; this fact is noted by P2(3, 2) = 17.
- The pair representation can be used to represent n-tuples. A triplet (a, b, c) is represented by P2(a, P2(b, c)). A 4-tuple (a, b, c, d) is represented by P2(a, P2(b, P2(c, d))). This procedure can be generalized for an arbitrary n:

  Pn(a1, ..., an) = P2(a1, Pn-1(a2, .., an)),

  where Pn denotes the n-tuple representation function, n >= 2. For example P3(2, 0, 1) = 12.
- A list of arbitrary length ha1, ..., ani is represented by

  L(ha1, ..., ani) = P2(n, Pn(a1, ..., an)).

For example, L(h0, 1i) = 12.

The Association of Convex Makers (ACM) uses this clever enumeration scheme in a polygon representation system. The system can represent a polygon, defined by integer coordinates, using a natural number as follows: given a polygon defined by a vertex sequence h(x1, y1), ..., (xn, yn)i assign the natural number:

L((hP2(x1, y1), ..., P2(xn, yn))).

ACM needs a program that, given a natural numbers that represents a polygon, calculates the area of the polygon. It is guaranteed that the given polygon is a simple one, i.e. its sides do not intersect.

As an example of the problem, the triangle with vertices at (1,1), (2,0) and (0,0) is codified with the number 2141. The area of this triangle is 1.

Input

The input consists of several test cases. Each test case is given in a single line of the input by a natural number representing a polygon. The end of the test cases is indicated with *.

Output

One line per test case, preserving the input order. Each output line contains a decimal number telling the area of the corresponding encoded polygon. Areas must be printed with 1 decimal place, truncating less significative decimal places.

Example

Input:

```
2141
206
157895330
*
```

Output:

```
1.0
0.5
1.0
```

Added by: Camilo Andrés Varela León
Date: 2006-10-25
Time limit: 5s
Source limit: 50000B
Languages: All
Resource: XX Colombian National Programming ACM 2006

SPOJ Problem Set (classical)

1021. Aibohphobia

Problem code: AIBOHP

BuggyD suffers from AIBOHPHOBIA - the fear of Palindromes. A palindrome is a string that reads the same forward and backward.

To cure him of this fatal disease, doctors from all over the world discussed his fear and decided to expose him to large number of palindromes. To do this, they decided to play a game with BuggyD. The rules of the game are as follows:

BuggyD has to supply a string **S**. The doctors have to add or insert characters to the string to make it a palindrome. Characters can be inserted anywhere in the string.

The doctors took this game very lightly and just appended the reverse of **S** to the end of **S**, thus making it a palindrome. For example, if **S** = "`fft`", the doctors change the string to "`ffttff`".

Nowadays, BuggyD is cured of the disease (having been exposed to a large number of palindromes), but he still wants to continue the game by his rules. He now asks the doctors to insert the minimum number of characters needed to make **S** a palindrome. Help the doctors accomplish this task.

For instance, if **S** = "`fft`", the doctors should change the string to "`tfft`", adding only 1 character.

Input

The first line of the input contains an integer **t**, the number of test cases. **t** test cases follow.

Each test case consists of one line, the string **S**. The length of **S** will be no more than 6100 characters, and **S** will contain no whitespace characters.

Output

For each test case output one line containing a single integer denoting the minimum number of characters that must be inserted into **S** to make it a palindrome.

Example

Input:
```
1
fft
```

Output:
```
1
```

Added by:	Matthew Reeder
Date:	2006-10-29
Time limit:	8s
Source limit:	30000B
Languages:	All
Resource:	Al-Khawarizm 2006

SPOJ Problem Set (classical)

1022. Angels and Devils

Problem code: ANGELS

It's the year 21546 AD, and due to increased population (you wouldn't believe me if I gave you the actual numbers), land has become very expensive. Because of the lack of space, Heaven and Hell were built in the same area. The area can be represented as a grid of **X** × **Y** unit squares. Some of the squares were captured by the Devil (and thus belong to Hell) and the rest is the Almighty's property. On each square, a room has been built with transparent glass walls. However, some of the heavenly rooms are already occupied by Angels. For security purposes, rooms occupied by Angels have concrete opaque walls.

Recently many fighters were killed in a tournament. Fighting is no longer considered cruel, so all the fighters will deserve spots in heaven. However, because of the space shortage, all of them may not be able to recieve a spot in heaven. The fighters still hold a grudge against each other so a fighter cannot be placed in a room from which he can see any other fighter. A fighter can only see in the four cardinal directions (North, South, East and West). He cannot look diagonally or in any other direction.

Find the maximum number of fighters who can have a heavenly room.

Input

The first line of the input contains an integer **t**, the number of test cases. **t** test cases follow.

The first line of each test case consists of two integers **X** <= 300 and **Y** <= 300, separated by a single space. Next, **X** lines follow, each having **Y** letters separated by spaces. The jth letter on the i-th line is one of the following (quotes are for clarity, and do not appear in the input):

1. "H", if the room at location (i, j) is heavenly and vacant.
2. "A", if the room at location (i, j) is heavenly and is already occupied by an angel. Note that these rooms are not transparent.
3. "D", if the room at location (i, j) belongs to the Devil.

Output

A single line for each test case containing an integer denoting the maximum number of fighters that can fit in heaven.

Example

Input:

```
1
4 7
H H H H H H H
H H H H H H H
H H H H H H H
```

```
H H H H H H H
```

Output:

```
4
```

Added by:	Matthew Reeder
Date:	2006-10-29
Time limit:	7s
Source limit:	30000B
Languages:	All
Resource:	Al-Khawarizm 2006

SPOJ Problem Set (classical)

1024. Complete Chess Boards

Problem code: COMCB

BuggyD has always been fascinated with chess boards (though he really sucks at chess). He makes an observation that a chess board is complete with respect to knights and rooks and incomplete with respect to bishops (unless the dimensions are 1×1). A complete chess board is one in which it is possible to traverse all the squares starting from one possible square. Knights have always been his favourite pieces and he has decided to analyze completeness with respect to knights. Given the dimesions of the chess board help BuggyD find the lexicographically first path that visits all squares of a chess board with a knight.

Each square must be traversed only once. Note that a knight can only move two squares in one direction and one square perpendicular to the previous direction.

Input

The first line of the input contains an integer **t**, the number of test cases. **t** test cases follow.

Each test case consists of a single line cotaining two integers (**X** and **Y**) separated by a single space, specifying the dimensions of the chess board. The numbers 1 to **X** denote rows and the capital letters A to Y denote the coloumns. Each square is represented by its column index followed by it's row index - for example, `B4` denotes the square in the 4th row and 2nd column.

The total number of squares on the chess board will be no more than 26.

Output

For each test case output one line consisting of the lexicographically first path of the knight, or "-1" (quotes for clarity) if the chess board is incomplete with respect to a knight.

Example

Input:
```
1
4 5
```

Output:
```
A1B3C1A2B4D3E1C2D4E2C3A4B2D1E3C4A3B1D2E4
```

Added by: Matthew Reeder
Date: 2006-10-29
Time limit: 3s
Source limit: 30000B
Languages: All
Resource: Al-Khawarizm 2006

SPOJ Problem Set (classical)

1025. Fashion Shows

Problem code: FASHION

A fashion show rates participants according to their level of hotness. Two different fashion shows were organized, one for men and the other for women. A date for the third is yet to be decided ;) .

Now the results of both fashion shows are out. The participants of both the fashion shows have decided to date each other, but as usual they have difficuly in choosing their partners. The Maximum Match dating serive (MMDS) comes to their rescue and matches them in such a way that that maximizes the hotness bonds for all couples.

If a man has been rated at hotness level `x` and a women at hotness level `y`, the value of their hotness bond is `x*y`.

Both fashion shows contain **N** participants each. MMDS has done its job and your job is to find the sum of hotness bonds for all the couples that MMDS has proposed.

Input

The first line of the input contains an integer **t**, the number of test cases. **t** test cases follow.

Each test case consists of 3 lines:

- The first line contains a single integer **N** (1 <= **N** <= 1000).
- The second line contains **N** integers separated by single spaces denoting the hotness levels of the men.
- The third line contains **N** integers separated by single spaces denoting the hotness levels of the women.

All hotness ratings are on a scale of 0 to 10.

Output

For each test case output a single line containing a single integer denoting the sum of the hotness bonds for all pairs that MMDS has proposed.

Example

Input:

```
2
2
1 1
3 2
3
2 3 2
1 3 2
```

Output:

```
5
15
```

Added by:	Matthew Reeder
Date:	2006-10-29
Time limit:	2s
Source limit:	30000B
Languages:	All
Resource:	Al-Khawarizm 2006

SPOJ Problem Set (classical)

1026. Favorite Dice

Problem code: FAVDICE

BuggyD loves to carry his favorite die around. Perhaps you wonder why it's his favorite? Well, his die is magical and can be transformed into an N-sided unbiased die with the push of a button. Now BuggyD wants to learn more about his die, so he raises a question:

What is the expected number of throws of his die while it has **N** sides so that each number is rolled at least once?

Input

The first line of the input contains an integer **t**, the number of test cases. **t** test cases follow.

Each test case consists of a single line containing a single integer **N** (1 <= **N** <= 1000) - the number of sides on BuggyD's die.

Output

For each test case, print one line containing the expected number of times BuggyD needs to throw his **N**-sided die so that each number appears at least once. The expected number must be accurate to 2 decimal digits.

Example

Input:
```
2
1
12
```

Output:
```
1.00
37.24
```

Added by: Matthew Reeder
Date: 2006-10-29
Time limit: 2s
Source limit: 30000B
Languages: All
Resource: Al-Khawarizm 2006

SPOJ Problem Set (classical)

1027. Fool the Police

Problem code: FPOLICE

Dhamaka Singh (a crook) has just robbed a bank and would like to get out of the country as soon as possible. But there is a slight problem, the police! On his way out of the country he has to pass through some police stations. Each police station has a certain risk (for Dhamaka Singh) associated with it. He wants to get to the airport within a certain time **T** or else he'll miss his flight. He also wants to take a path that minimizes the total risk associated with it. Help Dhamaka Singh get out of the country.

Input

The first line of the input contains an integer **t**, the number of test cases. **t** test cases follow.

The first line of each test case contains 2 integers **N** (3 <= **N** 100) and **T** (1 <= **T** <= 250), denoting the number of police stations and the total time he has to reach the airport, respectively.

Dhamaka Singh has to start from the first police station and reach the $\mathbf{N}^{th}$ one (the airport is just after the $\mathbf{N}^{th}$ police station). You can consider the time taken between the $\mathbf{N}^{th}$ police station and the airport to be negligible.

Next there are **N** lines with **N** numbers in each line, separated by single spaces. All numbers are separated by a single space. The j^{th} integer in the i^{th} line represents the time taken to reach the j^{th} police station from the i^{th} police station.

Next there are another **N** lines with **N** numbers in each line. All numbers are separated by a single space. The j^{th} integer in the i^{th} line represents the risk involved in travelling to the j^{th} police station from the i^{th} police station.

Output

For each test case output one line containing 2 integers separated by a single space.

The first integer denotes the minimum total risk to reach the airport. The second integer denotes the minimum time required to reach the airport at the minimum total risk.

If it is impossible to reach the airport within time **T** (inclusive), just print "-1" (quotes for clarity).

Example

Input:

```
1
4 10
0 6 2 3
6 0 2 3
3 1 0 2
3 3 2 0
0 2 2 7
2 0 1 2
```

```
2 2 0 5
7 2 5 0
```

Output:

```
4 9
```

Added by: Matthew Reeder
Date: 2006-10-29
Time limit: 3s
Source limit: 30000B
Languages: All
Resource: Al-Khawarizm 2006

SPOJ Problem Set (classical)

1028. Hubulullu

Problem code: HUBULLU

After duelling in quake (a multiplayer game), Airborne and Pagfloyd decide do test themselves out in another game called Hubulullu. The rules of the game are as follows:

N wooden pieces (marked with numbers 1 to N) are placed in a transparent bottle. On his turn the first player takes out some piece (numbered x) and all the pieces numbered by divisors of x that are present in the transparent bottle. The second player picks another number and removes it and its divisors as well. Play continues in an alternating fashion until all pieces have been removed from the bottle. The player who removes the last piece from the bottle wins the game.

Both players play optimally. Given **N** (the number of wooden pieces in the transparent bottle initially) and the name of the player who starts the game, determine the winner.

Input

The first line of the input contains an integer **t**, the number of test cases. **t** test cases follow.

Each test case consists of a single line containing two integers separated by a single space. The first integer is **N** (1 <= **N** <= 2000000000), indicating the number of pieces, and the second integer indicates the player who starts - "0" means Airborne starts the game and "1" means Pagfloyd starts the game (quotes for clarity).

Output

For each test case output one line containing either "Airborne wins." or "Pagfloyd wins."

For each N, it’s possible to determine a winner if both players play optimally.

Example

Input:
```
1
1 0
```

Output:
```
Airborne wins.
```

Added by: Matthew Reeder
Date: 2006-10-29
Time limit: 7s
Source limit: 30000B
Languages: All
Resource: Al-Khawarizm 2006

SPOJ Problem Set (classical)

1029. Matrix Summation

Problem code: MATSUM

A N × N matrix is filled with numbers. BuggyD is analyzing the matrix, and he wants the sum of certain submatrices every now and then, so he wants a system where he can get his results from a query. Also, the matrix is dynamic, and the value of any cell can be changed with a command in such a system.

Assume that initially, all the cells of the matrix are filled with 0. Design such a system for BuggyD. Read the input format for further details.

Input

The first line of the input contains an integer **t**, the number of test cases. **t** test cases follow.

The first line of each test case contains a single integer **N** (1 <= **N** <= 1024), denoting the size of the matrix.

A list of commands follows, which will be in one of the following three formats (quotes are for clarity):

1. `"SET x y num"` - Set the value at cell (x, y) to num (0 <= x, y < **N**).
2. `"SUM x1 y1 x2 y2"` - Find and print the sum of the values in the rectangle from (x1, y1) to (x2, y2), inclusive. You may assume that x1 <= x2 and y1 <= y2, and that the result will fit in a signed 32-bit integer.
3. `"END"` - Indicates the end of the test case.

Output

For each test case, output one line for the answer to each "SUM" command. Print a blank line after each test case.

Example

Input:

```
1
4
SET 0 0 1
SUM 0 0 3 3
SET 2 2 12
SUM 2 2 2 2
SUM 2 2 3 3
SUM 0 0 2 2
END
```

Output:

```
1
12
12
13
```

Warning: large Input/Output data, be careful with certain languages

---

Added by: Matthew Reeder
Date: 2006-10-29
Time limit: 7s
Source limit: 30000B
Languages: All
Resource: Al-Khawarizm 2006

SPOJ Problem Set (classical)

1030. Triple Fat Ladies

Problem code: EIGHTS

Pattern Matchers have been designed for various sorts of patterns. Mr. HKP likes to observe patterns in numbers. After completing his extensive research on the squares of numbers, he has moved on to cubes. Now he wants to know all numbers whose cube ends in 888.

Given a number **k**, help Mr. HKP find the k^{th} number (indexed from 1) whose cube ends in 888.

Input

The first line of the input contains an integer **t**, the number of test cases. **t** test cases follow.

Each test case consists of a single line containing a single integer **k** (1 <= **k** <= 2000000000000).

Output

For each test case, output a single integer which denotes the k^{th} number whose cube ends in 888. The result will be less than 2^{63}.

Example

Input:
```
1
1
```

Output:
```
192
```

Added by: Matthew Reeder
Date: 2006-10-30
Time limit: 5s
Source limit: 30000B
Languages: All
Resource: Al-Khawarizm 2006

SPOJ Problem Set (classical)

1031. Up Subsequence

Problem code: UPSUB

If $x = a_0 a_1 a_2 ... a_{n-1}$ is a string where a_i denotes the character at index i, a subsequence $a_{j0} a_{j1} a_{j2} ... a_{jn}$ is called an upsubsequence if $a_{j0} <= a_{j1} <= a_{j2} <= ... <= a_{jn}$ and $j0 < j1 < j2 < ... < jn$.

A maximal upsubsequence of a string is defined as the upsubsequence of maximum length. BuggyD observes that a string **x** can have many maximal upsubsequences. Help him find all the maximal upsubsequences in **x**.

Input

The first line of the input contains an integer **t**, the number of test cases. **t** test cases follow.

Each test case consists of a single line containing a string **x**, where the length of **x** is no more than 100. **x** will not contain any spaces, tabs or other whitespace characters.

Output

For each test csae, output all of the maximal upsubsequences of **x** in lexicographical order. Print a blank line after each test case.

Example

Input:
```
1
abcbcbcd
```

Output:
```
abbbcd
abbccd
abcccd
```

Added by: Matthew Reeder
Date: 2006-10-30
Time limit: 3s
Source limit: 30000B
Languages: All
Resource: Al-Khawarizm 2006

SPOJ Problem Set (classical)

1043. Can you answer these queries I

Problem code: GSS1

You are given a sequence A[1], A[2], ..., A[N] . (|A[i]| <= 15007 , 1 <= N <= 50000). A query is defined as follows:
Query(x,y) = Max { a[i]+a[i+1]+...+a[j] ; x <= i <= j <= y }.
Given M queries, your program must output the results of these queries.

Input

- The first line of the input file contains the integer N.
  - In the second line, N numbers follow.
  - The third line contains the integer M.
- M lines follow, where line i contains 2 numbers xi and yi.

Output

Your program should output the results of the M queries, one query per line.

Example

Input:
```
3
-1 2 3
1
1 2
```
Output:
```
2
```

Added by: Nguyen Dinh Tu
Date: 2006-11-01
Time limit: 1s-2s
Source limit:5000B
Languages: All

SPOJ Problem Set (classical)

1108. Card Trick

Problem code: CTRICK

The magician shuffles a small pack of cards, holds it face down and performs the following procedure:

1. The top card is moved to the bottom of the pack. The new top card is dealt face up onto the table. It is the Ace of Spades.

2. Two cards are moved one at a time from the top to the bottom. The next card is dealt face up onto the table. It is the Two of Spades.

3. Three cards are moved one at a time...

4. This goes on until the nth and last card turns out to be the n of Spades.

This impressive trick works if the magician knows how to arrange the cards beforehand (and knows how to give a false shuffle). Your program has to determine the initial order of the cards for a given number of cards, 1 <= n <= 20000.

Input

On the first line of the input is a single positive integer, telling the number of test cases to follow. Each case consists of one line containing the integer n.

Output

For each test case, output a line with the correct permutation of the values 1 to n, space separated. The first number showing the top card of the pack, etc...

Example

Input:
```
2
4
5
```

Output:
```
2 1 4 3
3 1 4 5 2
```

Added by: Camilo Andrés Varela León
Date: 2006-11-23
Time limit: 11s
Source limit:50000B
Languages: All
Resource: Nordic Collegiate Contest 2006

SPOJ Problem Set (classical)

1110. Sudoku

Problem code: SUDOKU

A Sudoku grid is a 16x16 grid of cells grouped in sixteen 4x4 squares, where some cells are filled with letters from A to P (the first 16 capital letters of the English alphabet), as shown in the figure. The game is to fill all the empty grid cells with letters from A to P such that each letter from the grid occurs once only in the line, the column, and the 4x4 square it occupies. The initial content of the grid satisfies the constraints mentioned above and guarantees a unique solution.

		A					C						O		I
	J			A		B		P		C	G	F		H	
		D			F		I		E					P	
	G		E	L		H					M		J		
				E					C			G			
	I			K		G	A		B				E		J
D		G	P			J		F					A		
	E				C		B			D	P			O	
E			F		M			D			L		K		A
	C									O		I		L	
H		P		C			F		A			B			
			G		O	D				J					H
K				J					H		A		P		L
		B			P			E			K			A	
	H			B			K			F	I		C		
		F				C			D			H		N	

a) Sudoku grid

F	P	A	H	M	J	E	C	N	L	B	D	K	O	G	I
O	J	M	I	A	N	B	D	P	K	C	G	F	L	H	E
L	N	D	K	G	F	O	I	J	E	A	H	M	B	P	C
B	G	C	E	L	K	H	P	O	F	I	M	A	J	D	N
M	F	H	B	E	L	P	O	A	C	K	J	G	N	I	D
C	I	L	N	K	D	G	A	H	B	M	O	P	E	F	J
D	O	G	P	I	H	J	M	F	N	L	E	C	A	K	B
J	E	K	A	F	C	N	B	G	I	D	P	L	H	O	M
E	B	O	F	P	M	I	J	D	G	H	L	N	K	C	A
N	C	J	D	H	B	A	E	K	M	O	F	I	G	L	P
H	M	P	L	C	G	K	F	I	A	E	N	B	D	J	O
A	K	I	G	N	O	D	L	B	P	J	C	E	F	M	H
K	D	E	M	J	I	F	N	C	H	G	A	O	P	B	L
G	L	B	C	D	P	M	H	E	O	N	K	J	I	A	F
P	H	N	O	B	A	L	K	M	J	F	I	D	C	E	G
I	A	F	J	O	E	C	G	L	D	P	B	H	M	N	K

b) Solution

Input

The first line of the input contains an integer K - determining the number of datasets (K<=10). Each data set encodes a grid and contains 16 strings on 16 consecutive lines as shown in the example input below. The ith string stands for the ith line of the grid, is 16 characters long, and starts from the first position of the line. String characters are from the set {A,B,...,P,-}, where - (minus) designates empty grid cells. The data sets are separated by single empty lines.

Output

For each data set in the input print the completed 16x16 Sudoku as specified by the rules above. The program prints the solution of the input encoded grids in the same format and order as used for input. The output for each data set should be separated by single empty lines.

Example

Input:

```
1
--A----C-----O-I
-J--A-B-P-CGF-H-
--D--F-I-E----P-
```

```
-G-EL-H----M-J--
----E----C--G---
-I--K-GA-B---E-J
D-GP--J-F----A--
-E---C-B--DP--O-
E--F-M--D--L-K-A
-C---------O-I-L-
H-P-C--F-A--B---
---G-OD---J----H
K---J----H-A-P-L
--B--P--E--K--A-
-H--B--K--FI-C--
--F---C--D--H-N-
```

Output:

```
FPAHMJECNLBDKOGI
OJMIANBDPKCGFLHE
LNDKGFOIJEAHMBPC
BGCELKHPOFIMAJDN
MFHBELPOACKJGNID
CILNKDGAHBMOPEFJ
DOGPIHJMFNLECAKB
JEKAFCNBGIDPLHOM
EBOFPMIJDGHLNKCA
NCJDHBAEKMOFIGLP
HMPLCGKFIAENBDJO
AKIGNODLBPJCEFMH
KDEMJIFNCHGAOPBL
GLBCDPMHEONKJIAF
PHNOBALKMJFIDCEG
IAFJOECGLDPBHMNK
```

Added by: P.Kasthuri Rangan-
Date: 2006-11-24
Time limit: 7s
Source limit: 50000B
Languages: All
Resource: ACM Southeastern European Regional Programming Contest - 2006

SPOJ Problem Set (classical)

1112. Number Steps

Problem code: NSTEPS

Starting from point (0,0) on a plane, we have written all non-negative integers 0, 1, 2,... as shown in the figure. For example, 1, 2, and 3 has been written at points (1,1), (2,0), and (3, 1) respectively and this pattern has continued.

Illustration

You are to write a program that reads the coordinates of a point (x, y), and writes the number (if any) that has been written at that point. (x, y) coordinates in the input are in the range 0...10000.

Input

The first line of the input is N, the number of test cases for this problem. In each of the N following lines, there is x, and y representing the coordinates (x, y) of a point.

Output

For each point in the input, write the number written at that point or write No Number if there is none.

Example

Input:

```
3
4 2
6 6
3 4
```

Output:

```
6
12
No Number
```

Added by:	Camilo Andrés Varela León
Date:	2006-11-25
Time limit:	2s
Source limit:	50000B
Languages:	All
Resource:	Asia - Tehran 2000

SPOJ Problem Set (classical)

1161. Tic-Tac-Toe (I)

Problem code: TOE1

Tic Tac Toe is a child's game played on a 3 by 3 grid. One player, X, starts by placing an X at an unoccupied grid position. Then the other player, O, places an O at an unoccupied grid position. Play alternates between X and O until the grid is filled or one player's symbols occupy an entire line (vertical, horizontal, or diagonal) in the grid.

We will denote the initial empty Tic Tac Toe grid with nine dots. Whenever X or O plays we fill in an X or an O in the appropriate position. The example below illustrates each grid configuration from the beginning to the end of a game in which X wins.

[IMAGE]

Your job is to read a grid and to determine whether or not it could possibly be part of a valid Tic Tac Toe game. That is, is there a series of plays that can yield this grid somewhere between the start and end of the game?

Input

The first line of input contains N, the number of test cases. 4N-1 lines follow, specifying N grid configurations separated by empty lines.

Output

For each case print "yes" or "no" on a line by itself, indicating whether or not the configuration could be part of a Tic Tac Toe game.

Example

Input:

```
2
X.O
OO.
XXX

O.X
XX.
OOO
```

Output:

```
yes
no
```

Added by:	Camilo Andrés Varela León
Date:	2006-12-14
Time limit:	1s
Source limit:	50000B
Languages:	All
Resource:	Waterloo local 2002.09.21

SPOJ Problem Set (classical)

1162. Tic-Tac-Toe (II)

Problem code: TOE2

In the game of tic-tac-toe, two players take turns marking squares of an initially empty 3 × 3 grid with either X's or O's. The first player always marks squares using X's, whereas the second player always marks squares using O's. If at any point during the game either player manages to mark three consecutive squares in a row, column, or diagonal with his/her symbol, the game terminates.

Given a board configuration, your goal is to determine whether the board configuration represents the possible final state of a valid tic-tac-toe game.

Input

The input test file will contain multiple cases. Each test case consists of a single line containing 9 characters, which represent the 9 squares of a tic-tac-toe grid, given one row at a time. Each character on the line will either be 'X', 'O' (the letter O), or '.' (indicating an unfilled square). The end-of-file is marked by a single line containing the word "end".

Output

For each input test case, write a single line containing either the word "valid" or "invalid" indicating whether the given board configuration is the final state of some possible tic-tac-toe game.

Example

Input:

```
XXXOO.XXX
XOXOXOXOX
OXOXOXOXO
XXOOOXXOX
XO.OX...X
.XXX.XOOO
OOXXXOOXO
end
```

Output:

```
invalid
valid
invalid
valid
valid
invalid
invalid
```

Added by:	Camilo Andrés Varela León
Date:	2006-12-14
Time limit:	1s
Source limit:	50000B
Languages:	All
Resource:	Stanford Local 2006

SPOJ Problem Set (classical)

1163. Java vs C ++

Problem code: JAVAC

Apologists of Java and C++ can argue for hours proving each other that their programming language is the best one. Java people will tell that their programs are clearer and less prone to errors, while C++ people will laugh at their inability to instantiate an array of generics or tell them that their programs are slow and have long source code.

Another issue that Java and C++ people could never agree on is identifier naming. In Java a multiword identifier is constructed in the following manner: the first word is written starting from the small letter, and the following ones are written starting from the capital letter, no separators are used. All other letters are small. Examples of a Java identifier are javaIdentifier, longAndMnemonicIdentifier, name, nEERC.

Unlike them, C++ people use only small letters in their identifiers. To separate words they use underscore character '_'. Examples of C++ identifiers are c_identifier, long_and_mnemonic_identifier, name (you see that when there is just one word Java and C++ people agree), n_e_e_r_c.

You are writing a translator that is intended to translate C++ programs to Java and vice versa. Of course, identifiers in the translated program must be formatted due to its language rules -- otherwise people will never like your translator.

The first thing you would like to write is an identifier translation routine. Given an identifier, it would detect whether it is Java identifier or C++ identifier and translate it to another dialect. If it is neither, then your routine should report an error. Translation must preserve the order of words and must only change the case of letters and/or add/remove underscores.

Input

The input file consists of several lines that contains an identifier. It consists of letters of the English alphabet and underscores. Its length does not exceed 100.

Output

If the input identifier is Java identifier, output its C++ version. If it is C++ identifier, output its Java version. If it is none, output 'Error!' instead.

Example

Input:

```
long_and_mnemonic_identifier
anotherExample
i
bad_Style
```

Output:

```
longAndMnemonicIdentifier
another_example
i
Error!
```

Added by: Camilo Andrés Varela León
Date: 2006-12-14
Time limit: 1s
Source limit: 50000B
Languages: All
Resource: Northeastern Europe 2006

SPOJ Problem Set (classical)

1166. Dead Fraction

Problem code: DEADFR

Mike is frantically scrambling to finish his thesis at the last minute. He needs to assemble all his research notes into vaguely coherent form in the next 3 days. Unfortunately, he notices that he had been extremely sloppy in his calculations. Whenever he needed to perform arithmetic, he just plugged it into a calculator and scribbled down as much of the answer as he felt was relevant. Whenever a repeating fraction was displayed, Mike simply reccorded the first few digits followed by "...". For instance, instead of "1/3" he might have written down "0.3333...". Unfortunately, his results require exact fractions! He doesn't have time to redo every calculation, so he needs you to write a program (and FAST!) to automatically deduce the original fractions.

To make this tenable, he assumes that the original fraction is always the simplest one that produces the given sequence of digits; by simplest, he means the the one with smallest denominator. Also, he assumes that he did not neglect to write down important digits; no digit from the repeating portion of the decimal expansion was left unrecorded (even if this repeating portion was all zeroes).

Input

There are several test cases. For each test case there is one line of input of the form "0.dddd..." where dddd is a string of 1 to 18 digits, not all zero. A line containing 0 follows the last case.

Output

For each case, output the original fraction.

Example

Input:

```
0.2...
0.20...
0.474612399...
0
```

Output:

```
2/9
1/5
1186531/2500000
```

Added by: Camilo Andrés Varela León
Date: 2006-12-16
Time limit: 4s
Source limit: 50000B
Languages: All
Resource: Waterloo Local Contest Sep. 27, 2003

SPOJ Problem Set (classical)

1167. Move To Invert

Problem code: MINCOUNT

A triangle made of coins of height h is as follows
It has h coins at the base and h-1 coins one level above base and so on.(Coins are placed as shown in the figure below)
And at the top most level there will be only one coin
Now given h the task is to invert this triangle by moving minimum number of coins. For example when h=4 triangle is
Invert
> For h=4 at least 3 coins must be moved to invert it.

Input<h3>

In the first line N will be given and then N lines follow with each line having a integer which is the height of triangle in that test case.00<=h<10^{10};

Output

For each test case output in a seperate line the minimum number of moves required to invert the triangle. Output fits in long long data type

Example

Inputt:
```
1
3
```

Output:
```
2
```

Added by:	**Abhilash I**
Date:	**2006-12-16**
Time limit:	**1s**
Source limit:	**50000B**
Languages:	**All**
Resource:	**IIIT Hyderabad Local Programming Contest**

SPOJ Problem Set (classical)

1182. Sorted bit squence

Problem code: SORTBIT

Let's consider the 32 bit representation of all integers i from m up to n inclusive (m <= i <= n; m × n >= 0, -2^31 <= m <= n <= 2^31-1). Note that a negative number is represented in 32 bit Additional Code. That is the 32 bit sequence, the binary sum of which and the 32 bit representation of the corresponding positive number is 2^32 (`1 0000 0000 0000 0000 0000 0000 0000 0000` in binary).

For example, the 32 bit representation of 6 is `0000 0000 0000 0000 0000 0000 0000 0110`

and the 32 bit representation of -6 is `1111 1111 1111 1111 1111 1111 1111 1010`

because

```
0000 0000 0000 0000 0000 0000 0000 0110 (6)
+
1111 1111 1111 1111 1111 1111 1111 1010 (-6)
-----------------------------------------------
= 1 0000 0000 0000 0000 0000 0000 0000 0000 (2^32)
```

Let's sort the 32 bit representations of these numbers in increasing order of the number of bit 1. If two 32 bit representations that have the same number of bit 1, they are sorted in lexicographical order.

For example, with m = 0 and n = 5, the result of the sorting will be

No.	Decimal number	Binary 32 bit representation
1	0	`0000 0000 0000 0000 0000 0000 0000 0000`
2	1	`0000 0000 0000 0000 0000 0000 0000 0001`
3	2	`0000 0000 0000 0000 0000 0000 0000 0010`
4	4	`0000 0000 0000 0000 0000 0000 0000 0100`
5	3	`0000 0000 0000 0000 0000 0000 0000 0011`
6	5	`0000 0000 0000 0000 0000 0000 0000 0101`

with m = -5 and n = -2, the result of the sorting will be

No.	Decimal number	Binary 32 bit representation
1	-4	`1111 1111 1111 1111 1111 1111 1111 1100`
2	-5	`1111 1111 1111 1111 1111 1111 1111 1011`
3	-3	`1111 1111 1111 1111 1111 1111 1111 1101`
4	-2	`1111 1111 1111 1111 1111 1111 1111 1110`

Given m, n and k (1 <= k <= min{n - m + 1, 2 147 473 547}), your task is to write a program to find a number corresponding to k-th representation in the sorted sequence.

Input

The input consists of several data sets. The first line of the input file contains the number of data sets which is a positive integer and is not bigger than 1000. The following lines describe the data sets.

For each data set, the only line contains 3 integers m, n and k separated by space.

Output

For each data set, write in one line the k-th number of the sorted numbers.

Example

Sample input:

```
2
0 5 3
-5 -2 2
```

Sample output:

```
2
-5
```

Added by: Le Đôn Khue
Date: 2006-12-28
Time limit: 20s
Source limit: 50000B
Languages: All
Resource: ACM ICPC 2006, Asia Regional Contest, site Hanoi

SPOJ Problem Set (classical)

1183. Accomodate the palace

Problem code: PALACE

There is a big palace in which rooms are constructed in the from of a square matrix.Now these rooms have to be filled with people.
As there are conflicts between people to maintain the equilibrium total number of people must be odd in every row and every column.
A room can accomodate only a single person.
Given the size of palace N one has to find total number of ways people can be accomodatede in that.

Input

First line consits of an integer K and then K test cases follow.

Output

For each test case you have to output the result%98777 in a seperate line.

Example

Input:
```
1
3
```
Output:
```
16
```

Added by:	Abhilash I
Date:	2006-12-30
Time limit:	3s
Source limit:	50000B
Languages:	All
Resource:	IIIT Hyderabad Local Programming Contest

SPOJ Problem Set (classical)

1267. Origin of Life

Problem code: ORIGLIFE

Conway's *Game of Life* is not really a game, but a *cellular automaton* -- a set of rules describing interactions among adjacent cells on a grid. In our game, we have an *n* by *m* rectangular grid of cells identified by integer coordinates (*x, y*). The game progresses through a series of steps; at each step a new *generation* is computed from the current *generation*. The game begins with the *first generation*. In any given generation, which we shall call the current generation, each cell is either *live* or *dead*. In the next generation, each cell's status may change, depending on the status of its immediate neighbours in the current generation. Two distinct cells (x_1, y_1) and (x_2, y_2) are immediate neighbours if they are horizontally, vertically, or diagonally adjacent; that is, if $|x_1 - x_2 <= 1|$ and $|y_1 - y_2 <= 1|$. Each cell that is not on the border of the grid has eight immediate neighbours. There are three integer parameters (*a, b, c*) which affect the game. The rules of the game are:

- If a live cell has fewer than *a* live neighbours in the current generation it dies of loneliness. That is, it is dead in the next generation.
- If a live cell has more than *b* live neighbours in the current generation it dies of overcrowding. That is, it is dead in the next generation.
- If a dead cell has more than *c* live neighbours in the current generation it is born. That is, it is live in the next generation.
- Otherwise, a cell's status is unchanged from the current generation to the next.

This process continues indefinitely. Eventually, a generation may be repeated in which case life goes on forever. Or all the cells may die. Similarly, if we explore previous generations that may have led to the current one, we may find one that is necessarily a first generation; that is, it could not have been created from a previous generation by following the rules. Such a generation is known as a Garden of Eden. Given the game parameters and the current generation, you are to determine whether or not the game might have started with a Garden of Eden. If so, output the number of steps necessary to reach the current generation from the Garden of Eden. If there are several possible answers, find the smallest. If there is none, output -1.

Input

Input begins with a single integer, the number of test cases. For each test case, there are *m*+1 lines of input in total. The first line contains the game parameters, which are five integers *m,n,a,b,c* each separated by one space. The constraints are 1<=*m*<=4, 1<=*n*<=5, 1<=*a*<*b*<=8, 1<=*c*<=8. The next *m* lines each contain a string of *n* characters representing a row of the current generation. In the string, an asterisk ("*") indicates live while a period (".") indicates dead. There are no blank lines between test cases.

Output

Output is one integer per test case denoting the minimum number of steps required to reach the input from a Garden of Eden. If there is no Garden of Eden, output -1.

Example

Input:

```
1
4 5 2 3 2
.****
.****
.****
.****
```

Output:

```
2
```

Output Explanation:

Assume the sample input is the "current" generation. A possible previous generation is

```
**.**
..*.*
....*
*****
```

The above generation can be derived from the following previous generation

```
.****
**.*.
*****
*..*.
```

This generation cannot be derived from any other generation. Furthermore, there is no shorter series of generations that has these properties.

Added by: Bobby Xiao
Date: 2007-01-24
Time limit: 5s
Source limit: 50000B
Languages: All
Resource: Canadian Computing Competition, 2006 Senior Stage 1

SPOJ Problem Set (classical)

1268. CN Tower (Easy)

Problem code: CNEASY

Christy C. Coder is traveling to Waterloo for a programming competition. On the way, she stops in Toronto to do some sightseeing. The unfortunate thing about travelling is that everyone back home expects her to bring back pictures of everything. Christy hates taking pictures: it makes her look like such a tourist! Fortunately, Christy has a plan to make her picture-taking quite painless.

At 553 m tall, CN Tower is the world's tallest free-standing building. 351 m up the tower is the "360" rotating restaurant, which rotates a full 360 degrees every 72 minutes. From there, Christy can see the whole city, and take close-up pictures of all the landmarks using her fancy new 100x optical zoom camera. Since the restaurant itself rotates, she only needs to stand in one place to take pictures in all directions.

The waiters insist that she order something or leave, and Christy is not interested in any of the items on the menu. Therefore, she must act quickly before she gets kicked out. Given the locations of the landmarks of which Christy wants to take a picture, your task is to determine the minimum amount of time that she must spend in the restaurant in order for it to rotate enough to bring all the landmarks in view. Assume that Christy always points her camera exactly perpendicular to the window to minimize distortion due to the glass. Note that multiple landmarks may occupy the same (angular) position, and these landmarks would only require one photograph to capture them.

Since the restaurant staff only realize she is a tourist once she starts taking pictures, we begin measuring the time required once she takes her first picture. Therefore, Christy can move to any position in the restaurant without hassle from the restaurant staff and begin taking pictures from there.

Input

The first line of input consists of the number of test cases. For each test case, the first line is an integer n ($2 <= n <= 1000$), the number of landmarks Christy wants to photograph. Each of the following n lines specify a landmark. Each landmark specification consists of the landmark name (a string of uppercase and lowercase letters of length at most 40 characters), a space, and the compass angle d, in degrees, to the landmark from the CN Tower (0 = north, 90 = east, 180 = south, 270 = west). Note that d is a real number which satisfies $0 <= d < 360$, with d specified to the hundredth of a degree (i.e., at most two decimal places).

Output

For each test case, output a single integer, the minimum number of seconds that Christy must remain in the restaurant. If the time is not an integer number of seconds, round it up to the nearest second (i.e., take the ceiling of the number).

Example

Input:

```
1
5
CasaLoma 231.0
OntarioParliament 123.0
SkyDome 75.0
RoyalYorkHotel 340.0
PearsonAirport 165.0
```

Output:

```
3012
```

Added by: Bobby Xiao
Date: 2007-01-24
Time limit: 2s
Source limit: 50000B
Languages: All
Resource: Canadian Computing Competition, 2006 Senior Stage 2

SPOJ Problem Set (classical)

1269. CN Tower (Hard)

Problem code: CNHARD

Christy C. Coder is traveling to Waterloo for a programming competition. On the way, she stops in Toronto to do some sightseeing. The unfortunate thing about travelling is that everyone back home expects her to bring back pictures of everything. Christy hates taking pictures: it makes her look like such a tourist! Fortunately, Christy has a plan to make her picture-taking quite painless.

At 553 m tall, CN Tower is the world's tallest free-standing building. 351 m up the tower is the "360" rotating restaurant, which rotates a full 360 degrees every 72 minutes. From there, Christy can see the whole city, and take close-up pictures of all the landmarks using her fancy new 100x optical zoom camera. Since the restaurant itself rotates, she only needs to stand in one place to take pictures in all directions.

The elevator normally takes 61 seconds to get from the ground up to the rotating restaurant. Unfortunately, when Christy arrives at the CN Tower, she learns that the elevator is out of service, so she has to take the stairs, which takes somewhat longer. Christy arrives at the top at 9:36 pm, by which time it is dark. She can only take pictures with a very powerful flash, which takes a long time to recharge between pictures. While the flash is charging, she cannot take any pictures. Thus, Christy needs a new program to calculate the minimum time that she must spend in the restaurant in order for it to rotate enough to bring all the landmarks in view, and taking into consideration the charging time of the flash. In addition to these difficulties, the restaurant closes at midnight. Thus, Christy may not have enough time to take pictures of all the landmarks.

As before, assume that Christy does not move around in the restaurant after choosing her initial position, but waits for it to rotate to the angle required to take each picture. As with the daytime problem (CNEASY), Christy can (very quickly) pick her initial position, since the restaurant is not that big. Christy always points her camera exactly perpendicular to the window to minimize distortion due to the glass. After taking the last picture, Christy must stay in the restaurant until her flash recharges. Since it is dark outside, if more than one landmark occupies an angular position, Christy can capture **only one one landmark per photo** (in order to keep the desired landmark in focus, blurring all others).

Input

The first line of input consists of the number of test cases. For each test case, the first line is an integer n ($1 <= n <= 1000$), the number of landmarks Christy wants to photograph. Each of the following n lines specify a landmark. Each landmark specification consists of the landmark name (a string of uppercase and lowercase letters), a space, and the compass angle, in degrees (specified with a maximum of 2 decimal places), to the landmark from the CN Tower (0 = north, 90 = east, 180 = south, 270 = west). Finally, the last line contains the amount of time, in seconds, required for the flash to charge.

Output

For each test case, output a single integer, the minimum number of seconds that Christy must remain in the restaurant. If the time is not an integer number of seconds, round it up to the nearest second. If it is not possible for Christy to take all the pictures before closing, instead output "not possible".

Example

Input:

```
1
5
CasaLoma 231.0
OntarioParliament 123.0
SkyDome 75.0
RoyalYorkHotel 340.0
PearsonAirport 165.0
10
```

Output:

```
3022
```

Added by: Bobby Xiao
Date: 2007-01-24
Time limit: 4s
Source limit: 50000B
Languages: All
Resource: Canadian Computing Competition, 2006 Senior Stage 2

SPOJ Problem Set (classical)

1270. Paint By Numbers

Problem code: PNTBYNUM

Years ago, there was a really bad craft/hobby called *paint-by-numbers*: you were given a line drawing, with numbers in each enclosed region, and the number corresponded to a particular colour. An example is shown below (unsolved on the left; solved on the right):

[IMAGE] [IMAGE]
(images from ThisLife.org)

The problem you have to solve is much more linear, in a way.

You will be given an *n*-by-*m* grid (1 <= *n, m* <= 32) which you will "colour" in with either a dot ('.') or a star ('*').

Of course, the grid will not be specified in the usual paint-by-numbers way, since this would be too easy.

Instead, you will you have to infer which cells are blank and which contain a star. The only information you will be given are a collection of *n* + *m* sequences of numbers, one sequence for each row and column. The sequence will indicate the size of each continuous block of stars. There must be at least one dot between two consecutive blocks of stars.

An example is shown below (which is supposed to look fish-like):

[IMAGE]

You may notice that some paint-by-number patterns are not uniquely solvable. For this problem, you may assume that *any* solution which satisfies the specification is correct.

Input

Input begins with a line with the number of test cases. Each test case consists of a total of *n* + *m* + 2 lines. The first line of the test case consists of an integer *n* (1 <= *n* <= 32), the number of rows. The second line consists of an integer *m* (1 <= *m* <= 32), the number of columns. On the next *n* lines, there will be sequences which describe each of the *n* rows (from top to bottom). Each line will contain some positive integers, with a space between adjacent integers, and the sequence will terminate with the integer 0. On the next *m* lines describe the *m* columns (from left to right), the same format as the rows are specified.

Output

Output consists of *n* lines for each corresponding test case, each line composed of *m* characters, where each character is either a dot ('.') or a star ('*'). Separate test cases with a blank line.

Example

Input:

```
2
4
7
2 2 0
5 0
5 0
2 2 0
1 1 0
1 1 0
2 0
2 0
4 0
4 0
2 0
4
4
2 1 0
3 0
3 0
1 1 0
4 0
3 0
3 0
1 0
```

Output:

```
**..**.
..*****
..*****
**..**.

**.*
***.
***.
*.*.
```

Added by: Bobby Xiao
Date: 2007-01-24
Time limit: 5s
Source limit: 50000B
Languages: All
Resource: Canadian Computing Competition, 2006 Senior Stage 2

SPOJ Problem Set (classical)

1296. 4 values whose sum is 0

Problem code: SUMFOUR

The SUM problem can be formulated as follows: given four lists A, B, C, D of integer values, compute how many quadruplet (a, b, c, d) belongs to A x B x C x D are such that a + b + c + d = 0 . In the following, we assume that all lists have the same size n

Input

The first line of the input file contains the size of the lists n (this value can be as large as 4000). We then have n lines containing four integer values (with absolute value as large as 2^{28}) that belong respectively to A, B, C and D .

Output

Output should be printed on a single line.

Example

Input:

```
6
-45 22 42 -16
-41 -27 56 30
-36 53 -37 77
-36 30 -75 -46
26 -38 -10 62
-32 -54 -6 45
```

Output:

```
5
```

Added by: Abhilash I
Date: 2007-02-06
Time limit: 10s
Source limit:50000B
Languages: All
Resource: South western 05-06

SPOJ Problem Set (classical)

1325. Partial Sums

Problem code: PARTSUM

Given a sequence of positive integers a_1, a_2, ..., a_N, and 1 <= i <= j <= N, the partial sum from i to j is $a_i + a_{i+1} + ... + a_j$.

In this problem, you will be given such a sequence and two integers P and K. Your task is to find the smallest partial sum modulo P that is at least K.

For example, consider the following sequence of integers:

```
12      13      15      11      16      26      11
```

Here N = 7. Suppose K = 2 and P = 17. Then, the answer is 2 because 11 + 16 + 26 = 53 and 53 mod 17 is 2. On the other hand, if K = 0 the answer is 0 since 15 + 11 + 16 + 26 = 68 and 68 mod 17 is 0.

You may assume 1 <= N <= 100000.

Input

The first line of the input contains the number of test cases, T.

Each test case begins with a line containing three integers, N, K and P. This is followed by the values of a_1, a_2, ..., a_N, one per line.

Output

Output one line per test case, containing the smallest partial sum modulo P that is at least K, as described above.

Example

Input:

```
1
7 2 17
12
13
15
11
16
26
11
```

Output:

```
2
```

Warning: large Input/Output data, be careful with certain languages

---

Added by: Stephen Merriman
Date: 2007-02-22
Time limit: 7s
Source limit: 50000B
Languages: All
Resource: Indian Computing Olympiad, Online Programming Contest, July 06

SPOJ Problem Set (classical)

1326. A Chase In WonderLand

Problem code: CHASE

Alice is in Wonderland. It is March and March Hare is raving mad. It begins to chase Alice. Alice runs as fast as she can, but she comes to the the edge of a quicksand pool. Now this pool has several safe spots where she may comfortably step on without being swallowed by the quicksand. She may step onto any safe spot from solid ground, but thereafter she can jump from spot to spot only in a straight line, and she cant turn back. March Hare is still hot on her heels, so she needs to know the maximum number of jumps she can make.

Input

On the first line there will be a single integer n, denoting the number of test cases. Each test case will consist of a single integer k by itself on a line, followed by k lines containing the x and y co-ordinates of the safe spots , seperated by a single space. Both coordinates are integer values. There are no leading or trailing spaces or blank lines. 0<k<=2200

Output

For each case print a single integer by itself on a line, with no leading or trailing spaces. Do not print blank lines.

Example

Input:

```
2
5
0 0
1 1
2 2
4 8
2 75
3
0 0
1 2
3 4
```

Output:

```
2
1
```

Added by: Abhilash I
Date: 2007-02-22
Time limit: 1s
Source limit:50000B
Languages: All
Resource: Code Craft 2007

SPOJ Problem Set (classical)

1329. Matrix

Problem code: KPMATRIX

The company you work in has got a secret job to do. Just a few persons know what it is all about. To keep a secret every programmer works on a small part of the project.

Your job is as follows. You are given a matrix of integer numbers with N rows and M columns. Also two integer numbers A and B are given. Your task is to find a number of submatrices of a given matrix with the sum of elements between A and B inclusively.

Input

The first line contains two integer numbers N and M (1 <= N, M <= 250). After that matrix description follows. N lines with M numbers each. The last line contains two integer numbers A and B (-10^9 <= A <= B <= 10^9). All numbers separated with spaces. It's guaranteed that for every submatrix the absolute value of sum of it's elements will not exceed 10^9.

Output

Write to the output the number of submatrices of a given matrix with sum of their elements between A and B inclusively.

Example

Input:
```
3 3
1 0 0
0 1 0
0 0 1
1 3
```

Output:
```
26
```

Added by: Pavel Kuznetsov
Date: 2007-02-23
Time limit: 1s-10s
Source limit:50000B
Languages: All
Resource: IT Festival Arkhangelsk 2006

SPOJ Problem Set (classical)

1335. Maze

Problem code: KPMAZE

The King of Byteland likes Greek mythology very much. The most impressive myth for him is the one about Minotaur. A creature which was imprisoned in a maze-like construction. Now The King wants to have similar maze. He ordered to his architect to build such construction.

The architect decided that maze will have rectangular form. Its floor will be made from large square plates. Also there will be many walls, each of which will separate two common floor plates. From the bird's eye whole construction will look like a grid with some cells separated by walls. The maze should be very tricky, that's why he calls the maze correct if and only if for every two plates there is exactly one path between them. Here path is a sequence of moves between plates that share a common side and are not separated by wall. Each plate can only appear once in a path.

Sooner or later, the architect started his work. After a couple of months he created a rectangular area with H rows and W columns. Also he has built K walls. Sounds perfect but he was seized with a lingering doubt about correctness of his maze.

That's why he asks you to help him. He wants to know how many different correct mazes can be built based on his current maze i.e. you can only add new walls but not to break any of the old ones.

For example (see figure 1.) the maze size is 2x2 and there are no walls. All four ways to complete this maze are shown on the right of the figure (new walls are dashed).

Figure 1

Figure 2. illustrates maze of size 3x3 with 3 walls. There are exactly 4 ways to complete it.

Figure 2

Figure 3. shows the maze that cannot be completed, because there is no path from lower right plate to upper left one.

Figure 3

Input

The first line contains two integer numbers W and H (1 <= W, H <= 5). Second line contains one integer number K (K >= 0). Next K lines contain description of walls. Each wall is determined by two plates it separates. Thus, each line contains four integer numbers: R_1, C_1, R_2 and C_2, here R_1 and C_1 - row and column coordinates of the first plate. Similar, R_2 i C_2 - are coordinates of the second plate (1 <= R_1, R_2 <= H, 1 <= C_1, C_2 <= W). Rows are numbered from up to bottom, colums - left to right started from 1. It is guaranteed that all walls are correct and there are no duplicates. Walls that form perimeter of the maze will not be specified.

Output

Output the number of different correct mazes that can be built based on the given one. There should be no leading zeroes.

Example

Input:
```
2 2
0
```

Output:
```
4
```

Input:
```
3 3
3
3 1 3 2
2 2 2 3
2 3 3 3
```

Output:
```
4
```

Input:
```
3 3
5
3 1 3 2
2 2 2 3
2 3 3 3
2 2 2 1
1 2 2 2
```

Output:
```
0
```

Added by: Pavel Kuznetsov
Date: 2007-02-24
Time limit: 1s
Source limit: 50000B
Languages: All
Resource: IT Festival Arkhangelsk 2006

SPOJ Problem Set (classical)

1391. Summing to a Square Prime

Problem code: CZ_PROB1

S_{P2} = {p | p = $x_1^2 + x_2^2$ for some x_1, x_2 belonging to Z} is the set of all primes that can be represented as the sum of any two squares. The function $S_{P2}(n)$ gives the n^{th} prime number from the set S_{P2}. Now, given two integers n (0<n<501) and k (0<k<4), find p($S_{P2}(n)$, k) where p(a, b) gives the number of unordered ways to sum to the given total 'a' with 'b' as its largest part. For example: p(5, 2) = 3 {2+2+1, 2+1+1+1, and 1+1+1+1+1}. Here 5 is the total with 2 as the largest part.

Input

The first line gives the number of test cases T followed by T lines of integer pairs, n and k.
Scope:
$0 < T < 501$
$0 < n < 501$
$1 < S_{P2}(n) < 7994$
$0 < k < 4$

Output

The p($S_{P2}(n)$, k) for each n and k. Append a newline character to every test cases' answer.

Example

Input:
```
3
2 2
3 2
5 3
```

Output:
```
3
7
85
```

Added by: Rahul
Date: 2007-03-10
Time limit: 2s
Source limit: 3000B
Languages: All
Resource: Sam Collins

SPOJ Problem Set (classical)

1417. University Employees

Problem code: EMP

On some island each inhabitant is either a knight who only tells the truth, or a liar who always lies. Also, on the island there exists a university of technology where some of the inhabitants work. All of the university employees will always tell you the following two things, no matter which employee you ask:

1. There are fewer than *N* employees who work more than me.

2. At least *M* employees of the university have a larger salary than me.

It is also known that no two employees of the university have an identical salary, and no two work equally. Write a program which will compute how many persons are employed by this university.

Input

The only input line contains two integers *N* and *M*, with one space between them [*N, M* <= 1000000000].

Output

The output must contain only one integer - the total number of employees of this university, or 0 if there is no way to find the number of employees.

Example

Input:
```
1 1
```

Output:
```
2
```

Author: Filimonenkov D.O.

Added by: Roman Sol
Date: 2006-04-24
Time limit: 1s
Source limit:50000B
Languages: All
Resource: ZCon 2007

SPOJ Problem Set (classical)

1418. The Cats and the Mouse

Problem code: CATM

In a rectangular field of size n by m squares there is a mouse and two cats. The mouse is the first to make a move, then each of the cats makes a move, then again its the mouse's turn, and so on. In each move both the mouse and the cats can move exactly one square vertically or horizontally. If the mouse is standing at the edge of the field then in its next move it can jump off the field and is saved from the cats. If in the next move one of the cats moves to the field with the mouse then there is no escape for the mouse ... =(
You are to write a program which, knowing the initial positions of mouse and the two cats, will find out if there is any way for the mouse to escape from the cats, assuming of course that each cat will do its best to catch the mouse.

Input

In the first line of input two integers n and m are given, not exceeding 100, where n is the number of rows, and m - the number of columns. The second line contains a number k [k <= 10], which defines the number of test cases for the given field. In the next k lines the initial positions of the mouse and the cats are given. The position in the field is given by two numbers: the first is the number of the row, the second is the number of the column. The first two integers are the coordinates of the mouse, the next four integers are the coordinates of the cats.

Output

You must output k lines with answers for each test case. The answer is YES, if the mouse can escape or NO otherwise.

Example

Input:
```
5 3
3
2 2 1 1 3 3
2 3 1 3 5 2
3 2 1 2 4 3
```

Output:
```
NO
YES
YES
```

Author: Filimonenkov D.O.

Added by:	Roman Sol
Date:	2006-05-04
Time limit:	1s
Source limit:	50000B
Languages:	All
Resource:	ZCon 2007

SPOJ Problem Set (classical)

1419. A Game with Numbers

Problem code: NGM

Nikifor and Trofim play the following game: they write some integer smaller then 2000000000 and take turns one after another. Nikifor is the first to make a move. The turn is made by the following rule: from the written integer any non-zero digit is subtracted, and the new integer replaces the old one on the desk. For example for integer 40534, the next move can be: 40530, 40531 or 40529. The winner is the player who writes zero on the desk.
Write a program to decide who will win if both players do their best.

Input

The input contains the integer from which the game is started.

Output

In the first line you must write 1 if Nikifor wins and 2 otherwise. If Nikifor wins then in the second line you must output the move in the first turn which guarantees victory for him. If there are many such moves then output any of them.

Example

Input:
```
14
```

Output:
```
1
4
```

Author: Filimonenkov D.O.

Added by: Roman Sol
Date: 2006-05-05
Time limit: 1s
Source limit:50000B
Languages: All
Resource: ZCon 2007

SPOJ Problem Set (classical)

1420. Geometry and a Square

Problem code: GEOM

Is there anyone who doesn't love geometry?! Just imagine: on the plane you are given a square ABCD, with vertices given in the clockwise direction. Also given is a point P which is different from all of A, B, C or D. Have you imagined it? Interested? Ok, let's continue!
Through vertex A a line *a* is drawn that is perpendicular to line BP, through vertex B a line *b* is drawn that is perpendicular to line CP, through vertex C a line *c* is drawn that is perpendicular to line DP, through vertex D a line *d* is drawn that is perpendicular to line AP. Do the lines *a*, *b*, *c* and *d* cross each other in one point? Ok, it depends on what the square is and what point P is given. Write the program that discovers if these lines cross in one point, and if so, finds the coordinates of this point.

Input

In the first line you are given the integer coordinates of the point in which diagonals of the square intersect. In the second line you are given one integer - the length of the side of the square. In the third line you are given the integer coordinates of point P. The integers do not exceed 100, in terms of absolute value.

Output

For each test case you must output YES if the sought point exsists, and NO otherwise. If you answer YES then in the second line you must output the coordinates of the intersection point. Coordinates must be rounded to one digit after the point.

Example

Input:
```
10 10
20
5 12
```

Output:
```
YES
8.0 5.0
```

Author: Filimonenkov D.O.

Added by: Roman Sol
Date: 2006-05-05
Time limit: 1s
Source limit:50000B
Languages: All
Resource: ZCon 2007

SPOJ Problem Set (classical)

1421. Goods

Problem code: FIRM

There are *n* dealers in the market. Each of them has some unique goods (nobody else has the same goods). Besides, each of them wants to obtain some other goods, which exist in the market. This is rather strange, but for each kind of goods on the market there exists exactly one dealer who wants to obtain it.
To prevent fraud, only exchanges in pairs are allowed in this market. Moreover, each dealer is allowed to make at most one exchange a day. But the total number of transactions isn't limited. A transaction means that all the goods of one dealer are exchanged for all the goods of the other participating dealer (partial transactions are not allowed).
You are to write a program which outputs the minimum number of days needed for each dealer to get the goods that he wants. Also output one of the possible variants of exchanges leading to this goal.

Input

The first line contains an integer *n* [*n* <= 5000]. In the second line exactly *n* numbers of goods are given, which the dealers require. If integer *j* appears as the *i*-th at input, then this means that goods required by dealer *i* are initially owned by dealer *j*.

Output

You must output the minimum number of days *m* which are needed to complete the transactions. In the next *m* lines you must output the way these transactions should be managed by the dealers. One line corresponds to one day. At the beginning of each line you must output the number of transactions on this day. After that output the pairs of dealers who exchange their goods on this day. Dealers in pairs are separated by '-' symbol. If there are many ways to perform the exchanges then output any of them.

Example

Input:
```
7
2 1 3 5 6 7 4
```

Output:
```
2
3 1-2 4-5 7-6
1 5-7
```

Author: Filimonenkov D.O.

Added by:	Roman Sol
Date:	2006-05-05
Time limit:	1s-10s
Source limit:	50000B
Languages:	All
Resource:	ZCon 2007

SPOJ Problem Set (classical)

1431. Projections Of A Polygon

Problem code: KPPOLY

You are given a convex polygon on Cartesian coordinate system. It has projections on X and Y-axis. You can arbitrary rotate this polygon. What minimum and maximum sum of projections can you achieve?

Input

First line contains one integer number N (3 <= N <= 100) - number of polygon's vertices. Following N lines contain vertex coordinates X_i and Y_i. All numbers are integers. Vertices are given in clockwise or anticlockwise direction. No two vertices coincide. No three consecutive vertices lie on the same line. All coordinates do not exceed 10000 by absolute value.

Output

Write minimum and maximum value of sum of the polygon's projections. Separate them by a space. Your answer should not differ with the correct one more than 10^{-6}.

Example

Input:
```
4
0 0
0 1
1 1
1 0
```

Output:
```
2 2.828427124
```

Added by: Pavel Kuznetsov
Date: 2007-03-25
Time limit: 1s
Source limit:50000B
Languages: All
Resource: Vologda 2007

SPOJ Problem Set (classical)

1433. The Sum

Problem code: KPSUM

One of your friends wrote numbers 1, 2, 3, ..., N on the sheet of paper. After that he placed signs + and - between every pair of adjacent digits alternately. Now he wants to find the value of the expression he has made. Help him.
For example, if N=12 then +1 -2 +3 -4 +5 -6 +7 -8 +9 -1+0 -1+1 -1+2 = 5

Input

Each line contains one integer number N ($1 <= N <= 10^{15}$). Last line contains 0 and shouldn't be processed. Number of lines in the input does not exceed 40.

Output

For every line in the input write the answer on a separate line.

Example

Input:
```
12
0
```

Output:
```
5
```

Added by: Pavel Kuznetsov
Date: 2007-03-26
Time limit: 1s
Source limit: 50000B
Languages: All
Resource: Vologda 2007

SPOJ Problem Set (classical)

1434. Equation

Problem code: KPEQU

You are given integer positive number N. Find the number of solutions in positive integer numbers of the following equation:

$$1/N! = 1/X + 1/Y$$

Input

Each line of input file contains one integer number N ($1 <= N <= 10^4$). The last line contains 0 and shouldn't be processed. Number of lines in the input does not exceed 30.

Output

For every line in the input write the answer on a separate line.

Example

Input:

```
1
2
0
```

Output:

```
1
3
```

Added by: Pavel Kuznetsov
Date: 2007-03-27
Time limit: 1s
Source limit:50000B
Languages: All
Resource: Vologda 2007

SPOJ Problem Set (main)

1435. Vertex Cover

Problem code: PT07X

You are given an unweighted, undirected tree. Write a program to find a vertex set of minimum size in this tree such that each edge has as least one of its end-points in that set.

Input

The first line of the input file contains one integer *N* --- number of nodes in the tree (0 < *N* <= 100000). Next *N*-1 lines contain *N*-1 edges of that tree --- Each line contains a pair (*u*, *v*) means there is an edge between node *u* and node *v* (1 <= *u*,*v* <= *N*).

Output

Print number of nodes in the satisfied vertex set on one line.

Example 1

Input:
```
3
1 2
1 3
```

Output:
```
1
```

Explanation:
```
The set can be {1}
```

Example 2

Input:
```
3
1 2
2 3
```

Output:
```
1
```

Explanation:
```
The set can be {2}
```

Added by: Thanh-Vy Hua
Date: 2007-03-28
Time limit: 1.200s-8s
Source limit:50000B
Languages: All
Resource: Co-author Amber

SPOJ Problem Set (classical)

1436. Is it a tree

Problem code: PT07Y

You are given an unweighted, undirected graph. Write a program to check if it's a tree topology.

Input

The first line of the input file contains two integers *N* and *M* --- number of nodes and number of edges in the graph (0 < *N* <= 10000, 0 <= *M* <= 20000). Next *M* lines contain *M* edges of that graph --- Each line contains a pair (*u*, *v*) means there is an edge between node *u* and node *v* (1 <= *u*,*v* <= *N*).

Output

Print *YES* if the given graph is a tree, otherwise print *NO*.

Example

Input:
```
3 2
1 2
2 3
```

Output:
```
YES
```

Added by: Thanh-Vy Hua
Date: 2007-03-28
Time limit: 1s
Source limit: 50000B
Languages: All
Resource: Co-author Amber

SPOJ Problem Set (classical)

1437. Longest path in a tree

Problem code: PT07Z

You are given an unweighted, undirected tree. Write a program to output the length of the longest path (from one node to another) in that tree. The length of a path in this case is number of edges we traverse from source to destination.

Input

The first line of the input file contains one integer *N* --- number of nodes in the tree (0 < *N* <= 10000). Next *N*-1 lines contain *N*-1 edges of that tree --- Each line contains a pair (*u*, *v*) means there is an edge between node *u* and node *v* (1 <= *u*,*v* <= *N*).

Output

Print the length of the longest path on one line.

Example

Input:
```
3
1 2
2 3
```

Output:
```
2
```

Added by: Thanh-Vy Hua
Date: 2007-03-28
Time limit: 1s
Source limit:50000B
Languages: All
Resource: Co-author Amber

SPOJ Problem Set (classical)

1440. Use of Function Arctan

Problem code: ARCTAN

It's easy to know that arctan(1/2)+arctan(1/3)=arctan(1).The problem is,to some fixed number A,you have to write a program to calculate the minimum sum B+C.A,B and C are all positive integers and satisfy the equation below:

arctan(1/A)=arctan(1/B)+arctan(1/C)

Input

The first line contains a integer number T.T lines follow,each contains a single integer A, 1<=A<=60000.

Output

T lines,each contains a single integer which denotes to the minimum sum B+C.

Example

Sample input:

```
1
1
```

Sample output:

```
5
```

Some new test data has been added on Feb.15, 2009, 36 users lose their Accepted.

Added by: Blue Mary
Date: 2007-03-31
Time limit: 10s
Source limit:256B
Languages: All except: C99 strict
Resource: Chinese National Olympiad in Informatics 2001,Day 1; translated by Blue Mary

SPOJ Problem Set (classical)

1441. The Clever Typist

Problem code: CLEVER

Blue Mary is a typist of some secret department.Now she have to type in many passwords in an hour,each of which has a fixed length: 6.Of course,the less times she presses the keyboard,the happier she is.

Unfortunately,the keyboard to type in the password is extraordinary designed to keep secrets.The keyboard has 6 particular keys instead of 10 number keys.To explain the uses of these keys,let's define the 6 position on the screen 1,2,3,4,5,6 from left to right.The keys' uses are shown below:

- Swap0: swap the digit in the cursor position and the digit in position 1.The cursor doesn't move.If the cursor is now in position 1,the digits on the screen won't be changed.
- Swap1: swap the digit in the cursor position and the digit in position 6.The cursor doesn't move.If the cursor is now in position 6,the digits on the screen won't be changed.
- Up: increase the digit in the cursor position by 1.If the digit in the cursor position is 9,no change will happen.
- Down: decrease the digit in the cursor position by 1.If the digit in the cursor position is 0,no change will happen.
- Left: move the cursor one position left.If the cursor is in position 1,no change will happen.
- Right: move the cursor one position right.If the cursor is in position 6,no change will happen.

At start,6 random digits will be given on the screen,and the cursor will in position 1.After some smart presses,she can type in the correct password,at that time the cursor position is unimportant.

Here is an example("()"denotes to the cursor):

```
key pressed       screen
                  (1)23456
Swap1             (6)23451
Right             6(2)3451
Swap0             2(6)3451
Down              2(5)3451
Right             25(3)451
Up                25(4)451
Right             254(4)51
Down              254(3)51
Right             2543(5)1
Up                2543(6)1
Swap0             6543(2)1
```

Now Mary wants to know the minimal number of keys she has to press.Can you help her?

Input

The first line contains a single integer t.t lines follow,each contains two 6-digit string,which show the digits on the screen at start and the password Mary is to type in,separated by a single space.

Output

t lines,each contains a single integer - the answer.

Example

Sample input:

```
1
123456 654321
```

Sample output:

```
11
```

Some new test data has been added on Feb 15,2009. 26 users lose their Accepted.

Added by: Blue Mary
Date: 2007-03-31
Time limit: 7s
Source limit:50000B
Languages: All except: C99 strict
Resource: Chinese National Olympiad in Informatics 2001,Day 1; translated by Blue Mary

SPOJ Problem Set (classical)

1442. Strange Food Chain

Problem code: CHAIN

There are 3 kinds of animals A,B and C. A can eat B,B can eat C,C can eat A. It's interesting,isn't it?

Now we have n animals,numbered from 1 to n. Each of them is one of the 3 kinds of animals:A,B,C.

Today Mary tells us k pieces of information about these n animals. Each piece has one of the two forms below:

- 1 x y: It tells us the kind of x and y are the same.
- 2 x y: It tells us x can eat y.

Some of these k pieces are true,some are false. The piece is false if it satisfies one of the 3 conditions below, otherwise it's true.

- X or Y in this piece is larger than n.
- This piece tells us X can eat X.
- This piece conflicts to some true piece before.

Input

The first line contains a single integer t.t blocks follow.

To every block,the first line contains two integers n(1<=n<=50000) and k (1<=k<=100000). k lines follow,each contains 3 positive integers D(1<=D<=2),X,Y,separated by single spaces.

Output

t lines,each contains a single integer - the number of false pieces in the corresponding block.

Example

Sample input:

```
1
100 7
1 101 1
2 1 2
2 2 3
2 3 3
1 1 3
2 3 1
1 5 5
```

Sample output:

```
3
```

Hint:

```
The false pieces are the 1st,the 4th and the 5th ones.
```

Warning: large Input/Output data, be careful with certain languages

---

Added by: Blue Mary
Date: 2007-03-31
Time limit: 1s
Source limit:50000B
Languages: All except: C99 strict
Resource: Chinese National Olympiad in Informatics 2001,Day 1; translated by Blue Mary

SPOJ Problem Set (classical)

1444. DEL Command II

Problem code: DELCOMM2

It is required to find out what's the maximum number of files that can be deleted from MS-DOS directory executing the DEL command of MS-DOS operation system only once. There are no nested subdirectories.

A note

DEL command has the following format: `DEL` *wildcard*

The actual wildcard as well as a full file name can be made up of a name containing 1 up to 8 case-sensitive characters. In a wildcard the characters '?' and '*' can be used. A question mark substitutes exactly one character of the full file name, an asterisk any sequence of characters even empty one.

MS-DOS system can permit maybe other wildcards but they can not be used in this task. File names consist only of Latin letters and digits.

Input

The first line of the input is an integer M, then a blank line followed by M datasets. There is a blank line between datasets.

Input data for each dataset contains a list of full file names without any extra empty lines and spaces. Each name is written in a separate line of input data file and ended with a control sign: '+' for delete or '-' for keep. Full file names are not repeated. The list comprises at least one file, and at least one file is marked to be deleted. There are no more than 250 files.

Output

For each dataset, write to the first line of output the maximum number of files one DEL command can delete.

Example

Input:

```
2

BP +
BPC +
TURBO -

EXCHANGE +
EXT +
HARDWARE +
MOUSE -
```

```
NETWORK -
```

Output:

```
2
2
```

Hint:

For the two tests above,the corresponding DEL commands are DEL BP* and DEL EX*.

Link

You can try problem DELCOMM first. It's far easier than this problem.

Added by: Blue Mary
Date: 2007-04-01
Time limit: 35s
Source limit:50000B
Languages: All except: C99 strict
Resource: Chinese National Olympiad in Informatics 1997,Day 1

SPOJ Problem Set (classical)

1447. A Game of Toy Bricks

Problem code: BRCKGAME

Blue Mary invents a game with toy bricks. The player has N cuboids numbered from 1 to N.

The rule of the game is discribed below:

- Choose some cuboids among the N cuboids, and divide them into M(1 <= M <= N) piles,named them $Pile_1$,$Pile_2$ ... $Pile_M$. There are at least 1 cuboid in each pile. To make the game easier, for any cuboid in $Pile_K$,its id should greater than any one in $Pile_{K+1}$ (1 <= K < M).
- For each pile of cuboids,the player will put them as a tower, and he should follow the two rules below:
- The up surface of each cuboid is touched and only touched another down surface. Luckily,to make the pile looking like a tower,the up surface of the lower cuboid should cover the down surface of the higher cuboid,i.e. the length of the lower up surface is not less than that of the higher down surface, and also to the width.
- In each pile,the lower cuboid has a less id than the higher cuboid.

Your task is to find a method,to make the sum of the height of each pile maximum.

Input

The very first line of the input contain the number t,then t cases follow.

For each case,The first line contain two integer number N and M. N(N<=100) is the total number of the cuboids, M(M<=N) is the number of the piles, separated by a single space.

Then N line follow, which are the description of the cuboids 1..N. Each line contains three integer numbers(<=1000)- the length, width and height of that cuboid,separated by spaces.

Output

For each case, the output contains only one line with a single integer number - the maximum sum.

Example

Sample Input:

```
1
4 2
10 5 5
8 7 7
2 2 2
6 6 6
```

Sample Output:

```
24
```

Added by: Blue Mary
Date: 2007-04-01
Time limit: 3s
Source limit:50000B
Languages: All except: C99 strict
Resource: Chinese National Olympiad in Informatics 1997,Day 2; translated by Blue Mary

SPOJ Problem Set (classical)

1448. 3D Cover

Problem code: COVER2

In the 3D Cartesian coordinate system, there are n cubes.These cubes are all axis-paralleled. What's the volume of the union of these cubes?

Input

There is a single integer m in the very first line of the input, the number of test cases. m blocks follow.

For each test, the first line contains a single integer n(1<=n<=100), the number of cubes. n lines follow, each contains four integers x,y,z,r(-1000<=x,y,z<=1000,1<=r<=200), separated by spaces. x,y,z are the X,Y,Z coordinates of the center of the cube, and r is the distance between the center and any surface of the cube.

Output

m lines,each contains a single integer - the answer.

Example

Sample Input:

```
1
3
0 0 0 3
1 -1 0 1
19 3 5 6
```

Sample Output:

```
1944
```

Added by: Blue Mary
Date: 2007-04-01
Time limit: 2s
Source limit:50000B
Languages: All except: C99 strict
Resource: Chinese National Olympiad in Informatics 1997,Day 2; description by Blue Mary

SPOJ Problem Set (classical)

1451. 01 Sequence

Problem code: SEQ1

The input consists of exactly 5 test cases in the following format:

Input

`N A0 B0 L0 A1 B1 L1 [3<=N<=1000,1<=A0<=B0<=L0<=N,1<=A1<=B1<=L1<=N]`

Output

Exactly 5 lines,each contains:

a) A sequence (We name it S) consisting only characters ’0’ and ’1’ and no extra whitespaces, which satisfy the following conditions:

- The number of ’0’ in any consecutive subsequence of S whose length is L0 is not more than B0 and not less than A0.
- The number of ’1’ in any consecutive subsequence of S whose length is L1 is not more than B1 and not less than A1.

or

b)A single number -1,if the sequence which satisfies the conditions above doesn’t exist.

Example

Input:

```
6 1 2 3 1 1 2
[and 4 test cases more]
```

Output:

```
010101
[and 4 test cases more]
```

Added by: Blue Mary
Date: 2007-04-01
Time limit: 1s
Source limit:50000B
Languages: All except: C99 strict
Resource: Chinese National Olympiad in Informatics 1999,Day 1; translated by Blue Mary

SPOJ Problem Set (classical)

1452. Birthday Cake

Problem code: CAKE

Adolf wants to send a cake to Blue Mary to celebrate her birthday. The cake looks like a tower which has M floors, each floor is a cylinder. The i-th cylinder counted from downside to upside has a integer height h_i and a integer radius r_i. These numbers fulfill the following two conditions:

- $h_1 > h_2 > h_3 > ... > h_M$
- $r_1 > r_2 > r_3 > ... > r_M$

Adolf is interested in minimising the area of the surface of the cake, except for the underside of the lowest cylinder. He needs your help because of his poor math knowledge.

Input

The very first line contains a integer number T. T test cases follow.

For each test case, the first line contains a single integer number N (N<=10000), the second line contains a single integer number M (M<=10). The cake must be made of M cylinders and its volume must be N*Pi.

Output

For each test case, a single line containing a single integer S must be written to output. The required minimum area must be S*Pi.

Example

Sample Input:
```
1
100
2
```

Sample Output:
```
68
```

Added by: Blue Mary
Date: 2007-04-01
Time limit: 1s
Source limit:50000B
Languages: All except: C99 strict
Resource: Chinese National Olympiad in Informatics 1999,Day 1; translated by Blue Mary

SPOJ Problem Set (classical)

1453. Optimal Connected Subset

Problem code: OPTSUB

- A point $P(x,y)$ is called an integer point if and only if both x and y are integers.
- W is the set which contains all the integer points on the plane.
- Two integer point $P(x,y)$ and $Q(x',y')$ are called adjacent if and only if $|x-x'|+|y-y'|=1$.
- S is a set of integer points if and only if S is a limited subset of W.
- If S is a set of integer points, R and T belong to S,and there exist a limited integer point sequence $Q_0(=R),Q_1,Q_2,...,Q_k,Q_{k+1}(=T)$ which satisfies that
  - $Q_i != Q_j$ iff $i!=j$
  - Q_i and Q_{i+1} are adjacent, for each $0<=i<=k$.
  - Q_i belongs to S, for each $0<=i<=k+1$.

  we call R and T are connected and the sequence $Q_i(0<=i<=k)$ is a path that connect R and T.
- If S is a set of integer points, X and Y are some integer points that belong to S, there exists one and only one path connected X and Y, then S is called an optimal set.

Given an optimal set V, your task is to find an optimal set B, satisfying that B is a subset of V and the sum of the weights of each integer point is maximum.

Input

The very first line of the input contains a single integer T, the number of test cases. T blocks follow.

For each test case, the first line contains a single integer $N=|V|(N<=1000)$. N lines follow, each contains 3 integers, the X-coordinate, the Y-coordinate and the weight(the absolute value of the weight<=100) of the ith integer point, separated by single spaces.

Output

T lines,each contains a single integer - the maximum sum of weights.

Example

Input:
```
1
5
0 0 -2
0 1 1
1 0 1
0 -1 1
-1 0 1
```

Output:
```
2
```

Added by: Blue Mary
Date: 2007-04-01
Time limit: 1s
Source limit:50000B
Languages: All except: C99 strict
Resource: Chinese National Olympiad in Informatics 1999,Day 2; translated by Blue Mary

SPOJ Problem Set (classical)

1454. Memory Distribution

Problem code: MEMDIS

EMS memory (called **memory** for short) is some important resource of a computer. When a program is running, the computer must distribute the memory for it.

The classical memory distribution process is like the following:

- The basic unit of the memory is called a **cell**, each cell is assigned to a fixed integer number called its **address**. The address starts from number 0. If two cell's address numbers are two consecutive integer numbers, the two cells are called (logically) consecutive. We name the *s* consecutive cells starting from address *i* a piece whose length is *s* and first address is *i*.
- Many programs need memory during their running. For each program, we need a application time *X*, a number of cells needed *M* and a running time *P* to describe it. When the program is running(during it starts running (time *T*) to *T+P*, including the left end and excluding the right end),The M cells cannot be used by other programs.
- Suppose a program apply *M* cells at time *X* and its running time is *P*,then
    - (A) If there is a piece in the memory at time *X*, each cell of which is not being used, and whose length is *M*, the computer will distribute these *M* cells to this program. If there are more pieces, the computer will choose the one whose first address is minimum.
    - (B) If such piece does not exist at time *X*, the program will be put into a queue.If after some time, there exist a piece whose length is *M* and the corresponding program is at the head of the queue, the computer will pop this program and distribute memory for it like (A) immediately. During the process of memory distribution, the programs which are not at the head of the queue cannot start to run before the one at the head of the queue.

Input

Ten test cases (given one after another, you have to process all!).For each test case:

The first line is a number N, which shows the number of memory cells. There addresses are 0..n-1. Less than 10000 lines follow, each contains 3 integers X, M(M<=N), P describing the programs. A line containing three zeroes denotes the end of a test case. The programs have been sorted by there application time X.

All numbers in the input and output file will be less than 10^9.

Output

For each test case output two lines. The first line contains a single integer, which shows the time when all the programs have been run and stops normally. The second line contains a single integer, which is the number of programs which has been put into the queue.

Example

Input:

```
10
1 3 10
2 4 3
3 4 4
4 1 4
5 3 4
0 0 0
[and 9 test cases more]
```

Output:

```
12
2
[and 9 test cases more]
```

Added by: Blue Mary
Date: 2007-04-01
Time limit: 17s
Source limit: 50000B
Languages: All except: C99 strict
Resource: Chinese National Olympiad in Informatics 1999,Day 2; translated by Blue Mary

SPOJ Problem Set (classical)

1455. Program Analyser

Problem code: ANALYSER

Input

A Program which has the following format:

```
<Program>::=<sentence><line break>{<sentence><line break>}
<setence>::=<level><space><body>
<body>::=<addition> | <output> | <goto> | <condition> | <end>
<addition>::=<variable>+<integer>
<output>::=<variable>?
<goto>::=GO<space><level>
<condition>::=IF<space><variable>=<integer><space><goto>
<end>::=END
<variable>::=<character>
<level>::=<integer>
<integer>::=<digit>{<digit>}
<character>::=A|B|C|D|E|F|G|H|I|J|K|L|M|N|O|P|Q|R|S|T|U|V|W|X|Y|Z
<digit>::= 0|1|2|3|4|5|6|7|8|9
<line break>::=(ASCII 10)
<space>::=(ASCII 32)
```

The program runs following the following rules:

- Program starts from the sentence whose level is minimum, and executed by the level from low to high except that the sentence is<goto>or<condition>.
- All variables are initialized to 0.
- <Addition>means<variable>+=<integer>in C++.
- <output>means write the value of<variable>to the output file(we aren't interesting about the real output file.)
- <condition>means if and only if the value of the <variable> equals to <integer>, <goto> will be executed, otherwise the next sentence executed is as usual.
- After<goto>, the next sentence executed is the sentence with level which equals to the level in<goto>.
- Program terminates by itself when <end> is executed.
- The numbers during the program is running will fit in a signed 32-bit integer.
- The number of sentences in the program is not more than 100.
- The length of each line in the input file is not more than 20.
- The input is correct.
- The sentence with the maximum level is always <end>.
- Any level is not more than 3000.

Input terminate by EOF.

Output

Output the number of sentences executed.If the program can not terminate by itself,output -1.

Example

Input:

```
10 A+1
20 IF A=5 GO 60
60 END
30 A+2
40 A?
50 GO 20
```

Output:

```
11
```

Hint:

```
10->20->30->40->50->20->30->40->50->20->60
```

Note

You may try problem ANALYS_T first. It's the same problem with this one and its time limit is not so strict.

Log

The time limit of this problem has been changed from 0.4/0.5 second to 1 second per test on Jun.5, 2008. All the solutions have been rejudged.

Added by: Blue Mary
Date: 2007-04-01
Time limit: 1s
Source limit:50000B
Languages: All except: C99 strict
Resource: Chinese National Olympiad in Informatics 2000,Day 1; translated by Blue Mary

SPOJ Problem Set (classical)

1457. Help Blue Mary Please! (Act I)

Problem code: BLUEEQ

This morning Blue Mary wrote some equations on a piece of paper and left it on her desk. After solving some problems in SPOJ, she found that her classmate H.L. replaced all characters on the paper with some other ones. H.L. told her he replaced the same characters with the same ones, and different characters with different ones because of his goodness. Now Mary needs your help to get the original equations back.

In Mary's equations, only 13 characters appear:0,1,2,3,4,5,6,7,8,9,+,*,=. There is one and only one "=" in each equation. In H.L.'s equations,only 13 Latin letters appear: a,b,c,d,e,f,g,h,i,j,k,l,m. All the equations are correct in decimal notation.

For example. If Mary wrote down 2+29=31, H.L. replaced 2 with i, + with l, 9 with k, = with e, 3 with m and 1 with a, we got ilikema .

Input

The first line contains a single integer t.t blocks follow.

To every block,the first line contains a single integer n(1<=n<=1000). n lines follow,each contains a string whose length is more than 4 and less than 12.The string contains only a-m and doesn't contain any whitespaces.

At least 90% of test cases satisfy that n<=5.

At least 80% of test cases satisfy that n<=2.

In at least 70% of test cases, there are at most 5 different characters in all the equations.

Output

If there doesn't exist n equations that can be translated to H.L.'s equations,print a line contains the word *noway* .Otherwise you should output all the corresponding relations that can be fixed in lexicographic order,see the example.

Example

Input

```
1
2
abcdec
cdefe
```

Output

```
a6
b*
d=
f+
```

hint

The two strings can be translated to the following equations possibly:

```
6*2=12 2=1+1
6*4=24 4=2+2
6*8=48 8=4+4
```

So the corresponding relations above can be fixed,others can not.

Added by: Blue Mary
Date: 2007-04-01
Time limit: 17s
Source limit:50000B
Languages: All except: C99 strict
Resource: Chinese National Olympiad in Informatics 2000,Day 2; translated by Blue Mary

SPOJ Problem Set (classical)

1458. Help Blue Mary Please! (Act II)

Problem code: BLUEEQ2

Today Mary’s math homework is to solve an equation.

[IMAGE]

She knows all ki and pi, and 1<=xi<=M. All xi must be integers. She must work out the number of different solutions of this equation this day.Can you give her a hand?

Input

There is a single integer T in the very first line of the input denoted the number of tests.T blocks follow.

For each test case:

The first line contains a single integer n(n<=6).The second line contains a single integer m(m<=150).n lines follow,each contains two space-separated integers ki and pi, i=1,2,...,n. All pi are positive.

[IMAGE]

Output

For each test case output a single line ,which contains a single integer - the answer.You may assume this number is less than 2^31.

Example

Input:
```
1
3
150
1 2
-1 2
1 2
```

Output:
```
178
```

Warning: The time limit is very strict for this problem.

Blue Mary’s Note: test data were modified on Dec.4, 2007. All the solutions have been rejudged.

Added by: Blue Mary
Date: 2007-04-01
Time limit: 11s
Source limit:50000B
Languages: All except: C99 strict
Resource: Chinese National Olympiad in Informatics 2001,Day 2; translated by Blue Mary

SPOJ Problem Set (classical)

1459. The Secret of an Aerolite

Problem code: AEROLITE

A huge aerolite had fallen in Antarctica!!!Many disasters happened and lots of people lost their lives, homes or even everything.Blue Mary, the well-known scientist, is to work out the secret of this aerolite for further study.

Blue Mary has found out that their are some numbers on this aerolite, 5 per line:

```
1 1 1 1 6
0 0 6 3 57
8 0 11 3 2845
```

With her genius, Blue Mary knows that the 5th number is the key to solve the problem, but in some lines the 5th numbers were destoryed and we cannot know what they are instantly.

After some days, she finds out the way to calculate the 5th number finally, which is:

- A Regular Expression(RE) is a string which only contains characters '{','[','(',')',']','}' and satisfies the rules below.
- An empty string is an RE.
- If there's no character '[',']','{','}' in RE A, then (A) is an RE.
- If there's no character '{','}' in RE A, then [A] is an RE.
- If A is an RE, {A} is an RE.
- If both A and B are REs, AB is an RE.

For example "()(())[]", "{()[()]}", "{{[[(())]]}}"(all without quotes) are REs and "()([])()","[()" are not REs.

The deep of an RE A, D(A), is defined as below:

- If A is an empty string,D(A)=0;
- If A can be written as BC, D(A)=max(D(B),D(C));
- If A can be written as (B) or [B] or {B}, D(A)=D(B)+1.

Such as D("(){()}[]")=2.

Suppose the first 4 numbers in current line are L1,L2,L3 and D, then the 5th number in current line is the number of REs modudo 11380.Each of the REs must have a depth of D and is made of L1 {}, L2 [] and L3 ().

Now Mary needs your help to work out the 5th number.

Input

Input contains exactly 10 test cases.Each test case contains one line, which contains 4 numbers L1,L2,L3,D(0<=L1,L2,L3<=10, 0<=D<=30), separated by single spaces.

Output

Ten lines, each contains a single integer denoted the 5th number.

Example

Input:

```
1 1 1 1
0 0 6 3
1 1 1 2
[and 7 test cases more]
```

Output:

```
6
57
8
[and 7 test cases more]
```

Added by: Blue Mary
Date: 2007-04-01
Time limit: 7s
Source limit:50000B
Languages: All except: C99 strict
Resource: Chinese National Olympiad in Informatics 2001,Day 2; translated by Blue Mary

SPOJ Problem Set (classical)

1460. A Simple Calculator in the Galaxy

Problem code: GALAXY

Background

So if you want to survive the Earth's demise and see the galaxy, make sure you are wearing something more substantial than pajamas and a ratty bathrobe, watch the skies for flying saucers, start worrying when all the dolphins on the planet vanish, and keep your eyes peeled for an electronic gizmo with the words "Don't Panic" printed in large friendly letters on the cover.

--The Hitchhiker's Guide to the Galaxy

Problem

Addition,subtration and multiplication is necessary wherever you are in the galaxy.You are to write a program to perform these operations, and you can see 6*7=42 or 67-25=42 or 31+11=42, maybe one of them is the problem of Life, the Universe and Everything.

Please note that the solution may only be submitted in Brainfk, Whitespace and Intercal, other languages like C/C++/Pascal/Java are not allowed!** If you want to use other languages to solve this problem, you may try this one.

Input

Multiple test cases, the number of them T is given in the very first line, T<=99.

Each test case contains one line which has the following form:

num **oper** *num*

where *num* is an integer number between 1 and 99, and **oper** is '+' or '-' or '*' (without quotes).

There's no extra whitespace and leading zero in the input.

Output

For each test case you should output one line which contains the answer without any leading zeros.You may assume this number is always a positive one.

Example

Input:

```
3
6*7
67-25
31+11
```

Output:

```
42
42
42
```

Added by:	Blue Mary
Date:	2007-04-01
Time limit:	17s
Source limit:	17000B
Languages:	WSPC BF ICK
Resource:	Folklore, description and test data by Blue Mary

SPOJ Problem Set (classical)

1461. Greedy Hydra

Problem code: DRAGON

Hydra is some very greedy animal. A hydra has 9 heads when he is born, and many more new heads will come out when he grows up. Of course, some old heads will break off because of caducidy.

One day, a hydra with M heads finds a tree with N fruits on it. He is very delighted and wants to eat this tree instantly. Since he has M heads, he must divide these N fruit into M groups, each group contains at least 1 fruit, and each head will eat a group of fruits.

The biggest head among the M heads is named "Boss", it must eat neither more nor less than K fruits, and, in the nature of things, the biggest fruit included. These fruits are connected by N-1 branches, and there exists a path made up with branches between each pair of fruit.

If two fruit connected by a single branch is put in different groups, the corresponding two heads will break the branch and eat the two fruits, otherwise the corresponding head will eat the two fruits without breaking the branch. Eating branches is not very comfortable of course, so every branch has a weight of illness, and the weight of illness of this hydra is the sum of the weights of illness of all brances he has eaten.

Your task is to help the hydra to minimize his weight of illness.

The picture below is an example.

[IMAGE]

N=8,M=2,K=4.The bigger head eats 4 fruits(full points), the smaller head eats 4 fruits(empty points). The branch signed by a thin segment is eaten by the hydra.

Input

Ten test cases(Given one after another, you have to process all!). For each test case the first line contains 3 integers N(1<=N<=300),M(2<=M<=N),K(1<=K<=N), separated by single spaces. The N fruits are numbered 1..N, and the biggest fruit is always numbered 1. N-1 lines follow, each contains 3 integers i,j,k separated by spaces denoted that there is a branch between fruit i(1<=i<=N) and fruit j(1<=j<=N) and the weight of illness of this branch is k(0<=k<=100000).

Output

Ten lines, each contains a single integer - the minimum weight of illness of the hydra. If we can’t divide the fruit into M groups, output "-1"(without quotes).

Example

Input:

```
8 2 4
1 2 20
1 3 4
1 4 13
2 5 10
2 6 12
3 7 15
3 8 5
[and 9 test cases more]
```

Output:

```
4
[and 9 test cases more]
```

Link

After solving this problem you can try the problem DRAGON2.

Blue Mary's Note: test data has been modified on Dec.12, 2007. All the solutions have been rejudged.

Added by: Blue Mary
Date: 2007-04-01
Time limit: 9s
Source limit:50000B
Languages: All except: C99 strict
Resource: Chinese National Olympiad in Informatics 2002,Day 1; translated by lcosvse

SPOJ Problem Set (classical)

1462. Barbarians

Problem code: BARB

There are N Barbarians living on an unknown island. On the island there are M caves, we can number them 1, 2, ..., M clockwise. When we find the island, the barbarians are living in N distinct caves numbered C1,C2,...,CN.Every year each barbarian walks out of his cave and goes along the road,passes Pi caves and then go into that cave.Every Barbarian has a living time: Li years, after Li years the ith barbarian died.

We are surprised to find out that no two barbarians live in one cave in the same year so no conflicts have happened.We are interesting about the minimum number of caves on the island.

Please note that this problem has a somewhat strict source limit and time limit.

Input

The very first line contains a single integer T,the number of test cases.T blocks follow.

For each test case, the first line contains a single integer N(N<=15).N lines follow,each contains 3 integers Ci(1<=Ci<=100),Pi(1<=Pi<=100),Li (1<=Li<=1,000,000).

Output

For each test case, the first and only line contains a single integer M - the answer.You may assume M<=1,000,000.

Example

Input:

```
1
3
1 3 4
2 7 3
3 2 1
```

Output:

```
6
```

Hints

Year	Barb. No.1	Barb. No. 2	Barb. No. 3
0	1	2	3
1	4	3	5
2	1	4	Died

```
|  3   |     4      |      5      |    Died     |
-----------------------------------------------
|  4   |     1      |    Died     |    Died     |
-----------------------------------------------
```

Added by: Blue Mary
Date: 2007-04-01
Time limit: 5s
Source limit:2048B
Languages: All except: C99 strict
Resource: Chinese National Olympiad in Informatics 2002,Day 2; translated by Blue Mary

SPOJ Problem Set (classical)

1463. Robot Number M

Problem code: ROBOT

Blue Mary, the well-known astronaut, had sent Robot No.1 to Mars finally.Robot No.1 was so smart that he could make one robot per second.

Assume the time Robot No.1 arrived was second 1. At second i, Robot No.1 made a new Robot: Robot No.i. (i>=2)

The new robots started work as soon as he was produced successfully. Robot No.M only had a rest at second t, where t is a multiple of M. For example, Robot No.3 worked at second 4,5,7,8,... and had a rest at second 6,9,...

When a robot was having a rest, he could send his own informations to the newly produced robot. For example, when Robot No.6 was produced successfully, Robot No.2 and Robot No.3 are having a rest, so Robot No.6 would get informations from Robot No.2 and No.3. We call Robot No.2 and No.3 are teachers of Robot No.6.

We call two Robots are independent if each of them wasn't a teacher of the other one and they had no common teachers. **Please note that Robot No.1 was independent to any other robots and wasn't a teacher of any other robots,** since only Robot No.1 could make robots.

The good number of Robot No.M is the number of robots who was produced earlier than No.M and independent to No.M. Here are some examples:

The good number of Robot No.1 is 0.

The good number of Robot No.2 is 1. No.1 was that robot.

The good number of Robot No.6 is 2. No.1 and No.5 were those robots. No.2 and No.3 were his teachers. No.4 and him had a common teacher: No.2.

The Robots had 3 kinds of occupations. To Robot No.M:

- If M can be written as the multiple of an even number of distinct odd primes, he was a businessman.
- If M can be written as the multiple of an odd number(1 included) of distinct odd primes, he was a hacker.
- All other robots were doctors.

Now Blue Mary was interesting to Robot No.M. She wants to know the sum of good numbers of all businessmen, hackers and doctors among Robot No.M and his teachers. She comes up with the answer quickly, and so can you.

Blue Mary is so kind that she gives you the prime divisors of M and you can only tell her the answers modudo 10000.

Input

The very first line contains a single integer T,the number of test cases.T blocks follow.

For each block, the first line contains a single integer K.K lines follow, each contains two integers p_i and a_i separated by a single space.

$M = p_1^{a_1} * p_2^{a_2} * p_3^{a_3} * ... * p_K^{a_K}$.

You can assume that:

- All p_i are distinct primes and are less than 10,000.
- $p_1 < p_2 < p_3 < ...$

  n.
- All a_i are positive and no more than 1,000,000.
- 1 <= k <= 1000.

Output

For each test case, output 3 lines.

The first line contains a single integer denotes to the sum of good numbers of all businessmen among Robot No.M and his teachers modudo 10000.

The second line contains a single integer denotes to the sum of good numbers of all hackers among Robot No.M and his teachers modudo 10000.

The third line contains a single integer denotes to the sum of good numbers of all doctors among Robot No.M and his teachers modudo 10000.

Example

Input:

```
1
3
2 1
3 2
5 1
```

Output:

```
8
6
75
```

Hints

In the sample input, $M=2^1*3^2*5^1=90$. Robot No.90 has 10 teachers.Among Robot No.90 and his teachers, Robot No.15 is a businessman; No.3 and No.5 are hackers; all others are doctors, their numbers are 2,6,9,10,18,30,45,90.

Added by: Blue Mary
Date: 2007-04-01
Time limit: 2s
Source limit: 50000B
Languages: All except: C99 strict
Resource: Chinese National Olympiad in Informatics 2002,Day 2; translated by Blue Mary

SPOJ Problem Set (classical)

1464. Editor II

Problem code: EDIT3

Background

After trying to solve problem EDIT1(Editor) and being ****ed by Brainf**k, Blue Mary decided to set another difficult problem about editor.

Description

Some definations:

- Text: It's a sequence that consists characters whose ASCII code is in [32,126].
- Cursor: It's a sign for pointing out the current position.It can be at the start or the end of the text or between two consecutive characters of the text.

Editor is a structure.It contains one text and one cursor.The operations are listed below:

Name	Input format	function
Move(k)	Move k	Move the cursor after the kth character in the text. If k=0, you should put the cursor at the start of the text.
Insert(n,s)	Insert n s	Insert string s whose length is n(>=1) after the cursor.The cursor doesn't move.
Delete(n)	Delete n	Delete n(>=1) characters after the cursor. The cursor doesn't move.
Get(n)	Get n	Output n(>=1) characters after the cursor.
Prev()	Prev	Move the cursor one character forward.
Next()	Next	Move the cursor one character backward.

If the text of a editor is empty,we say the editor is empty.

Here is an example._ denotes to the cursor,$ denotes to the start and the end.At start the editor is empty.

Operation	Text after the operation	Output
INSERT(13,"Balanced tree")	$_Balanced tree$	$$
MOVE(2)	$Ba_lanced tree$	$$
DELETE(5)	$Ba_d tree$	$$

NEXT()	$Bad_ tree$	$$	
INSERT(7," editor")	$Bad_ editor tree$	$$	
MOVE(0)	$_Bad editor tree$	$$	
GET(15)	$_Bad editor tree$	$Bad editor tree$	

Your task is:

- Build an empty editor.
- Read some operations from the standard input and operate them.
- For each Get operation, write the answer to the output.

Input

the very first line contains the number of testcases T(T<=4).T tests follow.

For each test, the first line is the number of operations N.N operations follow.

Blue Mary is so depressed with the problem EDIT1 that she decides to make the problem more difficult.So she inserts many extra line breaks in the string of the Insert operation.You must ignore them.

Except line breaks, all the charaters' ASCII code are in [32,126]. There's no extra space at the end of a line.

You can assume that for each test case:

- No invalid operation is in the input.
- Number of move operations is no more than 50000.
- Number of the total of insert and delete operations is no more than 4000.
- Number of the total of prev and next operations is no more than 200000.
- The characters inserted will not more than 2MB.The valid output will not more than 3MB.

Output

The output should contain T blocks corresponding to each testcase.

For each test case, the output should contain as many lines as the get operations in the input.Each line should contains the output of each get operation.

Example

Input:

```
1
15
Insert 26
abcdefghijklmnop
qrstuv wxy
Move 15
Delete 11
```

```
Move 5
Insert 1
^
Next
Insert 1
_
Next
Next
Insert 4
.\/.
Get 4
Prev
Insert 1
^
Move 0
Get 22
```

Output:

```
.\/.
abcde^_^f.\/.ghijklmno
```

Warning: large Input/Output data, be careful with certain languages

Blue Mary's note: the test case #1 has something wrong and it has been fixed on April 27th, 2007.Solutions has been rejudged. Please accept my apology.

Added by: Blue Mary
Date: 2007-04-01
Time limit: 3s
Source limit:50000B
Languages: All except: C99 strict
Resource: Chinese National Olympiad in Informatics 2003,Day 1

SPOJ Problem Set (classical)

1465. On the Way to Find Chris

Problem code: CHRIS

Do you know the famous game THE KING OF FIGHTERS? If the answer is yes, I'm sure you know THE THREE BLACK GROUP: Chris, Shermie and Yashiro. Today Chris is invited to one of his friends' home to play THE KING OF FIGHTERS.Blue Mary is now at Chris' home, she knows where Shermie's and Yashiro's home is, but she doesn't know where Chris actually is.So she decides that:

- If Yashiro's home is nearer than Shermie's, she will go to Yashiro's home first, if she doesn't find Chris, she will then go to Shermie's, and vice versa.
- The map of the city is strange. Each of the houses is assigned to a unique number in the range[1,n], where n is the number of houses. Between some pairs of houses there are some roads. There exists one and only one path from any house to any other house. She will go along the only path between the two houses.

Unfortunately, you don't know where Chris' home actually is, and the same as Yashiro's and Shermie's. Now you are interesting in the maximum time from the time when Blue Mary starts from Chris' home to the time when Blue Mary finds Chris in the worst case.

Input

The number of test cases T is given in the very first line.T blocks follow.

For each test case, the first line contains 2 space-separated integers N(3<=N<=200000) and M, which denotes the number of houses and the number of roads in the city.M lines follow, each contains 3 space-separated integers x,y,z(1<=x,y<=n,1<=z<=10^9).It tells us there exist a road between house No.x and house No.y, and to go from x to y or from y to x along this road will take z minutes.No two roads are repeated.

Output

For each test case you should output a single line, which contains a single integer - the maximum time measured in minutes.

Example

Input:

```
1
4 3
1 2 1
2 3 1
3 4 1
```

Output

```
4
```

Warning: large Input/Output data, be careful with certain languages

---

Added by: Blue Mary
Date: 2007-04-01
Time limit: 5s
Source limit:50000B
Languages: All except: C99 strict
Resource: Chinese National Olympiad in Informatics 2003,Day 2; translated by Blue Mary

SPOJ Problem Set (classical)

1466. Blue Mary Needs Help Again

Problem code: CASHIER

Blue Mary is a cashier of a big company.The boss of this company is so annoying that he always increases or decreases wage of all workers.He increases all the workers' wage with a same number when he is happy or decreases all the worker's wage with a same number when he is depressed.

All the workers are angry with the boss, especially when he decreases their wage. A worker will leave the company and never go back when he finds his wage is lower than the least wage written on his contract. Blue Mary must delete the worker's files at that time. Another task she is to do is to build a file when a new worker joins this company.

The boss usually asks Blue Mary how much money the worker who gets the k-th most wage gets. Blue Mary is very tired with her work. Could you give her a hand?

Input

T

[the number of tests <= 10]

M MIN

[M is the number of commands below, MIN is the least wage]

C K

[C is one of the 4 characters I,A,S,F. I denotes that Mary should build a new file, and the new worker's wage is K(0<=K<=100000) at start.If K is less than MIN, the worker will not join the company. A denotes that the boss increases all the workers' wage with K(0<=K<=1000). S denotes that the boss decreases all the workers' wage with K(0<=K<=1000). F denotes that the boss asks Blue Mary a question: how much money does the worker who gets the k-th most wage get(K>0)?]

[M-1 other commands]

[other tests]

You may assume that:

- The number of worker in the company at start is 0.
- The number of command I is no more than 100000.
- The total numner of command A and S is no more than 100.
- The number of command F is no more than 100000.

Output

For each test case:

For each F command you must output one line contains a single integer which is the answer or -1 if K is more than the number of workers in the company at that time.

In the last line you must output a single integer - the number of workers who leave the company(excluded the ones who don't join the company)

Example

Sample Input:

```
1
9 10
I 60
I 70
S 50
F 2
I 30
S 15
A 5
F 1
F 2
```

Sample Output:

```
10
20
-1
2
```

Warning: large Input/Output data, be careful with certain languages

---

Added by: Blue Mary
Date: 2007-04-01
Time limit: 7s
Source limit:50000B
Languages: All except: C99 strict
Resource: Chinese National Olympiad in Informatics 2004,Day 1; translated by Blue Mary

SPOJ Problem Set (classical)

1468. Outside it is now raining

Problem code: RAIN2

Country M is pluvian. In City P, it rains frequently and many people complain that they always have trouble crossing the streets when raining. To make people cross the streets easier, the government set many "Automatic Umbrellas" above every crosswalk, shown in the picture below.

[IMAGE]

Each of these "Automatic Umbrellas" looks like a rectangle board, and their thickless is approximate zero. They can sop up the rain instantly. They are left unused when it’s not raining and shuttle from one side to another in the same speed otherwise. The walkers will not be wringing-wet if he walks under the umbrella when it’s raining.

When many people want to cross the street, one "Automatic Umbrella" is not enough obviously. The government set many "Automatic Umbrella" on some main crosswalks. The width of each of the "Automatic Umbrella" equals to the width of the crosswalk, and any two of these umbrellas have different height.Their length and speed may be different.

You are to write a program to calculate the total volume of the rain falling to the ground from the time when it starts to rain to T seconds later.

Input

The very first line comes a single integer Q, the number of test cases.Q blocks follow.

For each test case:

The first line contains 4 space-separated integers N(<=10), W(<=100), T(<=100), V(<=50), the number of "Automatic Umbrella", the length of the crosswalk in meters, the total time in seconds and the volume of rain falling to the ground per square meter per second.

To simplify the description, we can build a Cartesian coordinate system in the following way: let the left side of the street be the origin, the street be the positive Ox-axes, and the vertical line to the ground be the positive Oy-axes,see the picture below.

[IMAGE]

Each of the next N lines contains 3 integers xi, li, vi, the initial position(in meter), the length(in meter), the speed(in meter per second) of the i-th umbrella. If vi>0, the umbrella moves to the right side initially; if vi<0, the umbrella moves to the left side initially;if vi=0, the umbrella doesn’t move at all.

You can assume that the width of the umbrella and the crosswalk is 1 meter, the rain falls vertically, the speed of the rain will not change and the umbrellas and the crosswalk are absolutely horizontal.

Output

For each test case, you should output a single real number(rounded to 2 decimal places) - the answer.You can assume the total distance of all the umbrella's movement will not exceed 550*W.

Example

Input:

```
1
2 4 3 10
0 1 1
3 1 -1
```

Output:

```
65.00
```

Added by: Blue Mary
Date: 2007-04-01
Time limit: 7s
Source limit:50000B
Languages: All except: C99 strict
Resource: Chinese National Olympiad in Informatics 2004,Day 2; translated by Blue Mary

SPOJ Problem Set (classical)

1470. Another Sequence Problem

Problem code: SEQ2

You are to write a program to perform some operations on a given sequence.These operations are listed below:

Name	Input format	function
Modify	MAKE-SAME i t c	Modify all the t numbers from the ith number(included) to number c.
Insert	INSERT i t s	Insert t numbers after the ith number.s is a sequence of t numbers which should be inserted one-to-one.If i=0,you should insert s in the first of the sequence.
Delete	DELETE i t	Delete t numbers after the ith number(included).
Reverse	REVERSE i t	Reverse t numbers after the ith number(included).
Get sum	GET-SUM i t	Output the sum of t numbers after the ith number(included).
Get maximum partial sum	MAX-SUM	Output the maximum partial sum in the sequence now.

See the example.

Input

The very first line contains a single integer T(T<=4), the number of test cases.T blocks follow.

For each test case:

The first line contains two integers n and m(m<=20000), the number of numbers in the sequence in the beginning and the number of operations.

The second line contains n integers seperated by spaces, the sequence in the beginning.

Next m lines, each contains an operation listed above.

You can assume that for each test case:

- No invalid operation is in the input.
- Number of numbers in the sequence is no more than 500000 and not less than 1 at any time.
- All the numbers in the sequence is in range[-1000,1000] at any time.
- The total number of numbers inserted will be not more than 4,000,000.The input is no more than 20MB.

Output

For each Get sum and Get maximum partial sum operation,you should write the answer to the output,one per line.

Example

Input:

```
1
9 8
2 -6 3 5 1 -5 -3 6 3
GET-SUM 5 4
MAX-SUM
INSERT 8 3 -5 7 2
DELETE 12 1
MAKE-SAME 3 3 2
REVERSE 3 6
GET-SUM 5 4
MAX-SUM
```

Output:

```
-1
10
1
10
```

Hints:

```
After the 3rd op., the sequence is
2 -6 3 5 1 -5 -3 6 -5 7 2 3


After the 4th op., the sequence is
2 -6 3 5 1 -5 -3 6 -5 7 2


After the 5th op., the sequence is
2 -6 2 2 2 -5 -3 6 -5 7 2


After the 6th op., the sequence is
2 -6 6 -3 -5 2 2 2 -5 7 2
```

Warning: enormous Input/Output data, be careful with certain languages

---

Added by: Blue Mary
Date: 2007-04-01
Time limit: 7s
Source limit:50000B
Languages: All except: C99 strict
Resource: Chinese National Olympiad in Informatics 2005,Day 1; translated by lcosvse

SPOJ Problem Set (classical)

1471. A Game of Pearls

Problem code: PRLGAME

The game of pearls is to use the following 12 kinds of pearl designs each one and only one time to make the big pearl design.The pearl designs can be rotated and turned over arbitrarily. See the pictures below.

The empty grids

[IMAGE]

A sample big pearl design

[IMAGE]

A sample big pearl design used characters instead of colors

[IMAGE]

Part A

[IMAGE]

Part B

[IMAGE]

Part C

[IMAGE]

Part D

[IMAGE]

Part E

[IMAGE]

Part F

[IMAGE]

Part G

[IMAGE]

Part H

[IMAGE]

Part I

[IMAGE]

Part J

[IMAGE]

Part K

[IMAGE]

Part L

[IMAGE]

Input

Ten test cases(given one after another,you have to process all!), Each contains a big design, 'A'-'L' denote the filled grids, '.' denotes the empty grids,see the example. You can assume that the pearl designs used are completely put into the empty grids.

Output

A big design which has no grids that haven't been filled and each pearl design is used one and only one time in it,or 'No solution'(without quotes) if there's no solution.If there are multiple solutions,output any.

Example

Input:

```
.
..
...
....
.....
.....C
...CCC.
EEEHH...
E.HHH....
E.........
[and 9 test cases more]
```

Output:

```
B
BK
BKK
BJKK
JJJDD
GJGDDC
GGGCCCI
EEEHHIIA
ELHHHIAAF
ELLLLIFFFF
[and 9 test cases more]
```

Added by: Blue Mary
Date: 2007-04-01
Time limit: 20s
Source limit:50000B
Languages: All except: C99 strict
Resource: Chinese National Olympiad in Informatics 2005,Day 1; translated by Blue Mary

SPOJ Problem Set (classical)

1472. Tom and Jerry

Problem code: TOMJERRY

You may know the great American cartoon called *Tom and Jerry*. In that cartoon, Tom is a stupid cat, and Jerry is a cute mouse. They are all living in Blue Mary's house. (Maybe the name of the owner of the house they living in is not Blue Mary, but this is not important. ^_^)

One day Jerry gets a GPS, so after that he can detect Tom's position accurately. He decides to play tricks on Tom again. The house has N (1<= N <= 1000) rooms numbered from 1 to n and E(1<= E <=1000) corridors, each connects two different rooms, and there is at most one corridor between two rooms. When Jerry gets the GPS, he is in room No.P(1<= P <=N) while Tom is in room No.Q(1<= Q <= N). You may assume that room No.P and room No.Q is in one connected componenet and they are not the same room. Since Jerry is very cute and Tom is very stupid, if they are in the same room, Jerry can befool Tom as soon as possible. Now Jerry wants to get to Tom's room as soon as possible. At each time unit, he will detect Tom's position, and choose the room next to his room which is the nearest to Tom's position and go to that room. If there are several rooms satisfied the condition above, he will choose the one with the least room number. After that, if they are in the same room, Jerry will stay and play tricks on Tom, or he will repeat this progress one more time otherwise.

Now poor Tom doesn't know he'll be played, he is taking a walk in the house leisurely. Each time unit he will choose a room next to the room he is in and go to that room or stay in the room he is in now, with equal probability. For example, if Tom is now in room 1, room 2 and room 3 are the only neighbors of room 1 (i.e. there is a corridor between room 1 and room 2, and there is a corridor between room 1 and room 3), at the next time unit, the probability of that Tom is in room 1,2 or 3 are all 1/3.

Suppose at each time unit, Jerry moves first, and Tom will move after Jerry complete his move. You task is to calculate the expected time unit from the time when Jerry gets the GPS to the time when Jerry and Tom are in the same room.

Input

Multiple test cases, the number of them is given in the very first line.

For each test case:

The first line contains two space-separated integers N and E. The second line contains two space-separated integers P and Q. E lines then follow, each contains 2 space-separated integers A and B which shows that there is a corridor between room A and room B.

Output

For each test case:

Output one line, which contains a single real number - the expected time unit, rounded to 3 decimal places.

Example

Input:

```
2
4 3
1 4
1 2
2 3
3 4
9 9
9 3
1 2
2 3
3 4
4 5
3 6
4 6
4 7
7 8
8 9
```

Output:

```
1.500
2.167
```

Hint

The calculations for the first example can be found here.

Added by: Blue Mary
Date: 2007-04-01
Time limit: 6s
Source limit: 50000B
Languages: All except: C99 strict
Resource: Chinese National Olympiad in Informatics 2005,Day 2; translated by Blue Mary

SPOJ Problem Set (classical)

1473. Lemon Tree in the Moonlight

Problem code: LEMON

Blue Mary extremely like lemon trees. When the softness moon shines the ground, she lies under the lemon tree, thinking about the true meaning of life, the universe and everything quietly.

Before long, she is shadowed by the lemon tree. Blue Mary is such a cute girl, that she soon comes up with a question: what is the area of the shadow?

She knows it's hard for her to measure the shadow directly. So that, she tries to calculate it in geometrical way.

As Mary knows well about this beautiful lemon tree, she regards it as N frustums of cones,each frustums of cone is defined as a floor, numbered 1..N from the bottom to top. Of course, the Nth(Top) is a cone. Each frustum has two circle surfaces. Each two adjacent frustums shares a same circle surface. All the centres of the circle surfaces are on a plumb line. Mary knows that the height of every frustums is h1,h2,..,hn, and the undersurface of the 1st frustum has a height h0 from the ground.

Mary measures that the included angle of the moonshine and the ground is Alpha, which is an acute angle.

[IMAGE] [IMAGE]

For briefness, we suppose the moonshine is parallel and the ground is aclinic.

And we ignore the bole of the tree. Mary comes up with the answer quickly, and she wants your ideas all the same.

Input

The very first line of the input data contains one integer T, the number of tests. T blocks follow.

For each test:

The first line of the input data contains one integer number N(N<=500) and a real number Alpha (Alpha>0.3).

N denote the number of floors, Alpha denote the included angle of the moonshine and the ground(radian).

The second line contains N + 1 real number h0 h1 h2... hn.(hi<=100) h0 denotes the height of the undersurface of the 1st frustum. h1 .. hn denote the height of each floor.

The third line contains N real number r1 r2 .. rn(ri<=100), the radii of the undersurface in each floor.

All the data in each line is seperated by spaces.

Output

For each data set you should output one line containing a single real number - the area of the shadow. Numbers should be rounded to two decimal places.

Example

Input:

```
1
2 0.7853981633
10.0 10.00 10.00
4.00 5.00
```

Output:

```
171.97
```

Time limit has been changed from 30 seconds to 13 seconds, some naive solution gets TLE.

Added by: Blue Mary
Date: 2007-04-01
Time limit: 13s
Source limit:50000B
Languages: All except: C99 strict
Resource: Chinese National Olympiad in Informatics 2005,Day 2; translated by g201513

SPOJ Problem Set (classical)

1475. VII - Act IV

Problem code: WORMS

Background

This problem is completely unrelated to its problem code ^_^.

Description

A natural number x is called a good number if one or two of the next two conditions is satisfied:

- 7 is a divisor of x.
- 7 is a digit of x.

Task

Write a program that:

- reads a number n from the standard input,
- computes the number of good numbers in range$[1,10^n]$,
- writes the result to the standard output.

Solve the problem in at most 0.5kB of source code.

Input

The input begins with an integer t(t<= 210), the number of test cases.t test cases follow.

For each test case, the first and only line contains an integer n (1<=n<=500).

Output

For each test case the output consists of one line that contains the answer.

Example

Sample input:

```
3
1
2
1
```

Sample output:

```
1
30
1
```

Added by:	Blue Mary
Date:	2007-04-01
Time limit:	2s
Source limit:	512B
Languages:	All except: C99 strict
Resource:	Folklore, description, standard program and test data by Blue Mary

SPOJ Problem Set (classical)

1476. Maximum Profit

Problem code: PROFIT

CS&T, the well-known cellphone company, is going to set some new service stations among n possible ones, which are numbered 1,2,...,n. The costs of setting these stations are known as P1,P2,..,Pn. Also the company has made a survey among the cellphone users, and now they know that there are m user groups numbered 1,2,...,m, which will communicate by service station Ai and Bi, and the company can profit Ci.

Now CS&T wants to know which service stations are to be set that the company will profit most.

Input

```
T [The number of tests]
n m [n<=5000 m<=50000]
P1 P2 P3 ... Pn [Pi<=100]
A1 B1 C1
A2 B2 C2
...
Am Bm Cm [1<=Ai,Bi<=n, Ci<=100]
[other tests]

At least 80% of the tests satisfy that n<=200, m<=1000.
```

Output

```
MaximumProfit
[other tests]
```

Example

Input:

```
1
5 5
1 2 3 4 5
1 2 3
2 3 4
1 3 3
1 4 2
4 5 3
```

Output:

```
4
```

Hints:

```
The service stations to be set are 1,2,3.
```

Added by: Blue Mary
Date: 2007-04-01
Time limit: 7s
Source limit:50000B
Languages: All except: C99 strict
Resource: Chinese National Olympiad in Informatics 2006,Day 2; translated by Blue Mary

SPOJ Problem Set (classical)

1477. Play with a Tree

Problem code: PT07A

Hey, ACRush and Jelly are playing a game ! Let take a look at its rule:

You are given a tree. Two players take turns cutting edges on a tree. Some nodes is on the "ground". When a player cuts an edge, all the edges that are no longer connected to the ground disappear. The player who can not take a move loses.

ACRush plays first. Both of them are very good players. If you know state of the tree they are playing with, can you guess who will win?

[IMAGE] Node 4 is on the ground.

Input

Input consists of multiple test-cases. The first line contains one integer *t* - number of cases (0 < *t* <= 20). For each case, the input format is following. The first line contains one integer *N* (1 <= *N* <= 100000). The next line *N* integers s[i] (1 or 0). If s[i] is 1, the *i*-th node is on the ground. If s[i] is 0, the *i*-th node is not on the ground. Each line of the following *N* - 1 lines contains two integers u, v. They denote there is an edge between node u and node v (1 <= u,v <= N).
There is no blank line after each case.

Output

For each case, output who will win the game. If ACRush wins, output *1*; otherwise, output *0* (Jelly wins).
There is no blank line after each case.

Example

Input:
```
1
4
0 0 0 1
1 2
2 3
2 4
```

Output:
```
1
```

Added by: Thanh-Vy Hua
Date: 2007-04-07
Time limit: 0.5s-1s
Source limit:50000B
Languages: All
Resource: Co-author Amber

SPOJ Problem Set (classical)

1478. The Easiest Problem

Problem code: PT07B

You are given an unweighted, undirected tree *T*. We say *T* is special iff it has this property:

"All nodes of degree greater than or equal to 3 are surrounded by at most two nodes of degree two or greater."

Finding maximal size subtree of this tree so that it's a special tree.

Input

The first line of the input file contains one integer *N* --- number of nodes in the tree ($0 < N <= 10^6$). Next *N*-1 lines contain *N*-1 edges of that tree --- Each line contains a pair (*u*, *v*) means there is an edge between node *u* and node *v* ($1 <= u, v <= N$).

Output

At the first line, output number of nodes in the optimal subtree you found. Next lines, print all edges belong to that subtree, each line contains a pair *u v* means an edge between node *u* and node *v*.

Example

Input:

```
5
1 2
2 3
2 4
2 5
```

Output:

```
5
1 2
2 3
2 4
2 5
```

Added by: Thanh-Vy Hua
Date: 2007-04-07
Time limit: 0.100s-10s
Source limit:50000B
Languages: All
Resource: Co-author Amber

SPOJ Problem Set (classical)

1479. The GbAaY Kingdom

Problem code: PT07C

Jiajia is the king of the GbAaY Kingdom. He always squeezes his 20 ministers as coolies. There are *n* cities and *m* two-way roads connecting cities in the kingdom. Because of the increasing of the oil fee, he want to simplify the road system in the GbAaY Kingdom to save the traffic cost. Thus, some of roads will be removed. But he requests the ministers guarantee that there is always a path between any two cities. GbAaY Minister Loner suggests Jiajia for the convenience of the traffic management, the farthest distance between cities should be minimal. Unhesitatingly, Jiajia agrees this resolution. As the GbAaY Kingdom's minister (cooly), you must work hard for Jiajia to make the simplification plan.

Input

The first line contains two integers *n*, *m* (1 <= *n* <= 200, *n* - 1 <= *m* <= 20000). Each line of the following *m* lines contains three integers *u*, *v*, *w* (*u* != *v*, 0 <= *w* <= 10^5). They denote there is a road with length *w* between city *u* and city *v*.

Output

The first line contains one integer which is the farthest distance between cities after the simplification. Each line of the follow *n* - 1 contains two integers *u*, *v* (*u* < *v*). They denote there is an road between city *u* and city *v* in the simplification plan. If there are many optimal solutions, any of them will be accepted.

Example

Input:

```
3 3
1 2 1
2 3 1
1 3 1
```

Output:

```
2
1 2
1 3
```

Added by: Thanh-Vy Hua
Date: 2007-04-07
Time limit: 1s
Source limit:50000B
Languages: All
Resource: Co-author Amber

SPOJ Problem Set (classical)

1480. Let us count 1 2 3

Problem code: PT07D

Given two integer n, p. 4 kinds of query is needed to solve:

1. Counting the number of labeled unrooted trees with n nodes
2. Counting the number of labeled rooted trees with n nodes
3. Counting the number of unlabeled rooted trees with n nodes
4. Counting the number of unlabeled unrooted trees with n nodes

Calculate the answer modulo p.

Input

Each line contains three integers k, n, p. k denotes which kind of query this case is.
For Kind 1 or Kind 2 query, $1 <= n <= 10^9$.
For Kind 3 or Kind 4 query, $1 <= n <= 10^3$ and $n <= p$.
For all queries, $2 <= p <= 10^4$ and p is a prime.

Output

For each query, output a line which contains only one integer.

Example

Input:
```
1 2 2
2 2 3
3 2 5
4 2 3
```

Output:
```
1
2
1
1
```

Added by: Thanh-Vy Hua
Date: 2007-04-07
Time limit: 0.300s-4s
Source limit: 50000B
Languages: All
Resource: Co-author Amber

SPOJ Problem Set (classical)

1482. A short vacation in Disneyland

Problem code: PT07F

After a lot of exams at school, Amber and his friends Ahyangyi, Dragon have a short vacation in Hong Kong Disneyland. Many interesting places they want to visit there: resort, castle,... . In each place and between them, there are special bidirectional rails, so that the visitors can drive a small special car, go around and have a sightseeing tour. This rail system is quite optimal it has tree shape ! Each time, you start a new route (or you can call it "path") with a car, you must purchase a new ticket.

Amber and his friends surely want to visit all places, and each place exactly once, so bored to visit one place many times. But the trouble is they don't carry much money. So Amber thinks about a good way to purchase as small number of tickets as possible (i.e. minimal number of routes). We don't care how they can switch cars during their trip.

Now you're given maps of the Disneyland, please help them to find an optimal solution.

Take a look at the figure below:
[IMAGE]

There are many optimal solutions here and Amber must purchase at least 3 tickets, for 3 disjoint routes. Two possible solutions are:
Solution 1:
1-st route: they visit 1 2 3
2-nd route: they visit 4
3-rd route: they visit 5 6 7
Solution 2:
1-st route: they visit 1 2 4 6 5
2-nd route: they visit 3
3-rd route: they visit 7

Input

There may be many maps in one input file. The first line of file is number of maps *T* (0 < T <= 10). The following line is blank. Then, there are the descriptions of *T* maps.

For each map, the first line contains one integer *N* --- number of places in the Disneyland (0 < N <= 10000). We number places from 1 to N. Next N-1 lines contain N-1 rails between places --- Each line contains a pair (u, v) means there is a rail between place u and place v (1 <= u,v <= N).
There is a blank line after each description.

Output

For each map, the first line, write minimal number of routes *K*. Next *K* lines, show out the routes in your solution, each has form u[1] u[2]...u[m], means the route starts at place u[1] then visits place u[2],..., ends at place u[m]. So between 2 consecutive places in each route must have a rail. If there are many solutions, any of them will be accepted.

There is no blank line after each case.

Example

Input:

```
1

7
1 2
2 3
2 4
4 6
5 6
6 7
```

Output:

```
3
1 2 3
4
5 6 7
```

Added by: Thanh-Vy Hua
Date: 2007-04-07
Time limit: 0.100s-0.5s
Source limit: 50000B
Languages: All
Resource: Co-author Amber

SPOJ Problem Set (classical)

1483. Colorful Lights Party

Problem code: PT07G

ACRush and his friends want to open a party to celebrate the good result of THU in ICPC 2007. They will use all halls in THU for their party. There are 2 kinds of hall: the small and the large one. In each hall, there is an electronic light system, which forms a tree topology to reduce redundant wires.

- In a small hall, the light system is a general tree with n lights. The lights are numbered from 1 to *n*
- In a large hall, the light system is obtained from *k* chains of lights, each chain has length *t*. The first lights of these k chains are connected with a big light at the central stage of the hall. The big light has id 1, the first light of each chain has id from 2 to k+1, then we continue with the second light of each chain, so on...

Take a look at the figure below:
[IMAGE]

ACRush hopes in every hall, each light has an unique color and so do the wires!

For easier to remember and to hang up lights against the walls, he sets a rule:

- For each hall, we number the color from 0 to *n-1*, so each light will get a color id in set {0, 1, ..., *n-1*}.
- Color id of the wire connects i-th light and j-th light uniquely identified by the positive difference between color ids of i-th light and j-th light.

At first view, the rule seems easy, so everyone agrees with him. But it's really tough if the room is quite large, too hard to set colors for lights. After few seconds, ACRush says *"So in this hall, the 1-st light should have color 3, the 2-nd one should have color 0,..."*. Well, how can he do it very fast?

How about you ? Let write a program to help ACRush's friends setting colors for lights in all *T* halls.

Input

The first line of file is *T* -- number of halls in THU (0 < T <= 10). The following line is blank. Then, there are the descriptions of T halls.

For each hall, the first line contains one integer *kind*. *kind* denotes which kind of the current hall: 1 is a small hall, 2 is a large one. On next line, there are two cases:

- For Kind 1, first line is n (1 <= n <= 27) -- number of lights. Next *n - 1* lines describe wires in this hall. Each line is pair (u, v) -- there is a wire between light u and v (1 <= u, v <= n).
- For Kind 2, only one line contains two numbers k and t (1 <= k, t <= 1000).

There is a blank line after each description.

Output

For each hall, show us *n* numbers on one line, i-th number is the color id of i-th light. If there are many solutions any of them will be accepted. Otherwise, if there is no solution, all color id should be *-1*.
The color ids on one line are separated by exactly one blank, and you'd better not print any redundant blanks. There is no blank line after each case.

Example

Input:

```
2

1
3
1 2
2 3

2
2 1
```

Output:

```
0 2 1
0 4 2 1 3
```

Added by: Thanh-Vy Hua
Date: 2007-04-07
Time limit: 0.100s-2s
Source limit: 50000B
Languages: All
Resource: Co-author Amber

SPOJ Problem Set (classical)

1484. Search in XML

Problem code: PT07H

The XML (eXtensible Markup Language) is gaining popularity as a new standard for data representation and exchange on the internet. XML provides a text-based means to describe and apply a tree-based structure to information. The XML document consists of nested elements, some of which usually have attributes and content. But for simplifying this problem, we needn't consider the attributes and content, i.e. only tags allowed. An element typically consists of two tags, a start tag and an end tag. The start tag consists of a name surrounded by angle brackets, like "<tag>"; the end tag consists of the same name surrounded by angle brackets, but with a slash preceding the name, like "</tag>". The element's content is empty or other sub-element (child) that appears between the start tag and the end tag. Specially, no XML element that has the same tag in its direct sub-elements (children), i.e. All sibling elements have different tag names. The following is an valid example for XML documents.

```
<THU>  <Team>  <ACRush></ACRush>  <Jelly></Jelly>  <Cooly></Cooly> </Team> <JiaJia>  <Team>  <Ahyangyi></Ahyangyi>  <Dragon></Dragon>  <Cooly><Amber></Amber></Cooly>  </Team> </JiaJia></THU>
```

For identifying the elements in a document, we number the elements in according to the order that the start tags of the elements appear in the document. For instances, "THU" is numbered 1. The first "Team" is numbered 2. "ACRush" is numbered 3. "Ahyangyi" is numbered 8.

The problem of querying XML documents has been given much attention by researchers. Now we are given a querying pattern of XML documents and a text of XML documents. The following is an valid example for pattern.

```
<Team><Cooly></Cooly></Team>
```

And we are requested to find all occurrences of the pattern in the text of XML documents. Here, the pattern occurs at a particular text position if placing the pattern with root element at that text position leads to a situation in which each pattern element overlaps some text element with the same label. Because the sibling elements have different labels, there is only one way to put the pattern into the text.

Input

There are two parts in the input file. The first part is a valid XML documents with exactly one root element. The second part is a valid XML documents as querying pattern with exactly one root element. Please ignore all whitespaces (unvisiable characters) in the input file, i.e. only consider the uppercase and lowercase letter and "/", "<", ">". Assume XML documents is always strictly a rooted tree. The input file is less than 100kb.

Output

Output all the occurrences of pattern in a text of XML documents. The first line consists of an integer *n* that denotes the number of the occurrences. Then the next *n* line, each line consists of an id number of an element that occurs the query pattern. Please print them in increasing order.

Example

Input:

```
<THU>    <Team>          <ACRush></ACRush>              <Jelly></Jelly>         <Cooly></Cooly> </Team> <JiaJia>              <Team>              <Ahyangyi></Ahyangyi>              <Dragon></Dragon>                    <Cooly><Amber></Amber></Cooly>          </Team> </JiaJia></THU>
<Team><Cooly></Cooly></Team>
```

Output:

```
2
2
7
```

Added by: Thanh-Vy Hua
Date: 2007-04-07
Time limit: 0.100s-3s
Source limit:50000B
Languages: All
Resource: Co-author Amber

SPOJ Problem Set (classical)

1487. Query on a tree III

Problem code: PT07J

You are given a node-labeled rooted tree with n nodes.

Define the query (x, k): Find the node whose label is k-th largest in the subtree of the node x. Assume no two nodes have the same labels.

Input

The first line contains one integer n ($1 <= n <= 10^5$). The next line contains n integers l_i ($0 <= l_i <= 10^9$) which denotes the label of the i-th node.

Each line of the following n - 1 lines contains two integers u, v. They denote there is an edge between node u and node v. Node 1 is the root of the tree.

The next line contains one integer m ($1 <= m <= 10^4$) which denotes the number of the queries. Each line of the next m contains two integers x, k. (k <= the total node number in the subtree of x)

Output

For each query (x, k), output the index of the node whose label is the k-th largest in the subtree of the node x.

Example

Input:

```
5
1 3 5 2 7
1 2
2 3
1 4
3 5
4
2 3
4 1
3 2
3 2
```

Output:

```
5
4
5
5
```

Added by: Thanh-Vy Hua
Date: 2007-04-07
Time limit: 1s-1.100s
Source limit: 50000B
Languages: All
Resource: Co-author Amber

SPOJ Problem Set (main)

1488. Balloons of JiaJia

Problem code: PT07K

In WinD's birthday, JiaJia opens a small contest for all guests: just typing all characters from 'a' to 'z', who fastest is the winner. The award is a very big bunch of balloons. Of course, JiaJia hopes his girlfriend - WinD can win, since he only carries two bunches with him and the award for this contest looks prettier. Unfortunately, life is not always as he expected, the winner is Amber. So JiaJia can only give WinD his second bunch of balloons. After the party ended, WinD said "I prefer that bunch of balloons to this one!! If you can't give me something the same as it, I don't want to see you anymore!". Well, JiaJia really has a brainstorm.

If we consider balloons as nodes, the colorful strings connect them as edges, and color of each balloon is label of each node then we have each bunch of balloons is a rooted, labeled, ordered tree.

We call *T* a labeled tree if each node is assigned a symbol from a fixed finite alphabet. And *T* is an ordered tree if a left-to-right order among siblings in *T* is given.

JiaJia can do these 3 operators to change the shape of a bunch of balloons (*T*):

- **Relabel** (recolor a balloon) Change the label of a node *v* in T.
- **Delete** (remove a balloon) Delete a non-root node *v* in T with parent *p*, making the children of *v* become the children of *p*. The children are inserted in the place of *v* as a subsequence in the left-to-right order of the children of *p*.
- **Insert** (add a balloon) The complement of **Delete**. Insert a node *v* with any label as a child of node *p* in *T*, making *v* be the parent of a consecutive subsequence of the children of *p*.

[IMAGE] [IMAGE]

Please help the poor guy JiaJia, use as few number of operators to make WinD's bunch of balloons exactly the same as Amber's bunch of balloons. His girlfriend can't wait any longer. Note that, he can only make changes with WinD's bunch.

Input

The input file contains two bunches of balloons (or trees). The first is of WinD, the second is of Amber.

The first line of the input file contains one integer *N* - number of balloons in the bunch *T1* ($1 <= N <= 500$). Balloons are numbered from 1 to *N*.

In next *N* lines, the *i*-th line contains the ordered children list of the *i*-th balloon. The first integer l[i] of the *i*-th line is the color id of the *i*-th balloon ($0 <= l[i] <= 10^4$). The second integer c[i] of the *i*-th line c[i] is the number of the children of the *i*-th balloon. Then c[i] integers followed, which means the ordered children list of the *i*-th balloon.

The remaining part of the input file is the description of the second bunch of balloons *T2*. The format is the same as *T1*.

Output

Output minimum number of operators JiaJia needs to do.

Example

Input:

```
3
1 2 2 3
2 0
1 0
2
1 1 2
3 0
```

Output:

```
2
```

Explanation

```
We cut the string connected between 1st balloon and
3rd balloons, then change the color of 2nd balloon to 3.
```

Added by: Thanh-Vy Hua
Date: 2007-04-07
Time limit: 0.100s-2.5s
Source limit:50000B
Languages: All
Resource: Co-author Amber

SPOJ Problem Set (classical)

1505. Whac-a-Mole

Problem code: MOLE

Map While visiting a traveling fun fair you suddenly have an urge to break the high score in the Whac-a-Mole game. The goal of the Whac-a-Mole game is to... well... whack moles. With a hammer. To make the job easier you have first consulted the fortune teller and now you know the exact appearance patterns of the moles. The moles appear out of holes occupying the n^2 integer points (x, y) satisfying 0 <= x, y <n in a two-dimensional coordinate system. At each time step, some moles will appear and then disappear again before the next time step. After the moles appear but before they disappear, you are able to move your hammer in a straight line to any position (x2 , y2) that is at ditance at most d from your current position (x1 , y1). For simplicity, we assume that yo can only move your hammer to a point having integer coordinates. A mole is whacke if the center of the hole it appears out of is located on the line between (x1 , y1) an (x2 , y2) (including the two endpoints). Every mole whacked earns you a point. Whe the game starts, before the first time step, you are able to place your hammer anywher you see fit.

Input

The input consists of several test cases. Each test case starts with a line containing three integers n, d and m, where n and d are as described above, and m is the total number of moles that will appear (1 <= n <= 20, 1 <= d <= 5, and 1 <= m <= 1000). Then follow m lines, each containing three integers x, y and t giving the position and time of the appearance of a mole (0 <= x, y < n and 1 <= t <= 10). No two moles will appear at the same place at the same time. The input is ended with a test case where n = d = m = 0. This case should not be processed.

Output

For each test case output a single line containing a single integer, the maximum possible score achievable.

Example

Input:

```
4 2 6
0 0 1
3 1 3
0 1 2
0 2 2
1 0 2
2 0 2
5 4 3
0 0 1
1 2 1
2 4 1
0 0 0
```

Output:

```
4
2
```

Added by: Abhilash I
Date: 2007-04-19
Time limit: 5s
Source limit: 50000B
Languages: All

SPOJ Problem Set (classical)

1526. Ranklist Sorting

Problem code: RSORTING

You are given the scores of several players in a competition. Your task is to create a ranklist of the players, sorted in decreasing order by score.

Unfortunately, the data structure used for the list of players supports only one operation, which moves a player from position i to position j without changing the relative order of other players. If i > j, the positions of players at positions between j and i - 1 increase by 1, otherwise if i < j the positions of players at positions between i + 1 and j decrease by 1.

This operation takes i steps to locate the player to be moved, and j steps to locate the position where he or she is moved to, so the overall cost of moving a player from position i to position j is i + j. Here, positions are numbered starting with 1.

Determine a sequence of moves to create the ranklist such that the sum of the costs of the moves is minimized.

Input

The input consists of **exactly 10** test cases. The first line of each test case contains n (2 <= n <= 1000), the number of players. Each of the following n lines contains one non-negative integer s_i (0 <= s_i <= 1000000), the scores of the players in the current order. You may assume that all scores are distinct.

Output

For each test case, print in one line the number of moves used to create the ranklist. The following lines should specify the moves in the order in which they are applied. Each move should be described by a line containing two integers i and j, which means that the player at position i is moved to position j. The numbers i and j must be separated by a single space.

Example

here only one test case

Input:
```
5
20
30
5
15
10
```

Output:
```
2
2 1
3 5
```

Added by:	Adrian Kuegel
Date:	2007-05-04
Time limit:	4s
Source limit:	50000B
Languages:	All
Resource:	own problem, used in BOI 2007

SPOJ Problem Set (classical)

1536. Help Blue Mary Please! (Act III)

Problem code: BLUEEQ3

Background

This morning Blue Mary wrote some equations on a piece of paper and left it on her desk.After solving some problems in SPOJ,she found that her classmate H.L. replaced all characters on the paper with some other ones.H.L. told her he replaced the same characters with the same ones,and different characters with different ones because of his goodness.Now Mary needs your help to get the original equations back.(See problem BLUEEQ)

Input

Ten test cases(given one after another,you have to process all!)For each test case, the first line is a single integer n(n<=21).Next 3 lines contains 3 strings, each of them has a length of n and contains only first n capital latin characters.The sum of the numbers the first two strings indicates equals to the number the third string incicates.The numbers can have leading zeros and each of their bases is n.

Output

For each test case you should output one line contains n numbers separated by spaces, which is a permutation of integer numbers 0 to n-1.Number x is on the k-th position iff x is replaced by the kth capital latin character.There is one and only one solution for each test case.

Example

Input:
```
5
ABCED
BDACE
EBBAA
[and 9 test cases more]
```
Output:
```
1 0 3 4 2
[and 9 test cases more]
```

Added by: Blue Mary
Date: 2007-05-08
Time limit: 17s
Source limit:50000B
Languages: All except: C99 strict
Resource: description by Blue Mary

SPOJ Problem Set (classical)

1538. Making Jumps

Problem code: MKJUMPS

A knight is a piece used in the game of chess. The chessboard itself is square array of cells. Each time a knight moves, its resulting position is two rows and one column, or two columns and one row away from its starting position. Thus a knight starting on row r, column c - which we'll denote as (r,c) - can move to any of the squares (r-2,c-1), (r-2,c+1), (r-1,c-2), (r-1,c+2), (r+1,c-2), (r+1,c+2), (r+2,c-1), or (r+2,c+1). Of course, the knight may not move to any square that is not on the board.

Suppose the chessboard is not square, but instead has rows with variable numbers of columns, and with each row offset zero or more columns to the right of the row above it. The figure to the left illustrates one possible configuration. How many of the squares in such a modified chessboard can a knight, starting in the upper left square (marked with an asterisk), not reach in any number of moves without resting in any square more than once?

[IMAGE]

If necessary, the knight is permitted to pass over regions that are outside the borders of the modified chessboard, but as usual, it can only move to squares that are within the borders of the board.

Input

There will be multiple cases to consider. The input for each case begins with an integer n, between 1 and 10, that specifies the number of rows in the modified chessboard. Following n there will be n pairs of integers, with the ith pair corresponding to the ith row of the chessboard. The first integer of each pair indicates the number of squares skipped at the beginning of the row. The second integer indicates the number of squares in the row (which will always be at least 1).The last case will be followed by the integer 0.

For example, input for the case illustrated by the chessboard shown above would be:

7 0 3 0 3 0 4 0 4 1 3 1 7 4 4

The maximum dimensions of the board will be 10 rows and 10 columns. That is, any modified chessboard specified by the input will fit completely on a 10 row, 10 column board.

Output

For each input case, display the case number (1, 2, ...), and the number of squares that the knight can not reach. Display the results in the format shown in the examples below.

Example

Input:

```
7 0 3 0 3 0 4 0 4 1 3 1 7 4 4
3 0 3 0 3 0 3
2 0 1 2 1
0
```

Output:

```
Case 1, 4 squares can not be reached.
Case 2, 1 square can not be reached.
Case 3, 0 squares can not be reached.
```

Added by: Camilo Andrés Varela León
Date: 2007-05-11
Time limit: 5s
Source limit: 50000B
Languages: All
Resource: North Central North America Regional Programming Contest - 2003

SPOJ Problem Set (classical)

1552. Mobiles

Problem code: MOBILE2

You have been asked to buy a gift for your baby brother, Ike. However, you have noticed that Ike has a very particular taste in gifts. He only likes gifts that are configured in his particular style.

You have found a shop that sells mobiles. A mobile is a multi-layered decoration that is typically hung from the roof. Each mobile consists of a series of horizontal rods connected by vertical wires. Each rod has a wire hanging from both ends, which holds either another horizontal rod or a toy.

A sample mobile is shown below:

[IMAGE]

To satisfy your brother, you need to find a mobile that can be reconfigured so that:

(i) any two toys are either at the same level (that is, joined to the roof by the same number of rods), or di.er by only one level;

(ii) for any two toys that differ by one level, the toy to the left is further down than the toy to the right.

Mobiles can be reconfigured by performing swaps. A swap involves taking some rod, unhooking whatever is hanging beneath the left and right ends, and reattaching them at opposite ends (that is, the right and left ends respectively). This process does not modify the ordering of any rods or toys further down.

Since you are training for the Informatics Olympiad, you decide to write a program to test whether a given mobile can be reconfigured into a gift that Ike will like!

As an example, consider the mobile illustrated earlier. Ike will not like this mobile. Although it satisfies condition (i), it breaks condition (ii) -- the toy at the leftmost end is at a higher level than the toys to its right.

However, the mobile can be reconfigured into a mobile that Ike will like. The following swaps are required:

1. First, the left and right ends of rod 1 are swapped. This exchanges the positions of rods 2 and 3, resulting in the following configuration:

[IMAGE]

2. Second, and finally, the left and right ends of rod 2 are swapped. This moves rod 4 to the left end of rod 2, and the toy to the right end of rod 2.

[IMAGE]

It can be seen that this final mobile satisfies Ike's requirements. All toys are at most one level apart, and the toys at a lower level are further to the left than the toys at a higher level.

Your task is, given a description of a mobile, to determine the smallest number of swaps required to reconfigure it so that Ike will like it (if this is possible). You may assume that the toys can never get in each other's way.

Input

Multiple test cases, the number of them will be given at the very first line.

For each test case:

The first line of input will contain the single integer n (1 <= n <= 100 000) representing the number of rods in the mobile. The rods are numbered 1, 2 , ..., n.

The following n lines will describe the connections for each rod. Specifically, the ith of these lines will describe rod i. Each of these lines will contain two integers *l r* separated by a single space, indicating what is hung beneath the left and right ends of the rod respectively. If a toy is hung beneath this rod, the corresponding integer *l* or *r* will be -1. Otherwise the integer *l* or *r* will be the number of a rod that is hung beneath this rod.

If there are any rods hanging beneath rod i, these rods will have numbers strictly greater than i. Rod 1 is the single rod at the top of the mobile.

Output

Output should consist of a single line containing a single integer, giving the smallest number of swaps required to reconfigure the mobile according to Ike's constraints. If this is not possible, you should output the integer -1.

Example

Input:

```
1
6
2 3
-1 4
5 6
-1 -1
-1 -1
-1 -1
```

Output:

```
2
```

Warning: large input/output data,be careful with certain languages

---

Added by: Blue Mary
Date: 2007-05-14
Time limit: 17s
Source limit:50000B
Languages: All except: C99 strict
Resource: Asia-Pacific Informatics Olympiad(APIO) 2007

SPOJ Problem Set (classical)

1553. Backup Files

Problem code: BACKUP

You run an IT company that backs up computer data for large offices. Backing up data is not fun, and so you design your system so that the different offices can back up each others' data while you sit at home and play computer games instead.

The offices are all situated along a single street. You decide to pair up the offices, and for each pair of offices you run a network cable between the two buildings so that they can back up each others' data.

However, network cables are expensive. Your local telecommunications company will only give you k network cables, which means you can only arrange backups for k pairs of offices (2k offices in total). No office may belong to more than one pair (that is, these 2k offices must all be different). Furthermore, the telecommunications company charges by the kilometre. This means that you need to choose these k pairs of offices so that you use as little cable as possible. In other words, you need to choose the pairs so that, when the distances between the two offices in each pair are added together, the total distance is as small as possible.

As an example, suppose you had five clients with offices on a street as illustrated below. These offices are situated 1 km, 3 km, 4 km, 6km and 12km from the beginning of the street. The telecommunications company will only provide you with k = 2 cables.

[IMAGE]

The best pairing in this example is created by linking the first and second offices together, and linking the third and fourth offices together. This uses k = 2 cables as required, where the first cable has length 3km - 1km = 2 km, and the second cable has length 6km - 4km = 2 km. This pairing requires a total of 4km of network cables, which is the smallest total possible.

Input

Multiple test cases, the number of them will be given at the very first line.

For each test case:

The first line of input will contain the integers n and k, representing the number of offices on the street ($2 \leq n \leq 100\,000$) and the number of available network cables ($1 \leq k \leq n/2$).

The following n lines will each contain a single integer ($0 \leq s \leq 1\,000\,000\,000$), representing the distance of each office from the beginning of the street. These integers will appear in sorted order from smallest to largest. No two offices will share the same location.

Output

Output should consist of a single positive integer, giving the smallest total length of network cable required to join 2k distinct offices into k pairs.

Example

Input:

```
1
5 2
1
3
4
6
12
```

Output:

```
4
```

Explanation

The sample input above represents the example scenario described earlier.

Warning: large input/output data,be careful with certain languages

Blue Mary's Note: test data has been modified on Dec. 5, 2007. All the solutions have been rejudged.

Added by: Blue Mary
Date: 2007-05-14
Time limit: 17s
Source limit:50000B
Languages: All except: C99 strict
Resource: Asia-Pacific Informatics Olympiad(APIO) 2007

SPOJ Problem Set (classical)

1554. Zoo

Problem code: ZOO

The pride of the Asia-Pacific region is the newly constructed Great Circular Zoo. Situated on a small Pacific island, it consists of a large circle of different enclosures, each containing its own exotic animal as illustrated below.

[IMAGE]

You are in charge of public relations for the zoo, which means it is your job to keep people as happy as possible. A busload of schoolchildren has just arrived, and you are eager to please them. However, this is no easy taskIthere are animals that some children love, and there are animals that some children fear. For example, little Alex loves monkeys and koalas because they are cute, but fears lions because of their sharp teeth. On the other hand, Polly loves lions because of their beautiful manes, but fears koalas because they are extremely smelly.

You have the option of removing some animals from their enclosures, so that children are not afraid. However, you are worried that if you remove too many animals then this will leave the children with nothing to look at. Your task is to decide which animals to remove so that as many children can be made happy as possible.

Each child is standing outside the circle, where they can see five consecutive enclosures. You have obtained a list of which animals each child fears, and which animals each child loves. A child will be made happy if either at least one animal they fear is removed from their field of vision, or at least one animal they love is not removed from their field of vision.

For example, consider the list of children and animals illustrated below:

[IMAGE]

Child	Enclosures Visible	Fears	Loves
Alex	2, 3, 4, 5, 6	Enclosure 4	Enclosures 2, 6
Polly	3, 4, 5, 6, 7	Enclosure 6	Enclosure 4
Chaitanya	6, 7, 8, 9, 10	Enclosure 9	Enclosures 6, 8
Hwan	8, 9, 10, 11, 12	Enclosure 9	Enclosure 12
Ka-Shu	12, 13, 14, 1, 2	Enclosures 12, 13, 2	-

Suppose you remove the animals from enclosures 4 and 12. This will make Alex and Ka-Shu happy, because at least one animal that they fear has gone. This will also keep Chaitanya happy, since both enclosures 6 and 8 still contain animals that he loves. However, both Polly and Hwan will be unhappy, since they cannot see any animals that they love but they can still see all the animals that they fear. This arrangement therefore gives a total of three happy children.

Now suppose you put these animals back into their enclosures, and remove the animals from enclosures 4 and 6 instead. Alex and Polly will be happy because the animals that they fear in enclosures 4 and 6 have gone. Chaitanya will be happy because, even though animal 6 has gone, he

can still see the animal in enclosure 8 which he loves. Likewise, Hwan will be happy because she can now see the animal in enclosure 12, which she loves. The only person unhappy will be Ka-Shu.

Finally, suppose you put the animals back once more and then remove only the animal from enclosure 13. Ka-Shu will now be happy since one animal that he fears has been removed, and Alex, Polly, Chaitanya and Hwan will all be happy since they can all see at least one animal that they love. Thus this arrangement gives five happy children, the largest number possible.

Input

Multiple test cases, the number of them will be given at the very first line.

For each test case:

The first line will be of the form N C, where N is the number of animal enclosures (10 <= N <= 10 000) and C is the number of children (1 <= C <= 50 000). The enclosures are numbered 1, 2, ...,N clockwise around the circle.

Following this will be C additional lines of input, where each line describes a single child. Each of these lines will be of the form:

```
E F L X1 X2 ... XF Y1 Y2 ... YL;
```

where:

- E is the first enclosure that the child can see (1 <= E <= N). In other words, the child can see enclosures E, E + 1, E + 2, E + 3 and E + 4. Note that numbers larger than N wrap back around the circle, so if N = 14 and E = 13 then the child can see enclosures 13, 14, 1, 2 and 3.
- F is the number of animals that the child fears, and L is the number of animals that the child loves.
- Enclosures X1,...,XF contain the animals that the child fears (1<=X1,...,XF<=N).
- Enclosures Y1,...,YL contain the animals that the child loves (1<=Y1,...,YL<=N).
- No two of the integers X1,...,XF,Y1,...,YL are equal, and all of these integers describe enclosures that the child can see.

Children will be listed in sorted order according to the first enclosure E (so the child with lowest E will appear first and the child with largest E will appear last). Note that more than one child may have the same first enclosure E.

Output

Output must consist of a single integer, giving the largest number of children that can be made happy.

Example

Sample Input:

```
2
14 5
2 1 2 4 2 6
3 1 1 6 4
6 1 2 9 6 8
8 1 1 9 12
12 3 0 12 13 2
12 7
```

```
1 1 1 1 5
5 1 1 5 7
5 0 3 5 7 9
7 1 1 7 9
9 1 1 9 11
9 3 0 9 11 1
11 1 1 11 1
```

Sample Output:

```
5
6
```

Explanation:

The first sample case is the example discussed earlier, in which all C = 5 children can be made happy. The second sample case is an example in which it is impossible to make all C = 7 children happy.

Warning: large input/output data,be careful with certain languages

---

Added by: Blue Mary
Date: 2007-05-14
Time limit: 17s
Source limit: 50000B
Languages: All except: C99 strict
Resource: Asia-Pacific Informatics Olympiad(APIO) 2007

SPOJ Problem Set (classical)

1557. Can you answer these queries II

Problem code: GSS2

Being a completist and a simplist, kid Yang Zhe cannot solve but get Wrong Answer from most of the OI problems. And he refuse to write two program of same kind at all. So he always failes in contests.

When having a contest, Yang Zhe looks at the score of every problems first. For the problems of the same score, Yang Zhe will do only one of them. If he's lucky enough, he can get all the scores wanted.

Amber is going to hold a contest in SPOJ. She has made a list of N candidate problems, which fit Yang Zhe very well. So Yang Zhe can solve any problem he want. Amber lined up the problems, began to select. She will select a subsequence of the list as the final problems. Being A girl of great compassion, she'd like to select such a subsequence (can be empty) that Yang Zhe will get the maximal score over all the possible subsequences.

Amber found the subsequence easily after a few minutes. To make things harder, Amber decided that, Yang Zhe can take this contest only if Yang Zhe can answer her Q questions. The question is: if the final problems are limited to be a subsequence of $list[X..Y]$ ($1 <= X <= Y <= N$), what's the maximal possible score Yang Zhe can get?

As we know, Yang Zhe is a bit idiot (so why did he solve the problem with a negative score?), he got Wrong Answer again... Tell him the correct answer!

Input

- Line 1: integer N ($1 <= N <= 100000$);
- Line 2: N integers denoting the score of each problem, each of them is a integer in range [-100000, 100000];
- Line 3: integer Q ($1 <= Q <= 100000$);
- Line 3+i ($1 <= i <= Q$): two integers X and Y denoting the ith question.

Output

- Line i: a single integer, the answer to the ith question.

Example

Input:

```
9
4 -2 -2 3 -1 -4 2 2 -6
3
1 2
1 5
4 9
```

Output:

```
4
5
3
```

Warning: large input/output data,be careful with certain languages

---

Added by: Blue Mary
Date: 2007-05-16
Time limit: 1s-5s
Source limit:50000B
Languages: All except: C99 strict
Resource: Description, standard program and test data by Yang Zhe

SPOJ Problem Set (classical)

1644. Trees

Problem code: TREEOI14

Byteasar has a cottage. Lately, he has bought n trees and had them planted all in one row. Byteasar does not, however, like the order which the trees have been planted in. It particularly annoys him that tall and short ones have been mixed up, and the composition does not meet his aesthetic criteria.

Byteasar has invented a *disorder coefficient* so as to allow the gardener to comprehend his intentions: the lower the value of the coefficient the prettier the row of trees. It is defined in the following way: $|h_1-h_2|+|h_2-h_3|+...+|h_{n-1}-h_n|$, where $h_1, h_2, \ldots, h_n$ are the heights of consecutive trees in a row.

Replanting is a very toilsome and cumbersome task, therefore Byteasar has ordered the gardener to replant two trees at the most (i.e. interchange their positions). The task of the gardener is to choose the pair to replant in a way that makes the disorder coefficient the smallest.

The gardener is not sure if he has chosen the correct pair of trees and he fears he may lose his job if he is mistaken. Help him: for each tree calculate the minimal disorder coefficient that may be attained by switching places with any other tree.

Task

Write a program which:

- reads the height of the consecutive trees in a row from the standard input,
- for each tree calculates the minimal disorder coefficient that may be attained should it switch places with some other tree (or should there be no change at all),
- writes the outcome to the standard output.

Input

The first line of the standard input contains one integer n ($2 <= n <= 50000$). The other contains n integers h_i ($1 <= h_i <= 100000000$) separated by single spaces, denoting the height of the consecutive trees in the row.

Output

The output should consist of precisely n lines. The i-th line should contain a single integer - the smallest disorder coefficient attainable when considering replanting of the i-th tree.

Example

Input:

```
5
7 4 5 2 5
```

Output:

```
7
7
8
7
7
```

*** Added some unofficial tests**

Added by: Thanh-Vy Hua
Date: 2007-06-09
Time limit: 1s-6s
Source limit: 50000B
Languages: All
Resource: Polish Olympiad 14

SPOJ Problem Set (classical)

1671. Another Mathematical Problem

Problem code: AMATH

Given two numbers n (1<=n<10^100) and k (1<=k<=100), you are to determine whether there exists a positive integer T which satisfies that for every positive integer a, n^(a+T)-n^a is divisible by 10^k.

Input

Multiple test cases. Each test case contains two space-separated integers n and k. Input terminate by EOF.

The number of test cases will not more than 20.

Output

For each test case, you should output the smallest positive integer number T which satisfies the condition above, or -1 if it doesn't exist.

Example

Input:
```
32 2
```

Output:
```
4
```

Added by: Blue Mary
Date: 2007-07-01
Time limit: 3s
Source limit: 2048B
Languages: All except: C99 strict
Resource: Folklore, description and standard program by Blue Mary

SPOJ Problem Set (classical)

1672. The Great Indian Wedding

Problem code: GIWED

A wedding is to be organized in a rectangular park of dimensions M by N. Some parts of the park are covered by K rectangular carpets. These carpets, produced by ItSucks Corporation are revolutionary self cleaning carpets - they suck any liquid they come in contact with! The organizer wants to water the park to keep the grass fresh. If there were no carpets, the organizer could have used a single pipe to water the whole park but unforunately, the water doesn't seep through the carpets. The organizer has at his diposal L pipes. The pipes would be placed at fixed locations chosen by the organizer and can't be moved. Water spreads from a pipe in all directions unless obstructed by the park boundary or a carpet. What is the maximum area that can be watered using these L pipes?

Input

The first line of the input contains a single integer T, the number of test cases (1<=T<=30) . Each test case starts with a single line containg the values M,N,K and L (1<=M<=10000, 1<=N<=10000, 0<=K<=50, 1<=L<=10). It is followed by K lines, each line containing 4 integers separated by single spaces, x_1, y_1, x_2, y_2 where (x_1, y_1) and (x_2, y_2) are the zero based coordinates of lower left and upper right vertex of the carpet. Assume that $x_1 < x_2$ and $y_1 < y_2$. The carpets may cover each other. Water would not be able to seep through even if two carpets touch in a corner.

Output

For each test case, print the maximum area that can be watered on a single line

Example

Input:

```
2
10 10 0 1
10 10 1 1
3 3 4 4
```

Output:

```
100
99
```

Added by: Rahul Garg
Date: 2007-07-04
Time limit: 1s
Source limit:50000B
Languages: All
Resource: C-maphore, Tryst 2007

SPOJ Problem Set (classical)

1673. Ambitious Manager

Problem code: AMBM

The Bogus Corporation distributes salary to its employees in a weird manner. The salary is distributed every K days, and instead of same salary for each day, the salary for the i^{th} day is a_i. An ambitious young manager, fresh from Institute of Mismanagement, observes that people usually prefer to take leave towards the end of this period of K days, when the workload is higher. Instead of revising each of the a_i's, the manager comes up with a quick fix solution - he redefines the new salary on the i^{th} day as $b_i = a_i + 2a_{i-1} + 2^2 a_{i-2} + 2^3 a_{i-3} + ... + 2^{i-1} a_1$. Baba, one of the employees, is in a dire financial crisis, and must accumulate at least N rupees at the end of the forthcoming period. Being a lazy worker that he is, he is interested in finding out if attending particular days would guarantee him exactly N rupees at the end of the period. Can you help Baba?

Input

First line contains a single integer integer T, the number of test cases (1<=T<=100). Each test case is described on two lines. First line contains two integers, N and K (1<=N<=2^{63}-1, 1<=K<=50) , the second line contains a space separated list of K integers, the a_i's (1<=a_i<=1000).

Output

For each test case, output on a single line 1-based indices of the days (separated by a single space) he should attend to ensure a salary of exactly N rupees at the end of the period. The indices should be printed in the sorted order. In case of multiple answers, output any one of them. If there is no answer, print -1.

Example

Input:
```
2
9 3
1 1 2
10 2
2 3
```
Output:
```
1 3
-1
```

Added by: Rahul Garg
Date: 2007-07-04
Time limit: 1s
Source limit: 50000B
Languages: All
Resource: C-maphore, Tryst 2007

SPOJ Problem Set (classical)

1674. The Explosion

Problem code: EXPLOSN

The day of 6.XII.2003 in Megabyteland began calm and quietly as any other day. Some people went to work, some - to school, some - to store to buy food. Drivers were traditionally stucked in traffic jams, drinking coffee and reading morning newspaper. Suddenly the regularity of this day was disturbed by huge explosion."They blew up the embassy of Bajtocja!!!" - somebody cried. Everybody began to run away in panic.

Police works pretty good in Megabyteland and first radiocars appeared near the embassy only few seconds after the explosion. All the people near the embassy were detained. Some of these people are the organizers of the explosion, but the others could by just occasional witnesses. During the testification each person named exactly one perpetrator. It is known, that if a man is not a perpetrator, than he always says the truth (he haven't a reason to lie, have he?). However, perpetrators want to make the work of the police more difficult, so a perpetrator can name any person during the testification (even himself).

The policemen are in the very hard situation. They should arrest some group of potential perpetrators, but it is difficult to determine who is guilty and who is not from the data they have. There exists many groups of potential perpetrators, that don't contradict to any of the testimonies. The policemen want to arrest as small innocent people as possible. So they would like to choose the group with minimal number of people.

Write a program that, given the number of detained people and their testimonies, will determine the number of people in the smallest group of potential perpetrators, that don't contradict to the testimonies.

Input

The first line of the input contains a single integer T, the number of testcases (1<=T<=10).

First line of each testcase contains integer number N (2 <= N <= 100000), equal to the number of detained people (the people are numbered from 1 to N). The i-th of the following N lines contain one integer number Pi (1 <= Pi <= N). Here Pi is the man whom i-th man testified to be guilty.

Output

The output should consist of T lines, containing one integer number for each testcase - the number of people in the smallest group of potential perpetrators, that don't contradict to the testimonies.

Example

Input:

```
1
3
2
3
```

```
1
```

Output:

```
2
```

Added by:	Jin Bin
Date:	2007-07-04
Time limit:	1s-2s
Source limit:	50000B
Languages:	All except: C99 strict
Resource:	First Minsk Training Camp for NEERC teams, day 2 - MWPZ' 2003

SPOJ Problem Set (classical)

1675. Fusion Cube

Problem code: FUSION

The Bogus Corporation claims to have solved the energy crisis by devising a method to perform controlled fusion reaction! The set up consists of a cube of side length N meters, which contain K point sources of electrons. Each of the sources can be configured to emanate an electron along any of the six possible directions corresponding to +X axis, -X axis, +Y axis, -Y axis, +Z axis, -Z axis. The cube is filled with a medium in which the electrons travel with a velocity of 1m/s. At time t=0, all the sources are switched on simultaneously, emanating a single electron along the configured direction. An electron travels in a straight line until it strikes the boundary of the cube or collides with another electron. A collision between two electrons can occur in two possible ways - a head-on collision, and a side-on collision. In a head-on collision, both the electrons rebound in opposite directions with the same speed. A side-on collision occurs when the colliding electrons are travelling in mutually perpendicular directions. After a side-on collision, the colliding electrons are deflected by 90 degrees, rebounding in mutually perpendicular directions with the same speed, the plane of motion remaining the same. Note that throughout the experiment, the direction of motion of an electron remains oriented along one of the coordinate axes. If more than two electrons collide simultaneously, resolve the collision pairwise, where any two of the colliding electrons can be paired. To maximize the chances of initiating the fusion reaction, we would like to maximize the time before an electron hits a boundary wall of the cube. Given the location of the K sources, determine the orientation of the sources such that this time is maximized. Output this maximum value.

Input

First line contains a single integer T, the number of test cases (1<=T<=50), followed by the description of the test cases. The first line of each test case contains two integers, K and N respectively (1<=K<=100, 1<=N<=1000). It is followed by K lines, where each line contains space separated three integers representing the X, Y and Z coordinates of the particular source. The coordinates of the diagonally opposite corners of the cube are (0,0,0) and (N,N,N). All the sources will lie strictly inside the cube.

Output

For each test case, output the maximum value of the time before the first electron hits the boundary of the cube on a single line.

Example

Input:

```
2
1 10
5 5 5
2 10
1 1 1
```

```
1 1 2
```
Output:
```
5
9
```

Added by:	Rahul Garg
Date:	2007-07-04
Time limit:	1s
Source limit:	50000B
Languages:	All
Resource:	C-maphore, Tryst 2007

SPOJ Problem Set (classical)

1676. Text Generator

Problem code: GEN

LoadingTime has been given a task these days. He is required to write a tool called Text Generator. This software is widely used among the kids who are under seven. It generates an article with the size of a given number L for users. If an article contains at least one word which the users know, we consider it readable. Now, LoadingTime wants to know, how many readable articles can it generates, and he can improve his Text Generator. Could you help him??

Input

The input contains multiple test cases.

The first line of each test case contains two integer N (1 <= N <= 10), L (1 <= L <= 1000000). The following N lines contain N words representing the words knew by the users. All the words and the generated article only contain uppercase letters, and the length of each word is not greater than 6.

Output

For each test case, your program should output a integer as LoadingTime required. As the number could be quite large, you only need to print the answer modulo 10007.

Example

Input:
```
2 2
A
B
2 10000
ABC
B
```

Output:
```
100
5960
```

Added by: Jin Bin
Date: 2007-07-05
Time limit: 5s
Source limit:50000B
Languages: All except: C99 strict
Resource:

SPOJ Problem Set (classical)

1677. Halloween treats

Problem code: HALLOW

Every year there is the same problem at Halloween: Each neighbor is only willing to give a certain total number of sweets on that day, no matter how many children call on him, so it may happen that a child will get nothing if it is too late. To avoid conflicts, the children have decided they will put all sweets together and then divide them evenly among themselves. From last year's experience of Halloween they know how many sweets they get from each neighbour. Since they care more about justice than about the number of sweets they get, they want to select a subset of the neighbours to visit, so that in sharing every child receives the same number of sweets. They will not be satisfied if they have any sweets left which cannot be divided.

Your job is to help the children and present a solution.

Input

The input contains several test cases.
The first line of each test case contains two integers **c** and **n** (*1 <= c <= n <= 100000*), the number of children and the number of neighbours, respectively. The next line contains **n** space separated integers $\mathbf{a_1}$ **, ... ,** $\mathbf{a_n}$ (*1 <=* a_i *<= 100000*), where $\mathbf{a_i}$ represents the number of sweets the children get if they visit neighbour **i**.

The last test case is followed by two zeros.

Output

For each test case output one line with the indices of the neighbours the children should select (here, index **i** corresponds to neighbour **i** who gives a total number of $\mathbf{a_i}$ sweets). If there is no solution where each child gets at least one sweet, print "no sweets" instead. Note that if there are several solutions where each child gets at least one sweet, you may print any of them.

Example

Input:
```
4 5
1 2 3 7 5
3 6
7 11 2 5 13 17
0 0
```

Output:
```
3 5
2 3 4
```

Added by: Simon Gog
Date: 2007-07-05
Time limit: 15s
Source limit:50000B
Languages: All
Resource: University of Ulm Local Contest 2007/2008

SPOJ Problem Set (classical)

1678. Royal Treasury

Problem code: TREASURY

Once upon a time in a kingdom far far away, the royal treasury started getting emptier and emptier. The king decided to change the situation and he invented a new system of cooperation with in the office of the royal treasurer. The clerks of the office are supposed to form pairs (in order to avoid being bribed) in such away that each pair is formed by a clerk and his/her direct subordinate. Your task is to compute, given the structure of the office of the treasurer, the maximum number of pairs that can be formed this way and in how many different ways this is possible.
The office of the treasurer is led by George Skinflint. Each clerk has zero, one or more subordinates and is a subordinate of a single clerk (except for George Skinflint who is responsible only to the king himself). The number of clerks does not exceed 1000. Your task is to compute the maximum number of pairs that can be formed by clerks in such a way that every pair is formed by a clerk and his/her direct subordinate. In addition, you should also compute the number of ways such pairs can be formed. Note that some clerks need not be contained in a pair.

Input

The input contains multiple testcases.

The first line of each testcase contains a single number N that represents the number of clerks 1 <= N <= 1000. The clerks are assigned unique ID numbers from the range between 1 and N. The ID number of the treasurer (Skinflint) is 1. Each of the following N lines corresponds to one of the clerks: it contains his/her ID number, the number K of his/her subordinates, 0 <= K <= 999, and the ID numbers of his/her K subordinates separated by single spaces. You can assume that the line corresponding to a clerk never appears before the line corresponding to his/her supervisor.

Output

The output for each testcase should consist of two lines. The first line of the output should contain a single number that represents the maximum number M of pairs that the clerks can form. The second line should contain the number of different ways in which the clerks can form M pairs obeying the rules given by the king.

Example

Input:

```
7
1 3 2 4 7
2 1 3
4 1 6
3 0
7 1 5
5 0
6 0
```

Output:

```
3
4
```

Added by:	Jin Bin
Date:	2007-07-06
Time limit:	0.5s
Source limit:	50000B
Languages:	All except: C99 strict
Resource:	CEOI 2007, day 2

SPOJ Problem Set (classical)

1681. Cylinder

Problem code: CYLINDER

Using a sheet of paper and scissors, you can cut out two faces to form a cylinder in the following way:

1. Cut the paper horizontally (parallel to the shorter side) to get two rectangular parts.
2. From the first part, cut out a circle of maximum radius. The circle will form the bottom of the cylinder.
3. Roll the second part up in such a way that it has a perimeter of equal length with the circle's circumference, and attach one end of the roll to the circle. Note that the roll may have some overlapping parts in order to get the required length of the perimeter.

Given the dimensions of the sheet of paper, can you calculate the biggest possible volume of a cylinder which can be constructed using the procedure described above?

Input Specification

The input consists of several test cases. Each test case consists of two numbers **w** and **h** (*1 <= w <= h <= 100*), which indicate the width and height of the sheet of paper.

The last test case is followed by a line containing two zeros.

Output Specification

For each test case, print one line with the biggest possible volume of the cylinder. Round this number to 3 places after the decimal point.

Sample Input

```
10 10
10 50
10 30
0 0
```

Sample Output

```
54.247
785.398
412.095
```

In the first case, the optimal cylinder has a radius of about 1.591549, in the second case, the optimal cylinder has a radius of 5, and in the third case, the optimal cylinder has a radius of about 3.621795.

Added by:	Adrian Kuegel
Date:	2007-07-06
Time limit:	4s
Source limit:	50000B
Languages:	All
Resource:	own problem, used in University of Ulm Local Contest 2007

SPOJ Problem Set (classical)

1683. Expressions

Problem code: EXPRESS

Arithmetic expressions are usually written with the operators in between the two operands (which is called infix notation). For example, *(x+y)*(z-w)* is an arithmetic expression in infix notation. However, it is easier to write a program to evaluate an expression if the expression is written in postfix notation (also known as reverse polish notation). In postfix notation, an operator is written behind its two operands, which may be expressions themselves. For example, *x y + z w - ** is a postfix notation of the arithmetic expression given above. Note that in this case parentheses are not required.

To evaluate an expression written in postfix notation, an algorithm operating on a stack can be used. A stack is a data structure which supports two operations:

1. **push**: a number is inserted at the top of the stack.
2. **pop**: the number from the top of the stack is taken out.

During the evaluation, we process the expression from left to right. If we encounter a number, we push it onto the stack. If we encounter an operator, we pop the first two numbers from the stack, apply the operator on them, and push the result back onto the stack. More specifically, the following pseudocode shows how to handle the case when we encounter an operator O:

```
a := pop();
b := pop();
push(b O a);
```

The result of the expression will be left as the only number on the stack.

Now imagine that we use a queue instead of the stack. A queue also has a push and pop operation, but their meaning is different:

1. **push**: a number is inserted at the end of the queue.
2. **pop**: the number from the front of the queue is taken out of the queue.

Can you rewrite the given expression such that the result of the algorithm using the queue is the same as the result of the original expression evaluated using the algorithm with the stack?

Input Specification

The first line of the input contains a number **T** (*T <= 200*). The following **T** lines each contain one expression in postfix notation. Arithmetic operators are represented by uppercase letters, numbers are represented by lowercase letters. You may assume that the length of each expression is less than *10000* characters.

Output Specification

For each given expression, print the expression with the equivalent result when using the algorithm with the queue instead of the stack. To make the solution unique, you are not allowed to assume that the operators are associative or commutative.

Sample Input

```
2
xyPzwIM
abcABdefgCDEF
```

Sample Output

```
wzyxIPM
gfCecbDdAaEBF
```

Added by: Adrian Kuegel
Date: 2007-07-06
Time limit: 4s
Source limit: 50000B
Languages: All
Resource: own problem, used in University of Ulm Local Contest 2007

SPOJ Problem Set (classical)

1684. Frequent values

Problem code: FREQUENT

You are given a sequence of **n** integers $\mathbf{a_1}$, $\mathbf{a_2}$, ... , $\mathbf{a_n}$ in non-decreasing order. In addition to that, you are given several queries consisting of indices **i** and **j** (*1 <= i <= j <= n*). For each query, determine the most frequent value among the integers $\mathbf{a_i}$, ... , $\mathbf{a_j}$.

Input Specification

The input consists of several test cases. Each test case starts with a line containing two integers **n** and **q** (*1 <= n, q <= 100000*). The next line contains **n** integers $\mathbf{a_1}$, ... , $\mathbf{a_n}$ (*-100000 <=* a_i *<= 100000*, for each *i (- {1, ..., n}*) separated by spaces. You can assume that for each *i (- {1, ..., n-1}:* a_i *<=* a_{i+1}. The following **q** lines contain one query each, consisting of two integers **i** and **j** (*1 <= i <= j <= n*), which indicate the boundary indices for the query.

The last test case is followed by a line containing a single *0*.

Output Specification

For each query, print one line with one integer: The number of occurrences of the most frequent value within the given range.

Sample Input

```
10 3
-1 -1 1 1 1 1 3 10 10 10
2 3
1 10
5 10
0
```

Sample Output

```
1
4
3
```

A naive algorithm may not run in time!

Added by: Adrian Kuegel
Date: 2007-07-06
Time limit: 5s
Source limit:50000B
Languages: All
Resource: University of Ulm Local Contest 2007

SPOJ Problem Set (classical)

1685. Grocery store

Problem code: GROCERY

A cashier in a grocery store seems to have difficulty in distinguishing the multiplication symbol and the addition symbol. To make things easier for him, you want to buy goods in such a way that the product of their prices is the same as the sum of their prices.

Of course, if you buy only one item, this is always true. With two items and three items, it still seems quite a boring task to you, so now you are interested in finding possible prices of four items such that the sum of the four prices is equal to the product of the four prices. You should consider the prices are in Euro with two digits after the decimal point. Obviously, each product costs at least one cent.

Input Specification

This problem has no input.

Output Specification

Print all solutions which have a sum of the four items of at most **20.00 Euro**. For each solution, print one line with the prices of the four items in non-decreasing order, with one space character between them. You may print the solutions in any order, but make sure to print each solution only once.

Sample Output

```
0.50 1.00 2.50 16.00
1.25 1.60 1.75 1.84
1.25 1.40 1.86 2.00
...
```

Added by: Adrian Kuegel
Date: 2007-07-06
Time limit: 1.600s
Source limit: 600B
Languages: All
Resource: University of Ulm Local Contest 2007

SPOJ Problem Set (classical)

1687. Logic II

Problem code: LOGIC2

Some day in 2003 in Byteland began calm and quietly as any other day. Some people went to work, some to school, some to store to buy food. Drivers were traditionally stucked in traffic jams, drinking coffee and reading morning newspaper. Suddenly the regularity of this day was disturbed by huge explosion."They blew up the embassy of Bajtocja!!!" somebody cried. Everybody began to run away in panic.

Police works pretty good in Byteland and first radiocars appeared near the embassy only few seconds after the explosion. All the people near the embassy were detained. Only one of the people is the organizer of the explosion, the others could by just occasional witnesses. It is known, among these M(1<=M<=11) people, N(1<=N<=M) people always lie because they want to make the work of police more difficult, the others always tell the truth.

All these people say P(1<=P<=30) sentences in total.All the useful sentences are in one of the 4 forms below, all the other sentences are useless and you can ignore them.

- I am guilty.
- I am not guilty.
- *Somebody* is guilty.
- *Somebody* is not guilty.
- Today is *Someday*.

Among these sentences, *Somebody* is a name of one of these M people, and *Someday* is Monday, Tuesday, Wednesday, Thursday, Friday, Saturday or Sunday.

Your task is to help the police to find out the only one organizer of the explosion.

Input

Multiple test cases. For each test case:

The first line contains three integers M, N and P. M lines follow, each contains a name of one of the people.All names contain only capital Latin letters and no whitespaces.P lines come next, each contains no more than 250 characters, the speaker and his/her speech. See the example.

You can assume that there’s no whitespace at the start and the end of a line, and there are no two consecutive spaces in the input.

Input terminate by EOF.

Output

The name of the organizer if you can determine, or *Cannot Determine* if you find more than one, or *Impossible* if you find no one.

Example

Input:

```
3 1 5
MIKE
CHARLES
KATE
MIKE: I am guilty.
MIKE: Today is Sunday.
CHARLES: MIKE is guilty.
KATE: I am guilty.
KATE: How are you??
```

Output:

```
MIKE
```

Added by: Blue Mary
Date: 2007-07-13
Time limit: 13s
Source limit: 50000B
Languages: All except: C99 strict
Resource: Description by Blue Mary

SPOJ Problem Set (classical)

1688. A Very Easy Problem!

Problem code: EASYPROB

Input

There's no input.

Output

Output some form of these numbers: 137, 1315, 73, 136, 255, 1384, 16385, one per line in the listed order.

Example

The first two lines of the CORRECT output file are:

```
137=2(2(2)+2+2(0))+2(2+2(0))+2(0)
1315=2(2(2+2(0))+2)+2(2(2+2(0)))+2(2(2)+2(0))+2+2(0)
```

The correct output file should contain 7 lines.

Added by: Blue Mary
Date: 2007-07-13
Time limit: 1s
Source limit:500B
Languages: TEXT
Resource: You can imagine it.

SPOJ Problem Set (classical)

1689. Hard Problem

Problem code: HARDP

[This space is intentionally left blank.]

Input

Multiple test cases. Each contains a single integer N(1<=N<=50). Input terminates by EOF.

Output

For each test case, output one line contains the answer. See the example.

Example

```
Input:
8
9
10


Output:
8=90*(0+0+0+0+45-3+20-42+60+10)
9=20*(0-3+10-14+15+2+0+0+0+0+10)
10=66*(0+0+0+0+0+33+5-33+66-66+55+6)
```

Added by: Blue Mary
Date: 2007-07-14
Time limit: 11s
Source limit:50000B
Languages: All except: C99 strict
Resource: Based on a problem from an ACM/ICPC Regional Contest

SPOJ Problem Set (classical)

1693. Coconuts

Problem code: COCONUTS

A group of n castle guards are voting to determine whether African swallows can carry coconuts. While each guard has his own personal opinion on the matter, a guard will often vote contrary to his beliefs in order to avoid disagreeing with the votes of his friends.
You are given a list of guards who either do or do not believe in the coconut-carrying capacity of African swallows, and a list of all pairs of guards who are friends. Your task is to determine how each guard must vote in order to minimize the sum of the total number of disagreements between friends and the total number of guards who must vote against their own beliefs.

Input

The input to this problem will contain multiple test cases. Each test case begins with a single line containing an integer n (where 2 <= n <= 300), the number of guards, and an integer m (where 1 <= m <= n(n-1)/2), the number of pairs of guards who are friends. The second line of the test case contains n integers, where the ith integer is 1 if the ith guard believes in the ability of African swallows to carry coconuts, and 0 otherwise. Finally, the next m lines of the test case each contain two distinct integers i and j (where 1 <= i, j <= n), indicating that guards i and j are friends. Guards within each pair of friends may be listed in any order, but no pair of guards will be repeated. The input is terminated by an invalid test case with n = m = 0, which should not be processed.

Output

For each input test case, print a single line containing the minimum possible sum of the total number of disagreements between all friends plus the total number of guards who must vote against their own beliefs.

Example

Input:

```
3 3
1 0 0
1 2
1 3
3 2
6 6
1 1 1 0 0 0
1 2
2 3
4 2
3 5
4 5
5 6
0 0
```

Output:

```
1
2
```

Added by: Jin Bin
Date: 2007-07-22
Time limit: 1s
Source limit: 5000B
Languages: All except: C99 strict
Resource: Pacific NW 2006

SPOJ Problem Set (classical)

1695. Grandpa’s Rubik Cube

Problem code: GRC

Documento sin título

A very well-known toy/pastime, called Rubik’s cube, consists of a cube as shown in Figure 1a, where letters stand for colors (e.g. B for blue, R for red,...). The goal of the game is to rotate the faces of the cube in such a way that at the end each face has a different color, as shown in Figure 1b. Notice that,

[IMAGE]

when a face is rotated, the configuration of colors in all the adjacent faces changes. Figure 2 illustrates a rotation of one of the faces. Given a scrambled configuration, reaching the final position can be quite challenging, as you may know.

[IMAGE]

But your grandpa has many years of experience, and claims that, given any configuration of the Rubik cube, he can come up with a sequence of rotations leading to a winning configuration. In order to show all faces of the cube we shall represent the cube as in Figure 3a. The six colors are Yellow, Red, Blue, Green, White and Magenta (represented by their first letters).
You will be given an initial configuration and a list of rotations. A rotation will be represented by an integer number, indicating the face to be rotated and the direction of the rotation (a positive value means clockwise rotation, negative value means counter-clockwise rotation). Faces of the cube are numbered as shown in Figure 3b. You must write a program that checks whether the list of rotations will lead to a winning configuration.

[IMAGE]

Input
The input contains several test cases. The first line of the input is an integer which indicates the number of tests. Each test description consists of ten lines of input. The first nine lines of a test will describe an initial configuration, in the format shown in Figure 3a. The next line will contain a list of rotations, ending with the value 0.

Output
For each test case your program should print one line. If your grandpa is correct, print "Yes, grandpa!", otherwise print "No, you are wrong!". (See example.)

[IMAGE]

Added by: Andrés Leonardo Rojas Duarte
Date: 2007-07-24
Time limit: 5s
Source limit: 50000B
Languages: All
Resource: ACM International Collegiate Programming Contest 2002 - South America

SPOJ Problem Set (classical)

1696. Will Indiana Jones Get There

Problem code: WIJGT

Documento sin título

Indiana Jones is in a deserted city, annihilated during a war. Roofs of all houses have been destroyed and only portions of walls are still standing. The ground is so full of mines that the only safe way to move around the city is walking over the remaining walls. The mission of our hero is to save a person who is trapped in the city. In order to move between two walls which are not connected Indiana Jones thought of taking with him a wooden board which he could place between the two walls and then cross from one to the other.

[IMAGE]

Initial positions of Indiana Jones and the trapped person are both on some section of the walls. Besides, walls are either in the direction South-North or West-East. You will be given a map of the city remains. Your mission is to determine the minimum length of the wooden board Indiana Jones needs to carry in order to get to the trapped person.

Input
Your program should process several test cases. Each test case starts with an integer N indicating the number of wall sections remaining in the city (2 <= N <= 1000). Each of the next N lines describes a wall section. The first wall section to appear is the section where Indiana Jones stands at the beginning. The second section to appear is the section where the trapped person stands. Each wall section description consists of three integers X , Y and L (-10000 <= X , Y , L <= 10000), where X an Y define either the southernmost point of a wall section (for South-North sections) or the westernmost point
(for West-East wall sections). The value of L determines the length and direction of the wall: if L > 0, the section is West-East, with length L ; if L< 0, the section is North-South, with length | L | . The end of input is indicated by N = 0.

Output
For each test case in the input your program should produce one line of output, containing a real value representing the length of the wooden board Indiana Jones must carry. The length must be printed as a real number with two-digit precision, and the last decimal digit must be rounded. The input will not contain test cases where differences in rounding are significant.

Sample input
14
1 1 5
6 8 2
7 2 -2
5 3 3
2 5 2
2 3 2
2 3 -2

4 3 -2
0 7 1
1 8 2
3 6 -2
4 7 2
6 6 1
6 6 -2
3
-10 0 20
-5 1 10
50 50 100
0

Output for the sample input

1.41
1.00

Added by: Andrés Leonardo Rojas Duarte
Date: 2007-07-25
Time limit: 10s
Source limit: 50000B
Languages: All
Resource: ACM International Collegiate Programming Contest 2002 - South America

SPOJ Problem Set (classical)

1697. Ohgas' Fortune

Problem code: OFORTUNE

The Ohgas are a prestigious family based on Hachioji. The head of the family, Mr. Nemochi Ohga, a famous wealthy man, wishes to increase his fortune by depositing his money to an operation company. You are asked to help Mr. Ohga maximize his profit by operating the given money during a specified period.

From a given list of possible operations, you choose an operation to deposit the given fund to. You commit on the single operation throughout the period and deposit all the fund to it. Each operation specifies an annual interest rate, whether the interest is simple or compound, and an annual operation charge. An annual operation charge is a constant not depending on the balance of the fund. The amount of interest is calculated at the end of every year, by multiplying the balance of the fund under operation by the annual interest rate, and then rounding off its fractional part. For compound interest, it is added to the balance of the fund under operation, and thus becomes a subject of interest for the following years. For simple interest, on the other hand, it is saved somewhere else and does not enter the balance of the fund under operation (i.e. it is not a subject of interest in the following years). An operation charge is then subtracted from the balance of the fund under operation. You may assume here that you can always pay the operation charge (i.e. the balance of the fund under operation is never less than the operation charge). The amount of money you obtain after the specified years of operation is called ‘‘the final amount of fund.’’ For simple interest, it is the sum of the balance of the fund under operation at the end of the final year, plus the amount of interest accumulated throughout the period. For compound interest, it is simply the balance of the fund under operation at the end of the final year.

Operation companies use C, C++, Java, etc., to perform their calculations, so they pay a special attention to their interest rates. That is, in these companies, an interest rate is always an integral multiple of 0.0001220703125 and between 0.0001220703125 and 0.125 (inclusive). 0.0001220703125 is a decimal representation of 1/8192. Thus, interest rates' being its multiples means that they can be represented with no errors under the double-precision binary representation of floating-point numbers.

For example, if you operate 1000000 JPY for five years with an annual, compound interest rate of 0.03125 (3.125 %) and an annual operation charge of 3000 JPY, the balance changes as follows.

The balance of the fund under operation(at the beginning of year)	Interest	The balance of the fund under operation (at the end of year)
A	B = A × 0.03125 (and rounding off fractions)	A + B - 3000
1000000	31250	1028250
1028250	32132	1057382
1057382	33043	1087425
1087425	33982	1118407
1118407	34950	1150357

After the five years of operation, the final amount of fund is 1150357 JPY.
If the interest is simple with all other parameters being equal, it looks like:

The balance of the fund under operation (at the beginning of year)	Interest	The balance of the fund under operation (at the end of year)	Cumulative interest
A	B = A × 0.03125 (and rounding off fractions)	A - 3000	
1000000	31250	997000	31250
997000	31156	994000	62406
994000	31062	991000	93468
991000	30968	988000	124436
988000	30875	985000	155311

In this case the final amount of fund is the total of the fund under operation, 985000 JPY, and the cumulative interests, 155311 JPY, which is 1140311 JPY.

Input

The input consists of datasets. The entire input looks like:

the number of datasets (=m)
1st dataset
2nd dataset
...
m-th dataset

The number of datasets, m, is no more than 100. Each dataset is formatted as follows.

the initial amount of the fund for operation
the number of years of operation
the number of available operations (=n)
operation 1
operation 2
...
operation n

The initial amount of the fund for operation, the number of years of operation, and the number of available operations are all positive integers. The first is no more than 100000000, the second no more than 10, and the third no more than 100.

Each ‘‘operation’’ is formatted as follows.

simple-or-compound annual-interest-rate annual-operation-charge
where simple-or-compound is a single character of either ’0’ or ’1’, with ’0’ indicating simple interest and ’1’ compound. annual-interest-rate is represented by a decimal fraction and is an integral multiple

of 1/8192. annual-operation-charge is an integer not exceeding 100000.

Output

For each dataset, print a line having a decimal integer indicating the final amount of fund for the best operation. The best operation is the one that yields the maximum final amount among the available operations. Each line should not have any character other than this number.

You may assume the final balance never exceeds 1000000000. You may also assume that at least one operation has the final amount of the fund no less than the initial amount of the fund.

Example

Input:

```
4
1000000
5
2
0 0.03125 3000
1 0.03125 3000
6620000
7
2
0 0.0732421875 42307
1 0.0740966796875 40942
39677000
4
4
0 0.0709228515625 30754
1 0.00634765625 26165
0 0.03662109375 79468
0 0.0679931640625 10932
10585000
6
4
1 0.0054931640625 59759
1 0.12353515625 56464
0 0.0496826171875 98193
0 0.0887451171875 78966
```

Output:

```
1150357
10559683
50796918
20829397
```

Added by: Camilo Andrés Varela León
Date: 2007-07-26
Time limit: 1s
Source limit: 50000B
Languages: All
Resource: Japan Domestic, 2005

SPOJ Problem Set (classical)

1698. Polygonal Line Search

Problem code: PLSEARCH

Multiple polygonal lines are given on the xy-plane. Given a list of polygonal lines and a template, you must find out polygonal lines which have the same shape as the template.

A polygonal line consists of several line segments parallel to x-axis or y-axis. It is defined by a list of xy-coordinates of vertices from the start-point to the end-point in order, and always turns 90 degrees at each vertex. A single polygonal line does not pass the same point twice. Two polygonal lines have the same shape when they fully overlap each other only with rotation and translation within xy-plane (i.e. without magnification or a flip). The vertices given in reverse order from the start-point to the end-point is the same as that given in order.

Figure 1 shows examples of polygonal lines. In this figure, polygonal lines A and B have the same shape.

Write a program that answers polygonal lines which have the same shape as the template.

[IMAGE]
Figure 1: Polygonal lines

Input

The input consists of multiple datasets. The end of the input is indicated by a line which contains a zero.

A dataset is given as follows.

n
Polygonal line0
Polygonal line1
Polygonal line2
...
Polygonal linen

n is the number of polygonal lines for the object of search on xy-plane. n is an integer, and 1 <= n <= 50. Polygonal line0 indicates the template.

A polygonal line is given as follows.

m
x1 y1
x2 y2
...
xm ym

m is the number of the vertices of a polygonal line (3 <= m <= 10). xi and yi, separated by a space, are the x- and y-coordinates of a vertex, respectively (-10000 < xi < 10000, -10000 <yi < 10000).

Output

For each dataset in the input, your program should report numbers assigned to the polygonal lines that have the same shape as the template, in ascending order. Each number must be written in a separate line without any other characters such as leading or trailing spaces.

Five continuous "+"s must be placed in a line at the end of each dataset.

Example

Input:

```
5
5
0 0
2 0
2 1
4 1
4 0
5
0 0
0 2
-1 2
-1 4
0 4
5
0 0
0 1
-2 1
-2 2
0 2
5
0 0
0 -1
2 -1
2 0
4 0
5
0 0
2 0
2 -1
4 -1
4 0
5
0 0
2 0
2 1
4 1
4 0
4
4
-60 -75
-60 -78
-42 -78
-42 -6
4
10 3
```

```
10 7
-4 7
-4 40
4
-74 66
-74 63
-92 63
-92 135
4
-12 22
-12 25
-30 25
-30 -47
4
12 -22
12 -25
30 -25
30 47
3
5
-8 5
-8 2
0 2
0 4
8 4
5
-3 -1
0 -1
0 7
-2 7
-2 16
5
-1 6
-1 3
7 3
7 5
16 5
5
0 1
0 -2
8 -2
8 0
17 0
0
```

Output :

```
1
3
5
+++++
3
4
+++++
+++++
```

Added by: Camilo Andrés Varela León
Date: 2007-07-26
Time limit: 3s
Source limit: 50000B
Languages: All
Resource: Japan Domestic, 2005

SPOJ Problem Set (classical)

1699. Numeral System

Problem code: NSYSTEM

Prof. Hachioji has devised a new numeral system of integral numbers with four lowercase letters "m", "c", "x", "i" and with eight digits "2", "3", "4", "5", "6", "7", "8", "9". He doesn't use digit "0" nor digit "1" in this system.

The letters "m", "c", "x" and "i" correspond to 1000, 100, 10 and 1, respectively, and the digits "2", ...,"9" correspond to 2, ..., 9, respectively. This system has nothing to do with the Roman numeral system.

For example, character strings

"5m2c3x4i", "m2c4i" and "5m2c3x"

correspond to the integral numbers 5234 (=5*1000+2*100+3*10+4*1), 1204 (=1000+2*100+4*1), and 5230 (=5*1000+2*100+3*10), respectively. The parts of strings in the above example, "5m", "2c", "3x" and "4i" represent 5000 (=5*1000), 200 (=2*100), 30 (=3*10) and 4 (=4*1), respectively.

Each of the letters "m", "c", "x" and "i" may be prefixed by one of the digits "2", "3", ..., "9". In that case, the prefix digit and the letter are regarded as a pair. A pair that consists of a prefix digit and a letter corresponds to an integer that is equal to the original value of the letter multiplied by the value of the prefix digit.

For each letter "m", "c", "x" and "i", the number of its occurrence in a string is at most one. When it has a prefix digit, it should appear together with the prefix digit. The letters "m", "c", "x" and "i" must appear in this order, from left to right. Moreover, when a digit exists in a string, it should appear as the prefix digit of the following letter. Each letter may be omitted in a string, but the whole string must not be empty. A string made in this manner is called an MCXI-string.

An MCXI-string corresponds to a positive integer that is the sum of the values of the letters and those of the pairs contained in it as mentioned above. The positive integer corresponding to an MCXI-string is called its MCXI-value. Moreover, given an integer from 1 to 9999, there is a unique MCXI-string whose MCXI-value is equal to the given integer. For example, the MCXI-value of an MCXI-string "m2c4i" is 1204 that is equal to 1000 + 2*100 + 4*1. There are no MCXI-strings but "m2c4i" that correspond to 1204. Note that strings "1m2c4i", "mcc4i", "m2c0x4i", and "2cm4i" are not valid MCXI-strings. The reasons are use of "1", multiple occurrences of "c", use of "0", and the wrong order of "c" and "m", respectively.

Your job is to write a program for Prof. Hachioji that reads two MCXI-strings, computes the sum of their MCXI-values, and prints the MCXI-string corresponding to the result.

Input

The input is as follows. The first line contains a positive integer n (<= 500) that indicates the number of the following lines. The k+1 th line is the specification of the k th computation (k=1, ..., n).

n
specification1
specification2
...
specificationn

Each specification is described in a line:

MCXI-string1 MCXI-string2

The two MCXI-strings are separated by a space.

You may assume that the sum of the two MCXI-values of the two MCXI-strings in each specification is less than or equal to 9999.

Output

For each specification, your program should print an MCXI-string in a line. Its MCXI-value should be the sum of the two MCXI-values of the MCXI-strings in the specification. No other characters should appear in the output.

Example

Input:

```
10
xi x9i
i 9i
c2x2i 4c8x8i
m2ci 4m7c9x8i
9c9x9i i
i 9m9c9x8i
m i
i m
m9i i
9m8c7xi c2x8i
```

Output:

```
3x
x
6cx
5m9c9x9i
m
9m9c9x9i
mi
mi
mx
9m9c9x9i
```

Added by: Camilo Andrés Varela León
Date: 2007-07-26
Time limit: 1s
Source limit: 50000B
Languages: All
Resource: Japan Domestic, 2005

SPOJ Problem Set (classical)

1700. Traveling by Stagecoach

Problem code: TRSTAGE

Once upon a time, there was a traveler.
He plans to travel using stagecoaches (horse wagons). His starting point and destination are fixed, but he cannot determine his route. Your job in this problem is to write a program which determines the route for him.
There are several cities in the country, and a road network connecting them. If there is a road between two cities, one can travel by a stagecoach from one of them to the other. A coach ticket is needed for a coach ride. The number of horses is specified in each of the tickets. Of course, with more horses, the coach runs faster.
At the starting point, the traveler has a number of coach tickets. By considering these tickets and the information on the road network, you should find the best possible route that takes him to the destination in the shortest time. The usage of coach tickets should be taken into account.
The following conditions are assumed.

- A coach ride takes the traveler from one city to another directly connected by a road. In other words, on each arrival to a city, he must change the coach.
- Only one ticket can be used for a coach ride between two cities directly connected by a road.
- Each ticket can be used only once.
- The time needed for a coach ride is the distance between two cities divided by the number of horses.
- The time needed for the coach change should be ignored.

Input

The input consists of multiple datasets, each in the following format. The last dataset is followed by a line containing five zeros (separated by a space).
n m p a b
t1 t2 ... tn
x1 y1 z1
x2 y2 z2
...
xp yp zp
Every input item in a dataset is a non-negative integer. If a line contains two or more input items, they are separated by a space.
n is the number of coach tickets. You can assume that the number of tickets is between 1 and 8. m is the number of cities in the network. You can assume that the number of cities is between 2 and 30. p is the number of roads between cities, which may be zero.
a is the city index of the starting city. b is the city index of the destination city. a is not equal to b. You can assume that all city indices in a dataset (including the above two) are between 1 and m.
The second line of a dataset gives the details of coach tickets. ti is the number of horses specified in the i-th coach ticket (1<=i<=n). You can assume that the number of horses is between 1 and 10.
The following p lines give the details of roads between cities. The i-th road connects two cities with city indices xi and yi, and has a distance zi (1<=i<=p). You can assume that the distance is between 1

and 100.
No two roads connect the same pair of cities. A road never connects a city with itself. Each road can be traveled in both directions.

Output

For each dataset in the input, one line should be output as specified below. An output line should not contain extra characters such as spaces.
If the traveler can reach the destination, the time needed for the best route (a route with the shortest time) should be printed. The answer should not have an error greater than 0.001. You may output any number of digits after the decimal point, provided that the above accuracy condition is satisfied.
If the traveler cannot reach the destination, the string "Impossible" should be printed. One cannot reach the destination either when there are no routes leading to the destination, or when the number of tickets is not sufficient. Note that the first letter of "Impossible" is in uppercase, while the other letters are in lowercase.

Example

Input:

```
3 4 3 1 4
3 1 2
1 2 10
2 3 30
3 4 20
2 4 4 2 1
3 1
2 3 3
1 3 3
4 1 2
4 2 5
2 4 3 4 1
5 5
1 2 10
2 3 10
3 4 10
1 2 0 1 2
1
8 5 10 1 5
2 7 1 8 4 5 6 3
1 2 5
2 3 4
3 4 7
4 5 3
1 3 25
2 4 23
3 5 22
1 4 45
2 5 51
1 5 99
0 0 0 0 0
```

Output:

```
30.000
3.667
Impossible
Impossible
2.856
```

```
Since the number of digits after the decimal point is
not specified, the above result is not the only
solution. For example, the following result is also acceptable.

30.0
3.66667
Impossible
Impossible
2.85595
```

Added by: Camilo Andrés Varela León
Date: 2007-07-26
Time limit: 3s
Source limit: 50000B
Languages: All
Resource: Japan Domestic, 2005

SPOJ Problem Set (classical)

1701. Earth Observation with a Mobile Robot Team

Problem code: EOWAMRT

A new type of mobile robot has been developed for environmental earth observation. It moves around on the ground, acquiring and recording various sorts of observational data using high precision sensors. Robots of this type have short range wireless communication devices and can exchange observational data with ones nearby. They also have large capacity memory units, on which they record data observed by themselves and those received from others.

Figure 1 illustrates the current positions of three robots A, B, and C and the geographic coverage of their wireless devices. Each circle represents the wireless coverage of a robot, with its center representing the position of the robot. In this figure, two robots A and B are in the positions where A can transmit data to B, and vice versa. In contrast, C cannot communicate with A or B, since it is too remote from them. Still, however, once B moves towards C as in Figure 2, B and C can start communicating with each other. In this manner, B can relay observational data from A to C. Figure 3 shows another example, in which data propagate among several robots instantaneously.

[IMAGE]
Figure 1: The initial configuration of three robots

[IMAGE]
Figure 2: Mobile relaying

[IMAGE]
Figure 3: Instantaneous relaying among multiple robots

As you may notice from these examples, if a team of robots move properly, observational data quickly spread over a large number of them. Your mission is to write a program that simulates how information spreads among robots. Suppose that, regardless of data size, the time necessary for communication is negligible.

Input

The input consists of multiple datasets, each in the following format.

N T R
nickname and travel route of the first robot
nickname and travel route of the second robot
...
nickname and travel route of the N-th robot

The first line contains three integers *N, T, and R* that are the number of robots, the length of the simulation period, and the maximum distance wireless signals can reach, respectively, and satisfy that $1 <= N <= 100$, $1 <= T <= 1000$, and $1 <= R <= 10$.

The *nickname and travel route* of each robot are given in the following format.

nickname
t0 x0 y0
t1 vx1 vy1
t2 vx2 vy2
...
tk vxk vyk

Nickname is a character string of length between one and eight that only contains lowercase letters. No two robots in a dataset may have the same nickname. Each of the lines following nickname contains three integers, satisfying the following conditions.

$0 = t_0 < t_1 < ... < t_k = T$
$-10 <= vx_1, vy_1, ..., vx_k, vy_k <= 10$

A robot moves around on a two dimensional plane. (x_0, y_0) is the location of the robot at time 0. From time t_{i-1} to t_i $(0 < i <= k)$, the velocities in the x and y directions are vx_i and vy_i, respectively. Therefore, the travel route of a robot is piecewise linear. Note that it may self-overlap or self-intersect.

You may assume that each dataset satisfies the following conditions.

- The distance between any two robots at time 0 is not exactly R.
- The x- and y-coordinates of each robot are always between -500 and 500, inclusive.
- Once any robot approaches within R + 10^{-6}. of any other, the distance between them will become smaller than R - 10^{-6}. while maintaining the velocities.
- Once any robot moves away up to R - 10^{-6}. of any other, the distance between them will become larger than R + 10^{-6}. while maintaining the velocities.
- If any pair of robots mutually enter the wireless area of the opposite ones at time t and any pair, which may share one or two members with the aforementioned pair, mutually leave the wireless area of the opposite ones at time t', the difference between t and t' is no smaller than 10^{-6}. time unit, that is, $|t - t'| >= 10^{-6}$.

A dataset may include two or more robots that share the same location at the same time. However, you should still consider that they can move with the designated velocities.

The end of the input is indicated by a line containing three zeros.

Output

For each dataset in the input, your program should print the nickname of each robot that have got until time *T* the observational data originally acquired by the first robot at time 0. Each nickname should be written in a separate line in dictionary order without any superfluous characters such as leading or trailing spaces.

Example

Input:

```
3 5 10
red
0 0 0
5 0 0
```

```
green
0 5 5
5 6 1
blue
0 40 5
5 0 0
3 10 5
atom
0 47 32
5 -10 -7
10 1 0
pluto
0 0 0
7 0 0
10 3 3
gesicht
0 25 7
5 -7 -2
10 -1 10
4 100 7
impulse
0 -500 0
100 10 1
freedom
0 -491 0
100 9 2
destiny
0 -472 0
100 7 4
strike
0 -482 0
100 8 3
0 0 0
```

Output:

```
blue
green
red
atom
gesicht
pluto
freedom
impulse
strike
```

Added by: Camilo Andrés Varela León
Date: 2007-07-26
Time limit: 2s
Source limit: 50000B
Languages: All
Resource: Japan Domestic, 2005

SPOJ Problem Set (classical)

1702. Cleaning Robot

Problem code: CLEANRBT

Here, we want to solve path planning for a mobile robot cleaning a rectangular room floor with furniture.

Consider the room floor paved with square tiles whose size fits the cleaning robot (1 × 1). There are 'clean tiles' and 'dirty tiles', and the robot can change a 'dirty tile' to a 'clean tile' by visiting the tile. Also there may be some obstacles (furniture) whose size fits a tile in the room. If there is an obstacle on a tile, the robot cannot visit it. The robot moves to an adjacent tile with one move. The tile onto which the robot moves must be one of four tiles (i.e., east, west, north or south) adjacent to the tile where the robot is present. The robot may visit a tile twice or more.

Your task is to write a program which computes the minimum number of moves for the robot to change all 'dirty tiles' to 'clean tiles', if ever possible.

Input

IThe input consists of multiple maps, each representing the size and arrangement of the room. A map is given in the following format.

w h
c11 c12 c13 ... c1w
c21 c22 c23 ... c2w
...
ch1 ch2 ch3 ... chw

The integers w and h are the lengths of the two sides of the floor of the room in terms of widths of floor tiles. w and h are less than or equal to 20. The character cyx represents what is initially on the tile with coordinates (x, y) as follows.

'.' : a clean tile
'*' : a dirty tile
'x' : a piece of furniture (obstacle)
'o' : the robot (initial position)

In the map the number of 'dirty tiles' does not exceed 10. There is only one 'robot'.

The end of the input is indicated by a line containing two zeros.

Output

For each map, your program should output a line containing the minimum number of moves. If the map includes 'dirty tiles' which the robot cannot reach, your program should output -1.

Example

Input:

```
7 5
.......
.o...*.
.......
.*...*.
.......
15 13
.......x.......
...o...x....*..
.......x.......
.......x.......
.......x.......
...............
xxxxx.....xxxxx
...............
.......x.......
.......x.......
.......x.......
..*....x....*..
.......x.......
10 10
..........
..o.......
..........
..........
..........
.....xxxxx
.....x....
.....x.*..
.....x....
.....x....
0 0
```

Output:

```
8
49
-1
```

Added by: Camilo Andrés Varela León
Date: 2007-07-26
Time limit: 5s
Source limit: 50000B
Languages: All
Resource: Japan Domestic, 2005

SPOJ Problem Set (classical)

1703. ACM (ACronymMaker)

Problem code: ACMAKER

The sadists who design problems for ACM programming contests often like to include the abbreviation "ACM" somewhere in their problem descriptions. Thus, in years past, the World Finals has had problems involving "Apartment Construction Management," the "Atheneum of Culture and Movies," the "Association of Cover Manufacturers," "ACM Airlines," the "Association for Computa- tional Marinelife," and even an insect named "Amelia Cheese Mite." However, the number of word combinations beginning with A, C, and M that make sense is finite and the problem writers are starting to run out of ideas (it's hard to think of problems about "Antidisestablishmentarianistic Chthonian Metalinguistics"). Fortunately, modern culture allows more flexibility in designing abbreviations -- consider, for example:

```
GDB -- Gnu DeBugger
LINUX -- either "LINus's UniX" or "LINUs's miniX" or "Linux Is Not UniX"
SNOBOL -- StriNg Oriented symBOlic Language
SPITBOL -- SPeedy ImplemenTation of snoBOL
```

The rules used in these examples seem to be:

- Insignificant words (such as "of", "a", "the", etc.) are ignored.
- The letters of the abbreviation must appear, in the correct order, as an ordered sublist of the letters in the significant words of the phrase to be abbreviated.
- At least one letter of the abbreviation must come from every significant word (multiple occurrences of a letter are, of course, treated as distinct).

Of course these rules are often broken in real life. For instance, RADAR is an abbreviation for "RAdio Detecting And Ranging". Under our rules (assuming that "and" is an insignificant word), this would not be a valid abbreviation (however, RADR or RADRAN or DODGING would be valid). You have been asked to take a list of insignificant words and a list of abbreviations and phrases and to determine in how many ways each abbreviation can be formed from the corresponding phrase according to the rules above.

Input

The input file consists of multiple scenarios. Each scenario begins with an integer $100 >= n >= 1$ followed by n insignificant words, all in lower case, one per line with no extra white space. (A line containing 0 indicates end of input.) Following this are one or more test cases for this scenario, one per line, followed by a line containing the phrase "LAST CASE". Each line containing a test case begins with an abbreviation (uppercase letters only) followed by a phrase (lowercase letters and spaces only). The abbreviation has length at least 1 and the phrase contains at least one significant word. No input line (including abbreviation, phrase, and spaces) will contain more than 150 characters. Within these limits, however, abbreviations and phrase words may be any length.

Output

For each test case, output the abbreviation followed by either

is not a valid abbreviation

or

can be formed in i ways

where *i* is the number of different ways in which the letters of the abbreviation may be assigned to the letters in the phrase according to the rules above. The value of *i* will not exceed the range of a 32-bit signed integer.

Example

Input:

```
2
and
of
ACM academy of computer makers
RADAR radio detection and ranging
LAST CASE
2
a
an
APPLY an apple a day
LAST CASE
0
```

Output:

```
ACM can be formed in 2 ways
RADAR is not a valid abbreviation
APPLY can be formed in 1 ways
```

Added by: Camilo Andrés Varela León
Date: 2007-07-26
Time limit: 1s
Source limit: 50000B
Languages: All
Resource: East Central North America 2005

SPOJ Problem Set (classical)

1704. Countdown

Problem code: CDOWN

Ann Sister owns a genealogical database service, which maintains family tree history for her clients. When clients login to the system, they are presented with a variety of services: searching, printing, querying, etc. One recent question that came up which the system was not quite prepared for was the following: "Which member of my family had the most grandchildren?" The client who posed this question eventually had to answer it by manually searching the family tree database herself. Ann decided to have software written in case this question (or ones similar to it asking for great-grandchildren, or great-great-grandchildren, etc.) is asked in the future.

Input

Input will consist of multiple test cases. The first line of the input will contain a single integer indicating the number of test cases. Each test case starts with a single line containing two positive integers n and d, where n indicates the number of lines to follow containing information about the family tree, and d indicates the specific question being asked about the tree: if d = 1, then we are interested in persons with the most children (1 generation away); if d = 2, then we are interested in persons with the most grandchildren (2 generations away), and so on. The next n lines are of the form

name m $dname_1$ $dname_2$ ... $dname_m$

where *name* is one of the family members' names, *m* is the number of his/her children, and $dname_1$ through $dname_m$ are the names of the children. These lines will be given in no particular order. You may assume that all n lines describe one single, connected tree. There will be no more than 1000 people in any one tree, and all names will be at most 10 characters long.

Output

For each test case, output the three names with the largest number of specified descendants in order of number of descendants. If there are ties, output the names within the tie in alphabetical order. Print fewer than three names if there are fewer than three people who match the problem criteria (you should not print anyone's name who has 0 of the specified descendants), and print more than three if there is a tie near the bottom of the list. Print each name one per line, followed by a single space and then the number of specified descendants. The output for each test case should start with the line

Tree *i*:

where *i* is the test case number (starting at 1). Separate the output for each problem with a blank line.

Example

Input:

```
3
8 2
Barney 2 Fred Ginger
Ingrid 1 Nolan
Cindy 1 Hal
Jeff 2 Oliva Peter
Don 2 Ingrid Jeff
Fred 1 Kathy
Andrea 4 Barney Cindy Don Eloise
Hal 2 Lionel Mary
6 1
Phillip 5 Jim Phil Jane Joe Paul
Jim 1 Jimmy
Phil 1 Philly
Jane 1 Janey
Joe 1 Joey
Paul 1 Pauly
6 2
Phillip 5 Jim Phil Jane Joe Paul
Jim 1 Jimmy
Phil 1 Philly
Jane 1 Janey
Joe 1 Joey
Paul 1 Pauly
```

Output:

```
Tree 1:
Andrea 5
Don 3
Cindy 2
Tree 2:
Phillip 5
Jane 1
Jim 1
Joe 1
Paul 1
Phil 1
Tree 3:
Phillip 5
```

Added by: Camilo Andrés Varela León
Date: 2007-07-26
Time limit: 3s
Source limit: 50000B
Languages: All
Resource: East Central North America 2005

SPOJ Problem Set (classical)

1705. The Game of Efil

Problem code: GAMEFIL

Almost anyone who has ever taken a class in computer science is familiar with the "Game of Life," John Conway's cellular automata with extremely simple rules of birth, survival, and death that can give rise to astonishing complexity.

The game is played on a rectangular field of cells, each of which has eight neighbors (adjacent cells). A cell is either occupied or not. The rules for deriving a generation from the previous one are:

- If an occupied cell has 0, 1, 4, 5, 6, 7, or 8 occupied neighbors, the organism dies (0, 1: of loneliness; 4 thru 8: of overcrowding).
- If an occupied cell has two or three occupied neighbors, the organism survives to the next generation.
- If an unoccupied cell has three occupied neighbors, it becomes occupied (a birth occurs).

One of the major problems researchers have looked at over the years is the existence of so-called "Garden of Eden" configurations in the Game of Life -- configurations that could not have arisen as the result of the application of the rules to some previous configuration. We're going to extend this question, which we'll call the "Game of Efil": Given a starting configuration, how many possible parent configurations could it have? To make matters easier, we assume a finite grid in which edge and corner cells "wrap around" (i.e., a toroidal surface). For instance, the 2 by 3 configuration:

[IMAGE]

has exactly three possible parent configurations; they are:

[IMAGE] [IMAGE] [IMAGE]

You should note that when counting neighbors of a cell, another cell may be counted as a neighbor more than once, if it touches the given cell on more than one side due to the wrap around. This is the case for the configurations above.

Input

There will be multiple test cases. Each case will start with a line containing a pair of positive integers m and n, indicating the number of rows and columns of the configuration, respectively. The next line will contain a nonnegative integer k indicating the number of "live" cells in the configuration. The following k lines each contain the row and column number of one live cell, where row and column numbering both start at zero. The final test case is followed by a line where $m = n = 0$ -- this line should not be processed. You may assume that the product of m and n is no more than 16.

Output

For each test case you should print one line of output containing the case number and the number of possible ancestors. Imitate the sample output below. Note that if there are 0 ancestors, you should print out

`Garden of Eden.`

Example

Input:

```
2 3
2
0 0
0 1
3 3
4
0 0
0 1
0 2
1 1
3 3
5
0 0
1 0
1 2
2 1
2 2
0 0
```

Output:

```
Case 1: 3 possible ancestors.
Case 2: 1 possible ancestors.
Case 3: Garden of Eden.
```

Added by: Camilo Andrés Varela León
Date: 2007-07-26
Time limit: 60s-65s
Source limit:50000B
Languages: All
Resource: East Central North America 2005

SPOJ Problem Set (classical)

1706. Queens, Knights and Pawns

Problem code: QKP

You all are familiar with the famous 8-queens problem which asks you to place 8 queens on a chess board so no two attack each other. In this problem, you will be given locations of queens and knights and pawns and asked to find how many of the unoccupied squares on the board are not under attack from either a queen or a knight (or both). We'll call such squares "safe" squares. Here, pawns will only serve as blockers and have no capturing ability. The board below has 6 safe squares. (The shaded squares are safe.)

[IMAGE]

Recall that a knight moves to any unoccupied square that is on the opposite corner of a 2x3 rectangle from its current position; a queen moves to any square that is visible in any of the eight horizontal, vertical, and diagonal directions from the current position. Note that the movement of a queen can be blocked by another piece, while a knight's movement can not.

Input

There will be multiple test cases. Each test case will consist of 4 lines. The first line will contain two integers n and m, indicating the dimensions of the board, giving rows and columns, respectively. Neither integer will exceed 1000. The next three lines will each be of the form

$k\ r_1\ c_1\ r_2\ c_2 \ldots r_k\ c_k$

indicating the location of the queens, knights and pawns, respectively. The numbering of the rows and columns will start at one. There will be no more than 100 of any one piece. Values of n = m = 0 indicate end of input.

Output

Each test case should generate one line of the form

Board b has s safe squares.

where *b* is the number of the board (starting at one) and you supply the correct value for *s*.

Example

```
4 4
2 1 4 2 4
1 1 2
1 2 3
2 3
1 1 2
1 1 1
0
1000 1000
```

```
1 3 3
0
0
0 0
```

Output:

```
Board 1 has 6 safe squares.
Board 2 has 0 safe squares.
Board 3 has 996998 safe squares.
```

Added by: Camilo Andrés Varela León
Date: 2007-07-26
Time limit: 1s
Source limit: 50000B
Languages: All
Resource: East Central North America 2005

SPOJ Problem Set (classical)

1707. Reliable Nets

Problem code: RELINETS

You're in charge of designing a campus network between buildings and are very worried about its reliability and its cost. So, you've decided to build some redundancy into your network while keeping it as inexpensive as possible. Specifically, you want to build the cheapest network so that if any one line is broken, all buildings can still communicate. We'll call this a *minimal reliable net*.

Input

There will be multiple test cases for this problem. Each test case will start with a pair of integers n (<=15) and m (<= 20) on a line indicating the number of buildings (numbered 1 through n) and the number of potential inter-building connections, respectively. (Values of n = m = 0 indicate the end of the problem.) The following m lines are of the form b_1 b_2 c (all positive integers) indicating that it costs c to connect building b_1 and b_2. All connections are bidirectional.

Output

For each test case you should print one line giving the cost of a minimal reliable net. If there is a minimal reliable net, the output line should be of the form:

The minimal cost for test case p is c.

where *p* is the number of the test case (starting at 1) and *c* is the cost. If there is no reliable net possible, output a line of the form:

There is no reliable net possible for test case p.

Example

Input:

```
4 5
1 2 1
1 3 2
2 4 2
3 4 1
2 3 1
2 1
1 2 5
0 0
```

Output:

```
The minimal cost for test case 1 is 6.
There is no reliable net possible for test case 2.
```

Added by: Camilo Andrés Varela León
Date: 2007-07-26
Time limit: 3s
Source limit: 50000B
Languages: All
Resource: East Central North America 2005

SPOJ Problem Set (classical)

1708. Square Count

Problem code: SQCOUNT

Little Bobby Roberts, age 8, has been dragged to yet another museum by his parents. While they while away the hours studying Etruscan pottery and Warhol soup cans, Bobby must depend on himself for entertainment. Having a mathematical bent, he recently started counting all the square tiles on the floors of the museum. He soon realized that the tiles could be grouped into larger squares that needed to be added to the count. The problem became a bit more complicated when he started counting squares contained in multiple rooms, since some squares overlapped both rooms. For example, the two rooms shown below contain a total of 86 squares: 45 1 × 1 squares, 28 2 × 2 squares and 13 3 × 3 squares. (Note the opening between the two rooms is only 3 squares wide.)

[IMAGE]

While this helped kill several days' worth of museum visits, it soon became rather tedious, so Bobby is now looking for a program to automate the counting process for him.

Input

Input will consist of multiple test cases. The first line of each case will be a positive integer n <= 1000 which will indicate the number of rooms in the museum. After this will be n lines, each containing a description of one room. Each room will be rectangular in shape and will be described by a line of the form

x_1 y_1 x_2 y_2

where (x_1, y_1) and (x_2, y_2) are opposing corner coordinates (integers) of the room. No two rooms will overlap, though they may share a side. If the shared side is of length m > 2, then a door of length m-2 exists between the two rooms, centered along the shared length. No square of any size will overlap more than two rooms. All *x* and *y* values will be <= 1, 000, 000. An input line of n = 0 terminates input and should not be processed.

Output

For each test case, output the total number of squares on a single line in the format shown below. All answers will fit within a 32-bit integer and cases are enumerated starting at 1.

Example

Input:

```
2
0 0 9 3
10 6 4 3
3
11 20 15 24
11 17 15 20
15 16 20 24
```

```
0
```

Output:

```
Case 1: 86
Case 2: 152
```

Added by: Camilo Andrés Varela León
Date: 2007-07-26
Time limit: 1s
Source limit: 50000B
Languages: All
Resource: East Central North America 2005

SPOJ Problem Set (classical)

1709. Swamp Things

Problem code: SWTHIN

Hugh F. Oh, in his never-ending quest to prove the existence of extraterrestrials, has gotten hold of a number of nighttime photographs taken by a research group that is examining glowing swamp gas. Hugh wants to see if any of the photos show, not swamp gas, but Little Grey Men in glowing suits. The photographs consist of bright dots appearing against a black background. Unfortunately, at the time the photos were taken, trains were travelling through the area (there is a train trestle over the swamp), and occasional lights from the train windows also appear in the photographs. Hugh, being a fastidious researcher, wants to eliminate these spots from the images. He can't tell from the photos exactly where the tracks are, or from what direction the photos were taken, but he knows that the tracks in that area are perfectly straight, so he's decided on the following approach: he will find the line with the maximum number of spots lying on it and, if there are four or more spots on the line, he will eliminate those points from his calculations, assuming that those are windows on the train. If two or more lines have the maximum number of points, Hugh will just randomly select one such set and delete it from the photo (he's not all that fastidious - after all, he believes in Little Grey Men). If there are fewer than four points lying along a common line, Hugh will assume that there is no train in the photograph and won't delete any points. Please write a program for him to process a set of photographs.

Input

There will be a series of test cases. Each test case is one photograph described by a line containing a positive integer n (<= 1000) the number of distinct spots in the photograph, followed by n lines containing the integer coordinates of the spots, one (x, y) pair per line. All coordinates are between 0 and 10000. The last photo description is followed by a line containing a zero, marking the end of the input. This line should not be processed.

Output

For each test case, output the photo number followed by the number of points eliminated from the photograph. Imitate the sample output below.

Example

Input:

```
6
0 1
0 2
1 2
2 2
4 5
5 6
4
3 5
4 4
6 5
```

```
7 4
0
```

Output:

```
Photo 1: 4 points eliminated
Photo 2: 0 points eliminated
```

Added by: Camilo Andrés Varela León
Date: 2007-07-26
Time limit: 25s
Source limit: 50000B
Languages: All
Resource: East Central North America 2005

SPOJ Problem Set (classical)

1710. Two Ends

Problem code: TWENDS

In the two-player game "Two Ends", an even number of cards is laid out in a row. On each card, face up, is written a positive integer. Players take turns removing a card from either end of the row and placing the card in their pile. The player whose cards add up to the highest number wins the game. Now one strategy is to simply pick the card at the end that is the largest -- we'll call this the greedy strategy. However, this is not always optimal, as the following example shows: (The first player would win if she would first pick the 3 instead of the 4.)

3 2 10 4

You are to determine exactly how bad the greedy strategy is for different games when the second player uses it but the first player is free to use any strategy she wishes.

Input

There will be multiple test cases. Each test case will be contained on one line. Each line will start with an even integer n followed by n positive integers. A value of $n = 0$ indicates end of input. You may assume that n is no more than 1000. Furthermore, you may assume that the sum of the numbers in the list does not exceed 1,000,000.

Output

For each test case you should print one line of output of the form:

In game m, the greedy strategy might lose by as many as p points.

where m is the number of the game (starting at game 1) and p is the maximum possible difference between the first player's score and second player's score when the second player uses the greedy strategy. When employing the greedy strategy, always take the larger end. If there is a tie, remove the left end.

Example

Input:

```
4 3 2 10 4
8 1 2 3 4 5 6 7 8
8 2 2 1 5 3 8 7 3
0
```

Output:

```
In game 1, the greedy strategy might lose by as many as 7 points.
In game 2, the greedy strategy might lose by as many as 4 points.
In game 3, the greedy strategy might lose by as many as 5 points.
```

Added by: Camilo Andrés Varela León
Date: 2007-07-26
Time limit: 1s
Source limit: 50000B
Languages: All
Resource: East Central North America 2005

SPOJ Problem Set (classical)

1712. Permalex

Problem code: PRMLX

Given a string of characters, we can permute the individual characters to make new strings. If we can impose an ordering on the characters (say alphabetic sequence), then the strings themselves can be ordered and any given permutation can be given a unique number designating its position in that ordering. For example the string ‘acab’ gives rise to the following 12 distinct permutations:

aabc	1	acab	5	bcaa	9
aacb	2	acba	6	caab	10
abac	3	baac	7	caba	11
abca	4	baca	8	cbaa	12

Thus the string ‘acab’ can be characterised in this sequence as 5.

Write a program that will read in a string and determine its position in the ordered sequence of permutations of its constituent characters. Note that numbers of permutations can get very large; however we guarantee that no string will be given whose position is more than 2^31 = 2.147.483.647

Input and Output

Input will consist of a series of lines, each line containing one string. Each string will consist of up to 30 lower case letters, not necessarily distinct. Test cases a separated by empty lines. The file will be terminated by a line consisting of a single #.

Output will consist of a series of lines, one for each line of the input. Each line will consist of the position of the string in its sequence, right justified in a field of width 10.

Sample Input

```
bacaa
abc
cba

bacaa
abc
cba
#
```

Sample Output

```
15
 1
 6

15
 1
 6
```

Added by:	Andrés Leonardo Rojas Duarte
Date:	2007-07-27
Time limit:	5s
Source limit:	50000B
Languages:	All
Resource:	ACM ICPC 1992 Northwestern European Regionals

SPOJ Problem Set (classical)

1713. Funny scales

Problem code: SCALE

Kinh_Can has a set of precious weights P_1, P_2, ..., P_N in which the mass of the i^{th} weight is $P_i = 3^{i-1}$, and a balance with 2 scales. On a nice day, Kinh_Can decided to show off his set of precious weights to his friends, and said that he can put them in equilibrium with any weight as long as its mass is not more than the mass of the sum of his weights. At first, his friends didn't believe, but after many trials they realized that Kinh_Can was right. In addition, while putting a thing whose mass is X on a scale, Kinh_Can could put right away the weights added on the 2 scales to keep their blance without any trial. With a random weight X (X is a natural number, X != 0). Your task is to put weights on scales in order to keep the 2 scales' balance like Kinh_Can. The first scale initially weights X, and the second one weights 0.

Input

Input has exactly one line consisting 2 numbers, the first is N and the second is X.

Output

- If there is no solution, you should write -1
- If there is at least one solution for the problem, you should write exactly 2 lines:
  - The first line contains some numbers descripting the indices of the weights in the first disc
  - The second line contains some numbers description the indices of the weights in the second disc
  - **Note:** One of 2 lines can be blank

Constraints

- 1 <= N <= 20
- 1 <= X <= 2000000000

Example

Input 1:
```
10 2
```

Output 1:
```
1
2
```

Input 2:
```
10 5
```

Output 2:
```
1 2
3
```

Added by: Nguye^~n Ha Du+o+ng
Date: 2007-07-29
Time limit: 1s
Source limit:50000B
Languages: All
Resource: Classical

SPOJ Problem Set (classical)

1715. Another Necklace Problem

Problem code: NCKLCE

T Corporation is a company which produces colorful necklaces. The necklaces designed by them are unique and fashionable, and because of the price, they are popular with the youth. Now, T Corporation intends to design a self-help Producing System.
This system includes hardware and software. The software is interactive and controls the hardware. Now the hardware has been completed, but the software is to develop. The workers find you, who is taking NOI. Could you please write a software system to simulate?
A necklace includes N beads. The color of each bead is one of 1..c. The necklace is fixed in a plain. One position of the plain is marked as Position 1, and the other positions are marked as 2..n in clockwise.
Your system should supply the orders as follow:

Order	Parameters restrictions	Content
R k	0 < k < N	It means Rotate K. Rotate the necklace by k positions in clockwise. i.e. The bead in former 1 position will be transfer to position k+1, the bead in former 2 position will be transfer to position k+2, and so on.
F		It means Flip. Flip the plain by the given axis. The bead in position 1 doesn’t move.The bead in position 2 will swap with the bead in position N,the bead in position 3 will swap with the bead in position n-1, and so on.
S i j	1 <= i,j <= n	Swap the bead in position i and j.
P i j x	1 <= i,j <= n , x<= c	It means Paint. Paint color x from position i to position j in clockwise.
C		It means Count. Ask how many parts are there in the necklace. We define some consecutive beads in same color as a "part". Pay attention that C is different from CS 1,n.
CS i j	1 <= i,j <= n	It means CountSegment i,j. Ask how many parts are there from position i to position j in clockwise, i and j included.

Input

The first line in input includes two integers N,C, representing the beads in the necklace and the number of colors. The second line contains N integers x1,x2...xn, representing the colors of beads from position 1 to position n,1<=xi<=c. The third line includes a integer q, as the number of orders. There is an order in the next q lines, as mentioned above.
For 60% test cases, n <= 1000, Q <= 1000;
For 100% test cases, n <= 500000, Q <= 500000.

Output

For every order starts with C and CS, print a integer as the answer.

Example

Input:

```
5 3
1 2 3 2 1
4
C
R 2
P 5 5 2
CS 4 1
```

Output:

```
4
1
```

Test data is unofficial. If you have any questions, please contact me.

Added by: Jin Bin
Date: 2007-08-03
Time limit: 0.100s-5s
Source limit:50000B
Languages: All except: C99 strict
Resource: Chinese National Olympiad in Informatics 2007,Day 2

SPOJ Problem Set (classical)

1716. Can you answer these queries III

Problem code: GSS3

You are given a sequence A of N (N <= 50000) integers between -10000 and 10000. On this sequence you have to apply M (M <= 50000) operations:
modify the i-th element in the sequence or for given x y print max{Ai + Ai+1 + .. + Aj | x<=i<=j<=y }.

Input

The first line of input contains an integer N. The following line contains N integers, representing the sequence A1..AN.
The third line contains an integer M. The next M lines contain the operations in following form:
0 x y: modify Ax into y (|y|<=10000).
1 x y: print max{Ai + Ai+1 + .. + Aj | x<=i<=j<=y }.

Output

For each query, print an integer as the problem required.

Example

Input:
```
4
1 2 3 4
4
1 1 3
0 3 -3
1 2 4
1 3 3
```

Output:
```
6
4
-3
```

Added by: Jin Bin
Date: 2007-08-03
Time limit: 1s
Source limit:5000B
Languages: All except: C99 strict
Resource: own problem

SPOJ Problem Set (classical)

1722. Life, the Universe, and Everything II

Problem code: RP

This problem tests your mathematic knowledge and your programming ability very much. Your task is to calculate the number of different Minimum Spanning Trees (MSTs) of a special undirected unweighted graph. The graph has n nodes numbered from 1 to n, and there is an edge between node i (1<=i<=n) and node j (1<=j<=n) if and only if 0<|i-j|<=k.

Input

Multiple test cases, the number of them(<=8) is given in the very first line.

Each test case contains one line with two space-separated numbers k(1<=k<=5) and n(1<=n<=10^{15}).

Output

For each test case you should output one line, the number of different MSTs of the corresponding graph modulo 65521.

Example

Input:
```
1
3 5
```

Output:
```
75
```

Added by: Blue Mary
Date: 2007-08-04
Time limit: 1s-4s
Source limit:50000B
Languages: All except: C99 strict
Resource: Chinese National Olympiad in Informatics 2007,Day 2; Translated by Blue Mary

SPOJ Problem Set (classical)

1723. Bee Maja

Problem code: BMJ

Documento sin título

Maja is a bee. She lives in a bee hive with thousands of other bees. This bee hive consists of many hexagonal honey combs where the honey is stored in.
But bee Maja has a problem. Willi told her where she can meet him, but because Willi is a male drone and Maja is a female worker they have different coordinate systems.

Maja's Coordinate System	**Willi's Coordinate System**
Maja who often flies directly to a special honey comb has laid an advanced two dimensional grid over the whole hive.	Willi who is more lazy and often walks around just numbered the cells clockwise starting from 1 in the middle of the hive.
Maja	Willi

Help Maja to convert Willi's system to hers. Write a program which for a given honey comb number gives the coordinates in Maja's system.

Input

The input contains one or more integers which represent Willi's numbers. Each number stands on its own in a separate line, directly followed by a newline. The honey comb numbers are all less than 100 000.

Output

You should output the corresponding Maja coordinates to Willi's numbers, each coordinate pair on a separate line.

Sample Input

```
1
2
3
4
5
```

Sample Output

```
0 0
0 1
-1 1
-1 0
0 -1
```

Added by: Andrés Leonardo Rojas Duarte
Date: 2007-08-04
Time limit: 20s
Source limit:50000B
Languages: All
Resource: Ulm Local 1999

SPOJ Problem Set (classical)

1724. Counting Triangles

Problem code: TRICOUNT

We define the LEVEL of a triangle as in the following illustrative image:
[IMAGE]
Task: Your task is very easy. All you have to do is to count all triangles in the biggest one (Level N).

Input

The first line of the input contains an integer T (T <= 10000) - the number of test cases and T lines follow. Each line contains an integer N (1 <= N <= 10^6) which is the level of the triangle in that test case.

Output

For each test case, you should write a seperate line: the number of triangles in the biggest one (Level N). (All answers will fit within the range of a 64-bit integer)

Example

Input:
```
3
1
2
3
```

Output:
```
1
5
13
```

Added by: Nguye^~n Ha Du+o+ng
Date: 2007-08-05
Time limit: 1s
Source limit:500B
Languages: All
Resource: Trân Huy Hu+ng

SPOJ Problem Set (classical)

1725. The Importance

Problem code: IMPORT1

Given an undirected weighted graph $\{V,E\}$. Your task to calculate the importance of each node.

The importance of a node *v* (I(v)) can be defined as follow:

[IMAGE]

$C_{s,t}$ is the number of different shortest paths from s to t, $C_{s,t}(v)$ is the number of different shortest paths from s to t through v.

Input

Multiple test cases, the number of them is given in the very first line.

For each test case:

The first line contains two space-separated integers n(n<=100) and m(m<=4500), the number of nodes in the graph and the number of edges in the graph. The nodes are numbered from 1 to n. m lines follow, each contains 3 integers a, b, c, 1<=a, b<=n, 1<=c<=1000, a!=b, which denotes that there is an undirected edge between node a and node b weighted c. You may assume that there is at most one edge between any pair of nodes, and the number of shortest paths between any pair of nodes is at least 1 and at most 10^{10}.

Output

For each test case:

Your Output should contains n lines, each contains one single real number, with 3 decimal places after radix point. The number in the *i*th line denotes the importance of the *i*th node.

Example

Input:
```
1
4 4
1 2 1
2 3 1
3 4 1
4 1 1
```

Output:
```
1.000
1.000
1.000
1.000
```

Added by: Blue Mary
Date: 2007-08-05
Time limit: 1s
Source limit:50000B
Languages: All except: C99 strict
Resource: Chinese National Olympiad in Informatics 2007,Day 1; Translated by Blue Mary

SPOJ Problem Set (classical)

1726. Exchange

Problem code: EXCHANGE

There are 3 kinds of money in a planet far away from the earth: Mone, Luck, and Rpin. There's a money exchange company in this planet. You must go to this company if you want to do some money exchange, and, more autocratically, this company regulate the exchange rate of each pair of these 3 kinds of money.

The money exchange will be done in the following two ways:

(A)

You give the company a real number x in the range (0,100], the company will exchange x% of your Mone and x% of your Luck to equal Rpin according to the exchange rate of that day.

(B)

You give the company a real number x, the company will exchange your x Rpin to some Mone and Luck, whose value is equal to x Rpin according to the exchange rate of that day, and, the value of Mone is *Rate* times of the value of Luck.

You can do many exchange operations in the same day.

Now, as the excellant spy in this planet, you know the exchange rate between Mone and Rpin of each of the next n days(a_i Mone per Rpin), and the exchange rate between Luck and Rpin of each of the next n days(b_i Luck per Rpin), and, each Rate of the next n days($Rate_i$). you have S Rpin in the start, and you want to get most Rpin in the nth day later.

Input

Multiple test cases, the number of them(<=5) is given in the very first line.

For each test case:

The first line contains a integer number n(1<=n<=100000) and a real number S.n lines follow, each contains 3 real numbers: a_i (between 0 and 10), b_i (between 0 and 10), $Rate_i$ (between 0 and 100).

Output

For each test case, output one line contains a real number with 3 digits after decimal point, which denotes to the answer. You can assume it is less than 1000000000.

Example

Input:
```
1
3 100
1 1 1
1 2 2
2 2 3
```

Output:
```
225.000
```

Warning: large input/output data, be careful with certain languages; the time limit is somewhat strict for this problem

---

Added by: Blue Mary
Date: 2007-08-05
Time limit: 2s
Source limit: 50000B
Languages: All except: C99 strict
Resource: Chinese National Olympiad in Informatics 2007,Day 1; Translated by Blue Mary

SPOJ Problem Set (classical)

1728. Common Permutation

Problem code: CPRMT

Given two strings of lowercase letters, ***a*** and ***b***, print the longest string ***x*** of lowercase letters such that there is a permutation of ***x*** that is a subsequence of ***a*** and there is a permutation of ***x*** that is a subsequence of ***b***.

Input

Input file contains several lines of input. Consecutive two lines make a set of input. That means in the input file line **1** and **2** is a set of input, line **3** and **4** is a set of input and so on. The first line of a pair contains ***a*** and the second contains ***b***. Each string is on a separate line and consists of at most **1000** lowercase letters.

Output

For each set of input, output a line containing ***x***. If several ***x*** satisfy the criteria above, choose the first one in alphabetical order.

Example

Sample input:

```
pretty
women
walking
down
the
street
```

Sample output:

```
e
nw
et
```

Added by: Andrés Leonardo Rojas Duarte
Date: 2007-08-05
Time limit: 1s
Source limit:50000B
Languages: All
Resource: University of Alberta Local Contest 1999

SPOJ Problem Set (classical)

1730. Counting Triangles II

Problem code: TCOUNT2

We define the LEVEL of a triangle as in the following illustrative image:

[IMAGE]

And we continue defining the LEVEL of a hexagon. It's called level N hexagon if it's joined by 6 triangles, each one is a level N triangle.

[IMAGE]

Task: All you have to do is to count all triangles in the "level N" hexagon.

Input

The first line of the input contains an integer T - the number of test cases and T lines follow. Each line contains an integer N which is the level of the hexagon in that test case.

Output

For each test case, you should write a seperate line: the number of triangles in the "level N" hexagon. (All answers will fit within the range of a 64-bit positive integer)

Example

Input:
```
1
1
```

Output:
```
6
```

Be careful with certain languages

Added by: Nguye^~n Ha Du+o+ng
Date: 2007-08-06
Time limit: 1s
Source limit:300B
Languages: C C99 strict C++ PAS gpc PAS fpc ASM D
Resource: Trân Huy Hu+ng

SPOJ Problem Set (classical)

1731. Counting Triangles III

Problem code: TCOUNT3

Have you felt bored when counting triangles?

Like TRICOUNT and TCOUNT2, we define the LEVEL of a triangle as in the following illustrative image:

[IMAGE]

And now we will continue defining the LEVEL of a hexagram. It's called level N hexagram if it's joined by 12 triangles, each one is a level N triangle.

[IMAGE]

Task: All you have to do is to count all triangles in the level N hexagram.

Input

The first line of the input contains an integer T - the number of test cases and T lines follow. Each line contains an integer N which is the level of the hexagram in that test case.

Output

For each test case, you should write a seperate line: the number of triangles in the level N hexagram. (All answers will fit within the range of a 64-bit positive integer)

Example

Input:
```
1
1
```

Output:
```
20
```

The author allows only few languages

Added by: Nguye^~n Ha Du+o+ng
Date: 2007-08-07
Time limit: 1s
Source limit: 128B
Languages: C C99 strict C++ PAS gpc PAS fpc ASM D PERL PYTH RUBY
Resource: Trân Huy Hu+ng

SPOJ Problem Set (classical)

1739. Yet Another Equation

Problem code: EQU2

Consider the equation

$$x^2 - ny^2 = 1$$

where n is some integer.

Find the smallest strictly positive integer solutions (x, y) for a given n.

Input

The number of test cases t (around 30), followed by a list of t values of n ($2 <= n <= 1000$). You can assume that the equation can be solved for all values of n in the input set.

Output

For every test case, the values of x and y separated by a space character, on separate lines.

Example

Input:
```
3
2
6
61
```

Output:
```
3 2
5 2
1766319049 226153980
```

Added by: Mauro Persano
Date: 2007-08-18
Time limit: 5s
Source limit: 50000B
Languages: All
Resource: Brahmagupta, circa 628 AD

SPOJ Problem Set (classical)

1740. Gerrymandering

Problem code: GERRYMDR

Politicians like to get elected. They will do just about anything to get elected. Including changing the rules of the voting: who can vote, where you can vote, when you can vote, etc. One very common practice is called *gerrymandering*, where the boundaries of "ridings" are redrawn to favour a particular candidate (the one doing the redrawing, of course).

Your task is to help determine how to do some simple, linear gerrymandering.

You will be given the information about N ridings ($2 <= N <= 10000$) where there are candidates from p ($2 <= p <= 10$) different parties. These N ridings are linear, in the sense that they are side-by-side; there are two ridings (on the ends) that have only one adjacent riding, with the rest of the ridings having two adjacent ridings. A picture is shown below for $N = 4$ and $p = 2$ (which is also the sample data):

	Riding 1	Riding 2	Riding 3	Riding 4
Votes for Party 1	1	4	1	6
Votes for Party 2	5	3	2	1

Note that Riding 1 and Riding 2 are adjacent, Riding 2 and 3 are adjacent, Riding 3 and 4 are adjacent. No other ridings are adjacent.

You have some financial backing that will let you bribe the people in charge of setting the boundaries of ridings: in particular, there is a fixed rate to merge two adjacent boundaries. When you merge two ridings, the votes of the ridings merge together, in the sense that the number of votes of party 1 is the sum of the votes of party 1 in each riding, and likewise for all other parties.

Your task is to merge the minimum number of regions such that the first party (your party!) has a majority of the ridings. Note that to win a riding, the party must have more votes than any other party in that riding. Also note that to have a majority of ridings, if there are Q ridings (where $Q <= N$), then your party has won at least floor(Q/2) + 1 of the ridings.

Input

The input begins with the integer t, the number of testcases. Then t testcases follow.
The first line of each testcase will consist of the integer N. The second line will consist of the integer p. The next N lines will each contain p non-negative integers (where each integer is at most 10000), separated by one space character. Specifically, the p integers on each line are v_1 v_2 ... v_p where v_1 is the number of votes that party 1 will receive in this riding, v_2 is the number of votes that party 2 will receive in this riding, etc. You may also assume that the total number of voters is less than 2 billion.

Output

For each testcase, print one line, consisting of an integer, which gives the minimum number of ridings that need to be merged in order for the first party to win a majority of ridings. If the first party cannot win, even with any number of mergers, output -1.

Example

Input:

```
2
4
2
1 5
4 3
1 2
6 1
3
3
2 0 1
1 3 0
0 0 1
```

Output:

```
1
-1
```

Added by: Jin Bin
Date: 2007-08-19
Time limit: 20s
Source limit: 50000B
Languages: All except: C99 strict
Resource: Canadian Computing Competition 2007, day 2

SPOJ Problem Set (classical)

1741. Tetris 3D

Problem code: TETRIS3D

The authors of the game "Tetris" have decided to make a new, three-dimensional version, in which cuboids would fall down on a rectangular platform. The blocks fall down separately in a certain order, just like in the two-dimensional game. A block falls down until it reaches an obstacle: the platform or another block, that has already stopped - then it stops and remains in this exact position till the game is over.
However, the authors wanted to change the spirit of the game, turning it from a simple arcade-game into a play far more puzzling. Knowing the order of the falling blocks and their flight path the player's task is to tell the height of the highest point of the arrangement after all blocks have fallen down (and stopped). All the blocks are falling down vertically and do not rotate while falling. For convenience we'll introduce a cartesian coordinate system on the platform, with the center in one of the platform's corners and the axes parallel to the platform's edges.
Write a program that:
* reads the descriptions of subsequent falling blocks from the standard input,
* determines the height of the highest point of the arrangement of blocks after all have fallen down and stopped,
* writes the result to the standard output.

Input

In the first line of the input there are three integers D, S and N (1<=N<=20 000, 1<=D, S<=1 000), separated by single spaces and denoting respectively: the length and the depth of the platform and the number of blocks that are going to fall down on it. In the following N lines the descriptions of subsequent blocks are given, one in each line.
Each description of a block consists of five integers: d, s, w, x and y (1<=d, 0<=x,d+x<=D, 1<=s,0<=y,s+y<=S, 1<=w<=100 000), representing a block of length d depth s and height w This very block will be falling down on the platform with its d×s face as the bottom, where the length and depth of the block are parallel to those of the platform. The coordinates of the vertices of the projection of the block on the platform are: (x, y), (x + d, y), (x, y + s) and (x + d, y + s).

Output

The first and only line of the standard output should contain exactly one integer, the height of the highest point of the arrangement of blocks after all have fallen down ad stopped.

Example

Input:

```
7 5 4
4 3 2 0 0
3 3 1 3 0
7 1 2 0 3
```

```
2 3 3 2 2
```

Output:

```
6
```

Added by:	Jin Bin
Date:	2007-08-19
Time limit:	2s-4s
Source limit:	50000B
Languages:	All except: C99 strict
Resource:	POI XIII - Stage I

SPOJ Problem Set (classical)

1744. Evaluate the polynomial

Problem code: POLEVAL

Your task consists of evaluate a polynomial of degree **n** (**0 <= n <= 999**) represented by its **n+1** coefficients of the form:

$$p_n(x) = c_n x^n + c_{n-1} x^{n-1} + ... + c_2 x^2 + c_1 x + c_0$$

in each one of the **k** (**1 <= k <= 100**) points x_1, x_2, ..., x_k. The coefficients of the polynomial and the values where they will be evaluated are integers in the interval **[-100, 100]** that guarantees that the polynomial's evaluation is at the most $2^{63} - 1$.

Input

There will be multiple test cases, each one with **4** lines that are described below
n: degree of polynomial.
$c_n\ c_{n-1}\ ...\ c_2\ c_1\ c_0$: coefficients of the polynomial separated by a single space.
k: number of points to evaluate the polynomial.
$x_1\ x_2\ ...\ x_{k-1}\ x_k$: points to evaluate the polynomial separated by a single space.
The final test case is a single line where **n = -1** and this case should not be processed.

Output

For each test case you should print **k + 1** lines of output, the very first line containing the case number and the following **k** lines with the result of the polynomial's evaluation in each one of the **k** given points. See the sample.

Example

Input:
```
21 -2 -150 1 -1 2 -232 1 -2 -140 -1 2 -2-1
```
Output:
```
Case 1:-1-22-17Case 2:-1015-9
```

Added by: Ivan Alfonso Olamendy
Date: 2007-08-25
Time limit: 2.5s
Source limit:50000B
Languages: All except: C99 strict
Resource: My own resource

SPOJ Problem Set (classical)

1748. Sequence Partitioning II

Problem code: SEQPAR2

Given a sequence of N ordered pairs of positive integers (A_i, B_i), you have to partition it into several contiguous parts. Let p be the number of these parts, whose boundaries are (l_1, r_1), (l_2, r_2), ... ,(l_p, r_p), which satisfy $l_i = r_{i-1} + 1$, $l_i <= r_i$, $l_1 = 1$, $r_p = n$. The parts themselves also satisfy the following restrictions:

1. For any two pairs (A_p, B_p), (A_q, B_q), where (A_p, B_p) is belongs to the T_pth part and (A_q, B_q) the T_qth part. If $T_p < T_q$, then $B_p > A_q$.

2. Let M_i be the maximum A-component of elements in the ith part, say

   $M_i = \max \{A_{l_i}, A_{l_{i+1}}, ..., A_{r_i}\}$, $1 <= i <= p$

   it is provided that

   [IMAGE]
   where Limit is a given integer.

Let S_i be the sum of B-components of elements in the ith part.

Now I want to minimize the value

$\max\{S_i : 1 <= i <= p\}$

Could you tell me the minimum?

Input

The input contains exactly one test case. The first line of input contains two positive integers N (N <= 50000), Limit (Limit <= 2^{31}-1). Then follow N lines each contains a positive integers pair (A, B). It's always guaranteed that

$\max\{A_1, A_2, ..., A_n\} <= \text{Limit}$
[IMAGE]

Output

Output the minimum target value.

Example

Input:

```
4 6
4 3
3 5
2 5
2 4
```

Output:

```
9
```

Explanation

An available assignment is the first two pairs are assigned into the first part and the last two pairs are assigned into the second part. Then $B_1 > A_3$, $B_1 > A_4$, $B_2 > A_3$, $B_2 > A_4$, max$\{A_1, A_2\}$+max$\{A_3, A_4\}$ <= 6, and minimum max $\{B_1+B_2, B_3+B_4\}$=9.

Added by: Jin Bin
Date: 2007-08-28
Time limit: 1s-15s
Source limit: 50000B
Languages: All except: C99 strict
Resource: POJ Monthly--2007.07.08

SPOJ Problem Set (classical)

1754. Divisor Summation (Hard)

Problem code: DIVSUM2

Given a natural number n (1 <= n <= 1e16), please output the summation of all its proper divisors.

Definition: A proper divisor of a natural number is the divisor that is strictly less than the number.

e.g. number 20 has 5 proper divisors: 1, 2, 4, 5, 10, and the divisor summation is: 1 + 2 + 4 + 5 + 10 = 22.

Input

An integer stating the number of test cases (equal to 500), and that many lines follow, each containing one integer between 1 and 1e16 inclusive.

Output

One integer each line: the divisor summation of the integer given respectively.

Example

Input:
```
3
2
10
20
```

Output:
```
1
8
22
```

warning: a naive algorithm may not run in time.

Added by: Jin Bin
Date: 2007-08-29
Time limit: 60s
Source limit:50000B
Languages: All except: C99 strict
Resource: own problem

SPOJ Problem Set (classical)

1771. Yet Another N-Queen Problem

Problem code: NQUEEN

After solving Solution to the *n* Queens Puzzle by constructing, LoadingTime wants to solve a harder version of the N-Queen Problem. Some queens have been set on particular locations on the board in this problem. Can you help him??

Input

The input contains multiple test cases. Every line begins with an integer N (N<=50), then N integers followed, representing the column number of the queen in each rows. If the number is 0, it means no queen has been set on this row. You can assume there is at least one solution.

Output

For each test case, print a line consists of N numbers separated by spaces, representing the column number of the queen in each row. If there are more than one answer, print any one of them.

Example

Input:
```
4 0 0 0 0
8 2 0 0 0 4 0 0 0
```

Output:
```
2 4 1 3
2 6 1 7 4 8 3 5
```

Added by: Jin Bin
Date: 2007-09-06
Time limit: 5s
Source limit: 10000B
Languages: All except: C99 strict
Resource: own problem

SPOJ Problem Set (classical)

1772. Find The Determinant II

Problem code: DETER2

In this problem you have to calculate the determinant of an N x N matrix whose entries are given by **m[i][j] = gcd(i,j)^{k}** , 1 <= i,j <= N.
Here gcd(i,j) denotes the greatest common divisor of i and j.
As the determinant D can grow very large, you have to print D%1000003.

Input

First line of input consists of a single integer containing the number of test cases T (equal to around 20), each of the following T lines contain two integers N and k where N is the size of the matrix and k is the exponent.
1 <= N <= 1000000
1 <= k <= 10^{9}

Output

One line corresponding to each test case containing the determinant modulo 1000003 for the corresponding test case.

Example

Input:
```
3
4 2
2 4
4 3
```

Output:
```
288
15
10192
```

Note: You may want to solve DETER first, in case you havent already solved it.

Added by: Ajay Somani
Date: 2007-09-07
Time limit: 1s
Source limit: 2048B
Languages: All
Resource: "The Art of Computer Programming"

SPOJ Problem Set (classical)

1774. All Discs Considered

Problem code: ALL

Operating systems are large software artefacts composed of many packages, usually distributed on several media, e.g., discs. You probably remember the time when your favourite operating system was delivered on 21 floppy discs, or, a few years later, on 6 CDs. Nowadays, it will be shipped on several DVDs, each containing tens of thousands of packages.

The installation of certain packages may require that other packages have been installed previously. Therefore, if the packages are distributed on the media in an unsuitable way, the installation of the complete system requires you to perform many media changes, provided that there is only one reading device available, e.g., one DVD-ROM drive. Since you have to start the installation somehow, there will of course be one or more packages that can be installed independently of all other packages.

Given a distribution of packages on media and a list of dependences between packages, you have to calculate the minimal number of media changes required to install all packages. For your convenience, you may assume that the operating system comes on exactly 2 DVDs.

Input

The input contains several test cases. Every test case starts with three integers N_1, N_2, D. You may assume that $1 \le N_1, N_2 \le 50000$ and $0 \le D \le 100000$. The first DVD contains N_1 packages, identified by the numbers $1, 2, \ldots, N_1$. The second DVD contains N_2 packages, identified by the numbers $N_1+1, N_1+2, \ldots, N_1+N_2$. Then follow D dependence specifications, each consisting of two integers x_i, y_i. You may assume that $1 \le x_i, y_i \le N_1+N_2$ for $1 \le i \le D$. The dependence specification means that the installation of package x_i requires the previous installation of package y_i. You may assume that there are no circular dependences. The last test case is followed by three zeros.

Output

For each test case output on a line the minimal number of DVD changes required to install all packages. By convention, the DVD drive is empty before the installation and the initial insertion of a disc counts as `one` change. Likewise, the final removal of a disc counts as `one` change, leaving the DVD drive empty after the installation.

Example

Input:

```
3 2 1
1 2
2 2 2
1 3
4 2
2 1 1
1 3
```

```
0 0 0
```

Output:

```
3
4
3
```

Added by: Wanderley Guimaraes
Date: 2007-09-14
Time limit: 5s
Source limit: 50000B
Languages: All
Resource: University of Ulm Local Contest 2004

SPOJ Problem Set (classical)

1775. Boolean Logic

Problem code: BOOLE

Propositions are logical formulas consisting of proposition symbols and connecting operators. They are recursively defined by the following rules:

1. All proposition symbols (in this problem, lower-case alphabetic characters, e.g., `a` and `z`) are propositions.
2. If `P` is a proposition, `(!P)` is a proposition, and `P` is a direct subformula of it.
3. If `P` and `Q` are propositions, `(P&Q)`, `(P|Q)`, `(P-->Q)`, and `(P<->Q)` are propositions, and `P` and `Q` are direct subformulas of them.
4. Nothing else is a proposition.

The operations `!`, `&`, `|`, `-->`, and `<->` denote logical negation, conjunction, disjunction, implication, and equivalence, respectively. A proposition `P` is a subformula of a proposition `R` if `P=R` or `P` is a direct subformula of a proposition `Q` and `Q` is a subformula of `R`.

Let `P` be a proposition and assign boolean values (i.e., `0` or `1`) to all proposition symbols that occur in `P`. This induces a boolean value to all subformulas of `P` according to the standard semantics of the logical operators:

negation	conjunction	disjunction	implication	equivalence
`!0=1`	`0&0=0`	`0|0=0`	`0-->0=1`	`0<->0=1`
`!1=0`	`0&1=0`	`0|1=1`	`0-->1=1`	`0<->1=0`
	`1&0=0`	`1|0=1`	`1-->0=0`	`1<->0=0`
	`1&1=1`	`1|1=1`	`1-->1=1`	`1<->1=1`

This way, a value for `P` can be calculated. This value depends on the choice of the assignment of boolean values to the proposition symbols. If `P` contains `n` different proposition symbols, there are 2^n different assignments. To evaluate all possible assignments we may use truth tables.

A truth table contains one line per assignment (i.e., 2^n lines in total). Every line contains the values of all subformulas under the chosen assignment. The value of a subformula is aligned with the proposition symbol, if the subformula is a proposition symbol, and with the center of the operator otherwise.

Input Specification

The input contains several test cases, each on a separate line. Every test case denotes a proposition and may contain arbitrary amounts of spaces in between. The input file terminates immediately after the newline symbol following the last test case.

Output Specification

For each test case generate a truth table for the denoted proposition. Start the truth table by repeating the input line. Evaluate the proposition (and its subformulas) for all assignments to its variables, and output one line for each assignment. The line must have the same length as the corresponding input line and must consist only of spaces and the characters `0` and `1`. Output an empty line after each test case.

Let $s_1, \ldots, s_n$ be the proposition symbols in the denoted proposition sorted in alphabetic order. Then, all assignments of `0` to s_1 must precede the assignments of `1` to s_1. Within each of these blocks of assignments, all assignments of `0` to s_2 must precede the assignments of `1` to s_2, and so on.

Sample Input

```
((b --> a) <-> ((! a) --> (! b)))
  ((y &  a)   -  ->(c |c))
```

Sample Output

```
((b --> a) <-> ((! a) --> (! b)))
  0  1  0   1    1 0   1   1 0
  1  0  0   1    1 0   0   0 1
  0  1  1   1    0 1   1   1 0
  1  1  1   1    0 1   1   0 1

  ((y &  a)   -  ->(c |c))
    0 0  0       1  0 00
    1 0  0       1  0 00
    0 0  0       1  1 11
    1 0  0       1  1 11
    0 0  1       1  0 00
    1 1  1       0  0 00
    0 0  1       1  1 11
    1 1  1       1  1 11
```

Added by: Wanderley Guimaraes
Date: 2007-09-14
Time limit: 1s
Source limit: 50000B
Languages: All
Resource: University of Ulm Local Contest 2004

SPOJ Problem Set (classical)

1776. DNA Laboratory

Problem code: DNALAB

Having started to build his own DNA lab just recently, the evil doctor Frankenstein is not quite up to date yet. He wants to extract his DNA, enhance it somewhat and clone himself. He has already figured out how to extract DNA from some of his blood cells, but unfortunately reading off the DNA sequence means breaking the DNA into a number of short pieces and analyzing those first. Frankenstein has not quite understood how to put the pieces together to recover the original
sequence.

His pragmatic approach to the problem is to sneak into university and to kidnap a number of smart looking students. Not surprisingly, you are one of them, so you would better come up with a solution pretty fast.

You are given a list of strings over the alphabet A (for adenine), C (cytosine), G (guanine), and T (thymine), and your task is to find the shortest string (which is typically not listed) that contains all given strings as substrings. If there are several such strings of shortest length, find the smallest in alphabetical/lexicographical order.

Input
The fist line contains the number of scenarios. For each scenario, the first line contains the number n of strings with 1<=n<=15. Then these strings with 1 <= length <= 100 follow, one on each line, and they consist of the letters ‘A’, ‘C’, ‘G’, and ‘T’ only.

Output
The output for every scenario begins with a line containing "Scenario #i:", where i is the number of the scenario starting at 1. Then print a single line containing the shortest (and smallest) string as described above. Terminate the output for the scenario with a blank line.

Sample Input
1
2
TGCACA
CAT

Sample Output

Scenario #1:
TGCACAT

Added by:	Andrés Leonardo Rojas Duarte
Date:	2007-09-15
Time limit:	12s
Source limit:	50000B
Languages:	All
Resource:	TUD Programming Contest 2004

SPOJ Problem Set (classical)

1784. IOICamp Sequence

Problem code: ICAMPSEQ

Let's say we have 4 N-elements sequences of real numbers: A[], B[], C[], D[] .
Funtion F(i, j) is defined: F(i, j) = $|A_i - A_j| + |B_i - B_j| + |C_i - C_j| + |D_i - D_j|$ (1 <= i, j <= N).
Your task is very easy: you have to find the maximum of F(i, j).

Input

The first line: N (N <= 100000).
Following are N lines: the i-th line contains four real numbers A_i, B_i, C_i, D_i. (-10^9 <= A_i, B_i, C_i, D_i <= 10^9)

Output

Only one line is the maximum of F(i, j).
(The result takes exactly 3 decimal places)

Example

Input:
```
2
1.0 1.0 2.0 0.5
1.0 1.0 0.5 2.0
```

Output:
```
3.000
```

Added by: Nghia Nguyen Hoang
Date: 2007-09-18
Time limit: 1s
Source limit: 50000B
Languages: All
Resource: IOICamp and Ms. Thanh Vy Hua Le

SPOJ Problem Set (classical)

1785. Code

Problem code: CODE

KEY Inc., the leading company in security hardware, has developed a new kind of safe. To unlock it, you don't need a key but you are required to enter the correct n-digit code on a keypad (as if this were something new!). There are several models available, from toy safes for children (with a 2-digit code) to the military version (with a 6-digit code).

The safe will open as soon as the last digit of the correct code is entered. There is no "enter" key. When you enter more than n digits, only the last n digits are significant. For example (in the 4-digit version), if the correct code is 4567, and you plan to enter the digit sequence 1234567890, the door will open as soon as you press the 7 key.

The software to create this effect is rather simple. In the n-digit version the safe is always in one of 10^{n-1} internal states. The current state of the safe simply represents the last n−1 digits that have been entered. One of these states (in the example above, state 456) is marked as the unlocked state. If the safe is in the unlocked state and then the right key (in the example above, 7) is pressed, the door opens. Otherwise the safe shifts to the corresponding new state. For example, if the safe is in state 456 and then you press 8, the safe goes into state 568.

A trivial strategy to open the safe is to enter all possible codes one after the other. In the worst case, however, this will require $n * 10^n$ keystrokes. By choosing a good digit sequence it is possible to open the safe in at most $10^n + n - 1$ keystrokes. All you have to do is to find a digit sequence that contains all n-digit sequences exactly once. KEY Inc. claims that for the military version (n=6) the fastest computers available today would need billions of years to find such a sequence - but apparently they don't know what some programmers are capable of...

Input Specification

The input contains several test cases. Every test case is specified by an integer n. You may assume that $1<=n<=6$. The last test case is followed by a zero.

Output Specification

For each test case specified by n output a line containing a sequence of $10^n + n - 1$ digits that contains each n-digit sequence exactly once.

Sample Input

```
1
2
0
```

Sample Output

```
0123456789
00102030405060708091121314151617181922324252627282933435363738394454647484955657585966768697787988990
```

Added by: Wanderley Guimaraes
Date: 2007-09-19
Time limit: 2s
Source limit:50000B
Languages: All
Resource: University of Ulm Local Contest 2004

SPOJ Problem Set (classical)

1786. In Danger

Problem code: DANGER

Flavius Josephus and 40 fellow rebels were trapped by the Romans. His companions preferred suicide to surrender, so they decided to form a circle and to kill every third person and to proceed around the circle until no one was left. Josephus was not excited by the idea of killing himself, so he calculated the position to be the last man standing (and then he did not commit suicide since nobody could watch).

We will consider a variant of this "game" where every second person leaves. And of course there will be more than 41 persons, for we now have computers. You have to calculate the safe position. Be careful because we might apply your program to calculate the winner of this contest!

Input Specification

The input contains several test cases. Each specifies a number `n`, denoting the number of persons participating in the game. To make things more difficult, it always has the format `"xyez"` with the following semantics: when `n` is written down in decimal notation, its first digit is `x`, its second digit is `y`, and then follow `z` zeros. Whereas `0<=x,y<=9`, the number of zeros is `0<=z<=6`. You may assume that `n>0`. The last test case is followed by the string `00e0`.

Output Specification

For each test case generate a line containing the position of the person who survives. Assume that the participants have serial numbers from `1` to `n` and that the counting starts with person 1, i.e., the first person leaving is the one with number 2. For example, if there are 5 persons in the circle, counting proceeds as `2, 4, 1, 5` and person `3` is staying alive.

Sample Input

```
05e0
01e1
42e0
66e6
00e0
```

Sample Output

```
3
5
21
64891137
```

Added by: Wanderley Guimaraes
Date: 2007-09-19
Time limit: 1s
Source limit: 50000B
Languages: All
Resource: University of Ulm Local Contest 2004

SPOJ Problem Set (classical)

1787. Run Length Encoding

Problem code: ENCONDIN

Your task is to write a program that performs a simple form of run-length encoding, as described by the rules below.

Any sequence of between 2 to 9 identical characters is encoded by two characters. The first character is the length of the sequence, represented by one of the characters `2` through `9`. The second character is the value of the repeated character. A sequence of more than 9 identical characters is dealt with by first encoding 9 characters, then the remaining ones.

Any sequence of characters that does not contain consecutive repetitions of any characters is represented by a `1` character followed by the sequence of characters, terminated with another `1`. If a `1` appears as part of the sequence, it is escaped with a `1`, thus two `1` characters are output.

Input Specification

The input consists of letters (both upper- and lower-case), digits, spaces, and punctuation. Every line is terminated with a newline character and no other characters appear in the input.

Output Specification

Each line in the input is encoded separately as described above. The newline at the end of each line is not encoded, but is passed directly to the output.

Sample Input

```
AAAAAABCCCC
12344
```

Sample Output

```
6A1B14C
11123124
```

Added by: Wanderley Guimaraes
Date: 2007-09-19
Time limit: 1s
Source limit:50000B
Languages: All
Resource: University of Ulm Local Contest 2004

SPOJ Problem Set (classical)

1788. Fractan

Problem code: FRACTAN

To play the "fraction game" corresponding to a given list $f_1, f_2, \ldots, f_k$ of fractions and starting integer N, you repeatedly multiply the integer you have at any stage (initially N) by the earliest f_i in the list for which the answer is integral. Whenever there is no such f_i, the game stops.

Formally, we define a sequence by $S_0 = N$, and $S_{j+1} = f_i S_j$, if for $1 <= i <= k$, the number $f_i S_j$ is an integer but the numbers $f_1 S_j, \ldots, f_{i-1} S_j$ are not.

For example, if we have the list of eight fractions $f_1 = 170/39$, $f_2 = 19/13$, $f_3 = 13/17$, $f_4 = 69/95$, $f_5 = 19/23$, $f_6 = 1/19$, $f_7 = 13/7$, $f_8 = 1/3$, and start with $N = 21$, we produce the (finite) sequence $(21,39,170,130,190,138,114,6,2)$. In general, the sequence may be infinite.

Given a fraction list and a starting integer calculate a part of the defined sequence. Actually, we are interested only in the powers of 2 that appear in the sequence.

Input Specification

The input contains several test cases. Every test case starts with three integers m, N, k. You may assume that $1 <= m <= 40$, $1 <= N <= 1000$, and $1 <= k <= 100$. Then follow k fractions $f_1, \ldots, f_k$. For each fraction, first its numerator is given, followed by its denominator. You may assume that both are positive integers less than 1000 and their greatest common divisor is 1. The last test case is followed by a zero.

Output Specification

For each test case output on a line m numbers $e_1, \ldots, e_m$, separated by one space character, such that $2^{e_1}, \ldots, 2^{e_k}$ are the first m numbers in the defined sequence that are powers of 2. You may assume that there are at least m powers of 2 among the first 7654321 elements of the sequence.

Sample Input

```
1 21 8 170 39 19 13 13 17 69 95 19 23 1 19 13 7 1 3
20 2 14 17 91 78 85 19 51 23 38 29 33 77 29 95 23 77 19 1 17 11 13 13 11 15 2 1 7 55 1
0
```

Sample Output

```
1
1 2 3 5 7 11 13 17 19 23 29 31 37 41 43 47 53 59 61 67
```

Added by: Wanderley Guimaraes
Date: 2007-09-19
Time limit: 2s
Source limit: 50000B
Languages: All
Resource: University of Ulm Local Contest 2004

SPOJ Problem Set (classical)

1789. Huffman´s Greed

Problem code: GREEDULM

In the following we define the basic terminology of trees. A **tree** is defined inductively: It has a **root** which is either an **external node** (a leaf), or an **internal node** having a sequence of trees as its children. An internal node is also called the **parent** of the roots of its child trees. The **level** of a node in a tree is defined inductively: The root has level 0, and the level of a node is 1 more than the level of its parent node.

Every internal node of a **binary tree** has precisely two children, its left sub-tree and its right sub-tree. Every internal node of a **labelled binary tree** is additionally marked with a string, its label. A **binary search tree** is a labelled binary tree where every internal node t satisfies the following condition: All labels of nodes in the left sub-tree of t are less than the label of t which is, in turn, less than all labels of nodes in the right sub-tree of t. For this condition, we assume lexicographic, i.e., alphabetic order on the strings.

An **inorder traversal** of a tree is defined recursively: A leaf is just visited, and for an internal node first its left sub-tree is traversed inorder, then the node itself is visited, finally its right sub-tree is traversed inorder. It follows that an inorder traversal of a binary search tree yields the labels in lexicographic order. Note that binary search trees whose shapes differ may nevertheless yield the same sequence of strings while being traversed inorder.

When a given string s is looked for in a binary search tree, we compare s to the label l of the root. We are done if s=l, otherwise if s<l we continue to search in the left sub-tree, and if s>l in the right sub-tree. If a leaf is reached, we know that s is not in the tree.

The number of comparisons performed in such a search procedure depends on s and the actual shape of the search tree. Therefore, there is an interest in constructing binary search trees that store a given sequence of strings but provide as efficient access as possible. Of course, we don't know in advance which strings will be looked up in the tree, so we need to make some assumptions.

Let n be the number of strings that are to be stored in the binary search tree. Let $K_1, \ldots, K_n$ be these strings in lexicographic order. Let $p_1, \ldots, p_n$ and $q_0, \ldots, q_n$ be 2n+1 non-negative real numbers such that $\sum_{i=1..n} p_i + \sum_{i=0..n} q_i = 1$. The interpretation of these numbers is:

- p_i = probability that the search argument s is K_i.
- q_i = probability that s lies (lexicographically) strictly between K_i and K_{i+1}.

By convention, q_0 is the probability that s is less than K_1, and q_n is the probability that s is greater than K_n. We want to find a binary search tree containing nodes with labels $K_1, \ldots, K_n$ that minimises the expected number of comparisons in the search, namely

$$\text{cost} = \sum_{i=1..n} p_i * (1 + \text{level of internal node } K_i) + \sum_{i=0..n} q_i * (\text{level of leaf between } K_i \text{ and } K_{i+1}).$$

The leaf between K_i and K_{i+1} is that leaf reached in the search for a string s that lies (lexicographically) strictly between K_i and K_{i+1}. Adhere to the convention stated above for the border cases.

The following figure illustrates the first test case of the sample input. It shows the two possible binary search trees, the probabilities and the associated costs.

[IMAGE]

Input Specification

The input contains several test cases. Every test case starts with an integer n. You may assume that $1<=n<=200$. Then follow $2n+1$ non-negative integers denoting frequencies. Let s be the sum of all frequencies. You may assume that $1<=s<=1000000$. The probabilities $p_1,...,p_n$ and $q_0,...,q_n$ are calculated in this order by dividing the frequencies by s. The last test case is followed by a zero.

Output Specification

For each test case devise a binary search tree whose cost is minimal for the specified probabilities. Output the integer $cost*s$ for such a tree.

Sample Input

```
2
20 15 15 25 25
35
142 35 58 5 20 5 10 9 15 23 129 4 52 5 38 18 9 7 2 4 266 93 5 18 18 27 5 10 11 180 4 32 21 3 21
0 55 27 36 85 31 58 3 334 0 98 27 113 89 180 0 62 12 0 37 0 3 64 70 0 277 0 0 0 170 0 18 76 27 3 29
0
```

Sample Output

```
160
13637
```

Added by: Wanderley Guimaraes
Date: 2007-09-19
Time limit: 2s
Source limit: 50000B
Languages: All
Resource: University of Ulm Local Contest 2004

SPOJ Problem Set (classical)

1790. Binary Search Heap Construction

Problem code: HEAPULM

Read the statement of problem G for the definitions concerning trees. In the following we define the basic terminology of heaps. A **heap** is a tree whose internal nodes have each assigned a **priority** (a number) such that the priority of each internal node is less than the priority of its parent. As a consequence, the root has the greatest priority in the tree, which is one of the reasons why heaps can be used for the implementation of priority queues and for sorting.

A binary tree in which each internal node has both a label and a priority, and which is both a binary search tree with respect to the labels and a heap with respect to the priorities, is called a **treap**. Your task is, given a set of label-priority-pairs, with unique labels and unique priorities, to construct a treap containing this data.

Input Specification

The input contains several test cases. Every test case starts with an integer `n`. You may assume that `1<=n<=50000`. Then follow `n` pairs of strings and numbers $l_1/p_1,...,l_n/p_n$ denoting the label and priority of each node. The strings are non-empty and composed of lower-case letters, and the numbers are non-negative integers. The last test case is followed by a zero.

Output Specification

For each test case output on a single line a treap that contains the specified nodes. A treap is printed as `(<left sub-treap><label>/<priority><right sub-treap>)`. The sub-treaps are printed recursively, and omitted if leafs.

Sample Input

```
7 a/7 b/6 c/5 d/4 e/3 f/2 g/1
7 a/1 b/2 c/3 d/4 e/5 f/6 g/7
7 a/3 b/6 c/4 d/7 e/2 f/5 g/1
0
```

Sample Output

```
(a/7(b/6(c/5(d/4(e/3(f/2(g/1)))))))
(((((((a/1)b/2)c/3)d/4)e/5)f/6)g/7)
(((a/3)b/6(c/4))d/7((e/2)f/5(g/1)))
```

Added by: Wanderley Guimaraes
Date: 2007-09-19
Time limit: 4s
Source limit: 50000B
Languages: All
Resource: University of Ulm Local Contest 2004

SPOJ Problem Set (classical)

1793. Text Generater II

Problem code: GEN2

HL, HJ and FGD set problem for ZMY everyday. The title of each problem bores them very much. The title is a string which consists only lower case letters. It's length is *L*, and, of course, the title must contain their names(hl, hj, fgd) as consecutive substrings.More apparently, *N* evil consecutive substrings should not appear in the title. Any string satisfying the two conditions above is OK.

The task ZMY is to do today is: if they give ZMY 8 problems every week, and the title of each problem should not be repeated, how many (complete) weeks can they set problems?

Input

Multiple test cases.Input terminates by EOF.

For each test case:

The first line contains two space-seperated integers L(0<=L<=10^9) and N(0<=N<=20). N lines follow, each contains a string (contains only lowercase letters), which is evil.You can assume the total length of the N evil strings is no more than 20.

Output

For each test case output one line contains the answer modudo 1000.

Example

Input:
```
10 1
zmy
```

Output:
```
245
```

Added by: Blue Mary
Date: 2007-09-20
Time limit: 7s
Source limit:50000B
Languages: All except: C99 strict
Resource: description by Blue Mary; standard program and test data by g201513

SPOJ Problem Set (classical)

1794. Greedy Hydra II

Problem code: DRAGON2

The problem description is the same as the problem DRAGON.

Input

The first line contains 3 integers N(1<=N<=3000),M(2<=M<=N),K(1<=K<=N), separated by single spaces. The N fruits are numbered 1..N, and the biggest fruit is always numbered 1. N-1 lines follow, each contains 3 integers i,j,k separated by spaces denoted that there is a branch between fruit i(1<=i<=N) and fruit j(1<=j<=N) and the weight of illness of this branch is k(0<=k<=100000).

Output

Output one line contains a single integer denoted the minimum weight of illness of the hydra. If we can't divide the fruit into M groups, output "-1"(without quotes).

Example

Input:

```
8 2 4
1 2 20
1 3 4
1 4 13
2 5 10
2 6 12
3 7 15
3 8 5
```

Output:

```
4
```

Blue Mary's Note: some new test cases were added on Dec.7, 2007.

Added by: Blue Mary
Date: 2007-09-20
Time limit: 0.5s-1s
Source limit:50000B
Languages: All except: C99 strict
Resource: description by Blue Mary; standard program and test data by lcosvse

SPOJ Problem Set (classical)

1797. Cardsharper

Problem code: CARD

Zenek is a well known (at least in Byteotia) card-sharper. He spent most of his best years practicing one card shuffle with his deck of n cards, which for simplicity we will call 1,2,...,n. Unfortunately, it turns out that knowing this one card shuffle a is not enough to earn a good living. To become rich and famous Zenek needs to know k shuffles $c_1,...,c_k$. As he doesn't have enough time to learn all of them, he decided to learn only one shuffle b so that using both a and b he will be able to perform as many of c_i as it is possible.

Each shuffle is described by n numbers $t_1,t_2,...,t_n$. Such description means that after performing shuffle, card that was originally at position i will be at position t_i.

Task

Find shuffle b maximizing number of shuffles that can be performed.

Input

First line contains n (2<=n<=52). Second line contains n numbers $a_1,a_2,...,a_n$ describing shuffle that Zenek already knows.Third line contains k (2<=k<=6). i-th of the next k lines contains description of c_i.

Output

First line contains description of the shuffle b that Zenek shoud learn. i-th of the next k lines contains:

- -1 when it is not possible to perform c_i using only a and b
- $m,r_1,r_2,...,r_m$ (0<=m<=500000, $0<=r_i<=10^6$) meaning that applying a r_1 times, then b r_2 times, then a r_3 times and so on is the same as applying shuffle c_i once.

Examples

Input

```
5
2 3 4 5 1
3
1 3 2 4 5
1 2 3 4 5
5 4 3 2 1
```

Output

```
2 1 3 4 5
3 4 1 1
0
9 1 1 3 1 4 1 1 1 1
```

Input

```
5
1 2 3 4 5
3
1 3 2 4 5
5 4 3 2 1
1 2 5 4 3
```

Output

```
1 3 2 4 5
2 0 1
-1
-1
```

Added by: Pawel Gawrychowski
Date: 2007-09-20
Time limit: 1s
Source limit: 50000B
Languages: All
Resource: Fajne Zawody Informatyczne

SPOJ Problem Set (classical)

1798. Assistance Required

Problem code: ASSIST

After the 1997/1998 Southwestern European Regional Contest (which was held in Ulm) a large contest party took place. The organization team invented a special mode of choosing those participants that were to assist with washing the dirty dishes. The contestants would line up in a queue, one behind the other. Each contestant got a number starting with 2 for the first one, 3 for the second one, 4 for the third one, and so on, consecutively.

The first contestant in the queue was asked for his number (which was 2). He was freed from the washing up and could party on, but every second contestant behind him had to go to the kitchen (those with numbers 4, 6, 8, etc). Then the next contestant in the remaining queue had to tell his number. He answered 3 and was freed from assisting, but every third contestant behind him was to help (those with numbers 9, 15, 21, etc). The next in the remaining queue had number 5 and was free, but every fifth contestant behind him was selected (those with numbers 19, 35, 49, etc). The next had number 7 and was free, but every seventh behind him had to assist, and so on.

Let us call the number of a contestant who does not need to assist with washing up a lucky number. Continuing the selection scheme, the lucky numbers are the ordered sequence 2, 3, 5, 7, 11, 13, 17, etc. Find out the lucky numbers to be prepared for the next contest party.

Input Specification

The input contains several test cases. Each test case consists of an integer `n`. You may assume that `1<=n<=3000`. A zero follows the input for the last test case.

Output Specification

For each test case specified by `n` output on a single line the `n`-th lucky number.

Sample Input

```
1
2
10
20
0
```

Sample Output

```
2
3
29
83
```

Added by: Wanderley Guimaraes
Date: 2007-09-21
Time limit: 1s
Source limit: 50000B
Languages: All
Resource: University of Ulm Local Contest 2003

SPOJ Problem Set (classical)

1799. The Bottom of a Graph

Problem code: BOTTOM

We will use the following (standard) definitions from graph theory. Let V be a nonempty and finite set, its elements being called vertices (or nodes). Let E be a subset of the Cartesian product $V \times V$, its elements being called edges. Then $G=(V,E)$ is called a directed graph.

Let n be a positive integer, and let $p=(e_1,\ldots,e_n)$ be a sequence of length n of edges $e_i \in E$ such that $e_i=(v_i,v_{i+1})$ for a sequence of vertices $(v_1,\ldots,v_{n+1})$. Then p is called a path from vertex v_1 to vertex v_{n+1} in G and we say that v_{n+1} is reachable from v_1, writing $(v_1 \rightarrow v_{n+1})$.

Here are some new definitions. A node v in a graph $G=(V,E)$ is called a sink, if for every node w in G that is reachable from v, v is also reachable from w. The bottom of a graph is the subset of all nodes that are sinks, i.e., $bottom(G)=\{v \in V \mid \forall w \in V: (v \rightarrow w) \Rightarrow (w \rightarrow v)\}$. You have to calculate the bottom of certain graphs.

Input Specification

The input contains several test cases, each of which corresponds to a directed graph G. Each test case starts with an integer number v, denoting the number of vertices of $G=(V,E)$, where the vertices will be identified by the integer numbers in the set $V=\{1,\ldots,v\}$. You may assume that $1 \le v \le 5000$. That is followed by a non-negative integer e and, thereafter, e pairs of vertex identifiers $v_1,w_1,\ldots,v_e,w_e$ with the meaning that $(v_i,w_i) \in E$. There are no edges other than specified by these pairs. The last test case is followed by a zero.

Output Specification

For each test case output the bottom of the specified graph on a single line. To this end, print the numbers of all nodes that are sinks in sorted order separated by a single space character. If the bottom is empty, print an empty line.

Sample Input

```
3 3
1 3 2 3 3 1
2 1
1 2
0
```

Sample Output

```
1 3
2
```

Added by: Wanderley Guimaraes
Date: 2007-09-21
Time limit: 1s
Source limit: 50000B
Languages: All
Resource: University of Ulm Local Contest 2003

SPOJ Problem Set (classical)

1800. Fixed Partition Contest Management

Problem code: CONTEST

A technique used in early programming contest strategies involved partitioning the available intellectual capacity of a team into a number of members with each member having a fixed amount of intelligence, different members potentially having different amounts. The sum of the brightness of all members equals the total intellectual capacity of the team.

Given a set of problems, it was the task of the team to assign the problems to different team members, so that they could be solved concurrently. This was made difficult due to the fact that the solution time of a problem might depend on the amount of intelligence available to it. Every problem has a minimum intelligence requirement, but if assigned to a brighter member its solution time might increase or decrease.

In this task, you have to determine optimal assignments of problems to team members. Your program is given the intellectual capacities of the team members available for the solution of problems, and for each problem a description of how its solution time depends on the amount of intelligence available to it. Your program has to find the solution schedule of the problems that minimizes the average solution time for the problems. A solution schedule is an assignment of problems to team members and times, such that no two problems use the same member at the same time, and no problem is assigned to a team member with less brightness than its minimum requirement. The solution time of the problem is the difference between the time when the problem was submitted to be solved (which is the start of the contest at time zero for all problems in this task), and the time that the problem is solved.

Input Specification

The input data will contain multiple test cases. Each test case begins with a line containing a pair of integers `m` and `n`. The number `m` specifies the number of team members (`1 <= m <= 3`), and `n` specifies the number of problems to be solved (`1 <= n <= 10`).

The next line contains `m` positive integers giving the intelligence amounts of the `m` team members. Following this are `n` lines, describing the time-brightness tradeoffs for each of the `n` problems. Each line starts with a positive integer `k` (`k <= 10`), followed by `k` pairs of positive integers $s_1, t_1, s_2, t_2, \ldots, s_k, t_k$ that satisfy $s_i < s_{i+1}$ for $1 \le i < k$. The minimum intelligence requirement of the problem is s_1, i.e. it cannot be solved by a member with less intellectual capacity than this number. If the problem is solved by a team member with brightness `s`, where $s_i \le s < s_{i+1}$ for some `i`, then its solution time will be t_i. Finally, if the problem is solved by a team member with intellectual capacity s_k or more, then its execution time will be t_k.

A pair of zeroes will follow the input for the last test case.

You may assume that each problem will be solved in exactly the time specified for the given brightness, regardless of the number of other problems being solved by other team members at the same time. No problem will have an intelligence requirement larger than that of the brightest team member.

Output Specification

For each test case, first display the case number (starting with `1` and increasing sequentially). Then print the minimum average solution time for the set of problems with two digits to the right of the decimal point. Follow this by the description of a solution schedule that achieves this average solution time. Display one line for each problem, in the order they were given in the input, that identifies the problem number, the member used to solve it (numbered in the order given in the input), the time when the member started to solve the problem, and the time when the problem was solved. Follow the format shown in the sample output, and print a blank line after each test case.

Sample Input

```
2 4
40 60
1 35 4
1 20 3
1 40 10
1 60 7
3 5
10 20 30
2 10 50 12 30
2 10 100 20 25
1 25 19
1 19 41
2 10 18 30 42
0 0
```

Sample Output

```
Case 1
Average solution time = 7.75
Problem 1 is solved by member 2 from 0 to 4
Problem 2 is solved by member 1 from 0 to 3
Problem 3 is solved by member 1 from 3 to 13
Problem 4 is solved by member 2 from 4 to 11

Case 2
Average solution time = 35.40
Problem 1 is solved by member 3 from 19 to 49
Problem 2 is solved by member 2 from 0 to 25
Problem 3 is solved by member 3 from 0 to 19
Problem 4 is solved by member 2 from 25 to 66
Problem 5 is solved by member 1 from 0 to 18
```

Added by: Wanderley Guimaraes
Date: 2007-09-21
Time limit: 2s
Source limit: 50000B
Languages: All
Resource: University of Ulm Local Contest 2003

SPOJ Problem Set (classical)

1801. Drink, on Ice

Problem code: DRINK

A good drink is always served on ice. That said, the amount of ice is what makes the difference. If it is too much, the drink will be well cooled, however, this is a bit of fraud as there could be less ice (and more Vodka for example). On the other hand, if there is too little ice the drink is warm which is unacceptable. You are to help the bartender, of course neither with mixing nor drinking, but with calculating the expected outcome of such mixtures.

To make things easier, we assume that pure water is mixed with ice in a closed system, i.e., there is no problem with the outside temperature or the warming of the bottle, etc. Therefore, after a some time has passed, the system may be regarded as balanced (there is no further change in temperature and no more melting or freezing). Your job is to calculate the final temperature of this balanced system and the amount of ice and water in this equilibrium state.

As you know from physics, it takes 4.19 Joule to heat one gram of water one Kelvin, whereas it takes 2.09 Joule if it is ice. We define the capacities $c_w = 4.19$ J/(g*K) and $c_i = 2.09$ J/(g*K). Melting one gram of ice takes 335 Joule, where the temperature remains constant at zero. We define the constant $e_m = 335$ J/g. The total thermal energy of the ice and the water before the experiment is equal to the thermal energy of the final mixture.

The figure below shows the energy of one gram of ice, ice-water-mixture, or water, where the temperature is measured relative to -30 degrees Celsius. The jump at 0 degrees represents the melting of ice to water. The amount of energy gained is proportional to the amount of ice already melted.

Input Specification

The input contains several test cases. Each test case consists of four real numbers m_w, m_i, t_w, t_i. The mass of water m_w and the mass of ice m_i are both non-negative, given in grams, and $m_w + m_i > 0$. The water temperature t_w and the ice temperature t_i follow, both given in degrees Celsius, and you may assume that $-30 < t_i <= 0 <= t_w < 100$. The last test case is followed by four zeroes.

Output Specification

For each test case output the amount of ice and water in grams and the final temperature of the mixture in degrees Celsius. All numbers must be rounded to one digit. Adhere to the sample output for the exact format to use.

Sample Input

[IMAGE]

```
100 20 50 -10
100 22 0  0
100 35 25 -10.5
10  90 25 -28
0 0 0 0
```

Sample Output

```
0.0 g of ice and 120.0 g of water at 27.5 C
22.0 g of ice and 100.0 g of water at 0.0 C
6.0 g of ice and 129.0 g of water at 0.0 C
100.0 g of ice and 0.0 g of water at -4.2 C
```

Added by: Wanderley Guimaraes
Date: 2007-09-21
Time limit: 1s
Source limit:50000B
Languages: All
Resource: University of Ulm Local Contest 2003

SPOJ Problem Set (classical)

1802. Edge

Problem code: EDGE

For products that are wrapped in small packings it is necessary that the sheet of paper containing the directions for use is folded until its size becomes small enough. We assume that a sheet of paper is rectangular and only folded along lines parallel to its initially shorter edge. The act of folding along such a line, however, can be performed in two directions: either the surface on the top of the sheet is brought together, or the surface on its bottom. In both cases the two parts of the rectangle that are separated by the folding line are laid together neatly and we ignore any differences in thickness of the resulting folded sheet.

After several such folding steps have been performed we may unfold the sheet again and take a look at its longer edge holding the sheet so that it appears as a one-dimensional curve, actually a concatenation of line segments. If we move along this curve in a fixed direction we can classify every place where the sheet was folded as either `type A` meaning a clockwise turn or `type V` meaning a counter-clockwise turn. Given such a sequence of classifications, produce a drawing of the longer edge of the sheet assuming 90 degree turns at equidistant places.

Input Specification

The input contains several test cases, each on a separate line. Each line contains a nonempty string of characters `A` and `V` describing the longer edge of the sheet. You may assume that the length of the string is less than 200. The input file terminates immediately after the last test case.

Output Specification

For each test case generate a PostScript drawing of the edge with commands placed on separate lines. Start every drawing at the coordinates (300,420) with the command "`300 420 moveto`". The first turn occurs at (310,420) using the command "`310 420 lineto`". Continue with clockwise or counter-clockwise turns according to the input string, using a sequence of "`x y lineto`" commands with the appropriate coordinates. The turning points are separated at a distance of 10 units. Do not forget the end point of the edge and finish each test case by the commands `stroke` and `showpage`.

You may display such drawings with the `gv` PostScript interpreter, optionally after a conversion using the `ps2ps` utility.

Sample Input

```
V
AVV
```

Sample Output

```
300 420 moveto
310 420 lineto
310 430 lineto
stroke
showpage
300 420 moveto
310 420 lineto
310 410 lineto
320 410 lineto
320 420 lineto
stroke
showpage
```

[IMAGE]

Added by: Wanderley Guimaraes
Date: 2007-09-21
Time limit: 1s
Source limit: 50000B
Languages: All
Resource: University of Ulm Local Contest 2003

SPOJ Problem Set (classical)

1803. Fold

Problem code: FOLD

Read the statement of problem E: Edge to understand how to fold a sheet of paper and how to interpret the input. We define a "stripe" to be a maximally large part of the sheet that has no folding line going through. Since the turns occur at equidistant places, all stripes are congruent.

In this problem you are given the description of the result of performing several folding steps as in problem E: Edge, i.e., in the unfolded state. Additionally, you know that the length of the sheet in its folded state is exactly the length of 1 stripe (again, we ignore thickness).

Find the minimum number of folding steps necessary to generate the described sheet from an initially flat sheet of paper. Note that performing a folding step may create more than one turn in the result because parts of the sheet already overlay due to previous folding steps. When a step is carried out, however, all overlaying parts of the sheet are affected, i.e., it is not allowed to fold, say, only the top three layers.

Finally, note that every result can be obtained by iterating through the turns in a fixed direction and performing a folding step at each turn, thereby accumulating a 1 stripe long stack of all stripes. If `n` is the number of turns in the input description, this procedure in fact requires `n` folding steps, which is not necessarily minimal as can be observed in the sample output.

Input Specification

The input contains several test cases, each on a separate line. Each line contains a nonempty string of characters `A` and `V` describing the longer edge of the sheet. You may assume that the length of the string is less than 200. The input file terminates immediately after the last test case.

Output Specification

For each test case print on a line the minimum number of folding steps required to produce the described sheet of paper.

Sample Input

```
V
AVV
AAVAAVVVAAV
```

Sample Output

```
1
2
4
```

Added by: Wanderley Guimaraes
Date: 2007-09-21
Time limit: 1s
Source limit: 50000B
Languages: All
Resource: University of Ulm Local Contest 2003

SPOJ Problem Set (classical)

1804. Genetic Code

Problem code: GENETIC

The connections between mathematics and biology are complicated. Most of the time they do not run along nice-looking links that merrily join at first glance, but they are abstract and not always easily established.

Lake Vostok - about 14000 square kilometers large, up to 650 meters deep, and covered by 3743 meters of ice - was recently discovered on the Antarctic continent. The lake remained under conditions of high pressure and no sunlight for several millions of years. It is believed that ordinary life has evolved to a more efficient form using a genetic code composed of only three bases (the current state of ignorance proclaims the four bases adenine, cytosine, guanine, and thymine). Until reasonable names are found, the three bases will be abbreviated as `N`, `O`, and `P`.

Moreover, the genome is single-stranded and directed, i.e., we may see it as a sequence over the alphabet `{N,O,P}`. Unless risking instability, it is necessary that the genome is a Thue-sequence, due to the Norwegian mathematician A. Thue (1863-1922). Define a subsegment of a sequence to be a connected subsequence, and call two subsegments adjacent if one follows immediately after the other in the sequence. A Thue-sequence is a sequence where no adjacent subsegments are equal. For example, `NOPNO` is and `NOPNPNO` is not a Thue-sequence, so that the first may be a genome whereas the second may not.

To be able to simulate experiments with the new genomes, you are asked to generate genomes of certain lengths.

Input Specification

The input contains several test cases. Each test case consists of an integer `n`. You may assume that `1<=n<=5000`. The last test case is followed by a zero.

Output Specification

For each test case specified by `n` output on a single line any genome of length `n`. If no genome of length `n` exists, output a blank line instead.

Sample Input

```
1
2
10
20
0
```

Sample Output

```
N
NO
NONPNOPNPO
NONPNOPNPONOPNONPNOP
```

Added by: Wanderley Guimaraes
Date: 2007-09-21
Time limit: 1s
Source limit: 50000B
Languages: All
Resource: University of Ulm Local Contest 2003

SPOJ Problem Set (classical)

1805. Largest Rectangle in a Histogram

Problem code: HISTOGRA

A histogram is a polygon composed of a sequence of rectangles aligned at a common base line. The rectangles have equal widths but may have different heights. For example, the figure on the left shows the histogram that consists of rectangles with the heights 2, 1, 4, 5, 1, 3, 3, measured in units where 1 is the width of the rectangles:

[IMAGE]

Usually, histograms are used to represent discrete distributions, e.g., the frequencies of characters in texts. Note that the order of the rectangles, i.e., their heights, is important. Calculate the area of the largest rectangle in a histogram that is aligned at the common base line, too. The figure on the right shows the largest aligned rectangle for the depicted histogram.

Input Specification

The input contains several test cases. Each test case describes a histogram and starts with an integer n, denoting the number of rectangles it is composed of. You may assume that $1 \le n \le 100000$. Then follow n integers $h_1, \ldots, h_n$, where $0 \le h_i \le 1000000000$. These numbers denote the heights of the rectangles of the histogram in left-to-right order. The width of each rectangle is 1. A zero follows the input for the last test case.

Output Specification

For each test case output on a single line the area of the largest rectangle in the specified histogram. Remember that this rectangle must be aligned at the common base line.

Sample Input

```
7 2 1 4 5 1 3 3
4 1000 1000 1000 1000
0
```

Sample Output

```
8
4000
```

Added by: Wanderley Guimaraes
Date: 2007-09-21
Time limit: 3s
Source limit: 50000B
Languages: All
Resource: University of Ulm Local Contest 2003

SPOJ Problem Set (classical)

1810. Nuclear Plants

Problem code: ORZ

The Great Plain of Algorithmia plays an extremely important role in the agriculture of the Bandulu Kingdom: this is the only place where barley (Hordeum vulgare), an essential ingredient of beer, can be produced. Unfortunately, it is not possible to grow barley on the full area of the plain, as several nuclear plants have recently been built, and you cannot grow barley near a nuclear plant (since you do not want to produce giant-size, aggressive, man-eating barley-mutants). Your task is to write a program that determines the size of the area that can be used for growing barley.

The Great Plain of Algorithmia is an n * m km^2 rectangle, the coordinates of the four corners being (0, 0),(0, m),(n, 0) and (n, m). There are two types of nuclear plants: small and large. You are not allowed to grow barley within 0.58km of a small nuclear plant or within 1.31km of a large nuclear plant.

Input

The input contains several blocks of test cases. Each block begins with a line containing four integers: 1 <= n, m <= 10000 describe the size of the plain, ks <= 100 is the number of small nuclear plants, and kl <= 100 is the number of large nuclear plants. The next ks lines describe the coordinates of the small nuclear plants, each line contains two integers 0 <= x <= n and 0 <= y <= m . The next kl lines describe the large nuclear plants in a similar fashion.

The input is terminated by a block with n = m = ks = kl = 0 .

Output

For each test case, you have to output a single line containing the area that can be used for growing barley. This number should be a real value with two digits of precision. To avoid rounding problems, we accept solutions with a maximum of 0.01(positive or negative) error.

Example

Input:

```
10 10 2 2
2 2
4 4
5 6
1 8
10 10 1 0
5 5
0 0 0 0
```

Output:

```
87.46
98.94
```

Time limit has been changed from 2 seconds to 5 seconds.

Added by: Blue Mary
Date: 2007-09-23
Time limit: 5s
Source limit: 50000B
Languages: All except: C99 strict
Resource: ACM Central European Programming Contest, Budapest 2005

SPOJ Problem Set (classical)

1811. Longest Common Substring

Problem code: LCS

A string is finite sequence of characters over a non-empty finite set S.

In this problem, S is the set of lowercase letters.

Substring, also called factor, is a consecutive sequence of characters occurrences at least once in a string.

Now your task is simple, for two given strings, find the length of the longest common substring of them.

Here common substring means a substring of two or more strings.

Input

The input contains exactly two lines, each line consists of no more than 250000 lowercase letters, representing a string.

Output

The length of the longest common substring. If such string doesn't exist, print "0" instead.

Example

Input:
```
alsdfkjfjkdsal
fdjskalajfkdsla
```

Output:
```
3
```

Notice: new testcases added

Added by: Jin Bin
Date: 2007-09-24
Time limit: 2s
Source limit:50000B
Languages: All except: C99 strict

SPOJ Problem Set (classical)

1812. Longest Common Substring II

Problem code: LCS2

A string is finite sequence of characters over a non-empty finite set S.

In this problem, S is the set of lowercase letters.

Substring, also called factor, is a consecutive sequence of characters occurrences at least once in a string.

Now your task is a bit harder, for some given strings, find the length of the longest common substring of them.

Here common substring means a substring of two or more strings.

Input

The input contains at most 10 lines, each line consists of no more than 100000 lowercase letters, representing a string.

Output

The length of the longest common substring. If such string doesn't exist, print "0" instead.

Example

Input:
```
alsdfkjfjkdsal
fdjskalajfkdsla
aaaajfaaaa
```

Output:
```
2
```

Notice: new testcases added

Added by: Jin Bin
Date: 2007-09-24
Time limit: 2s
Source limit: 50000B
Languages: All except: C99 strict

SPOJ Problem Set (classical)

1815. Problems Collection (Volume X)

Problem code: WA

These ten problems come from Chinese National Olympiad in Mathematics - Province Contest.

Problem 1 Polynomial $P(x)=x^5+a_1x^4+a_2x^3+a_3x^2+a_4x+a_5$, and we know when k=1, 2, 3, 4, P(k)=2007*k. Calculate P(10)-P(-5).

Problem 2 The sum of 100 positive integers a_1, a_2, ..., a_{100} is 2007. Calculate the maximum possible value of [IMAGE] $a_i*a_j*a_k$.

Problem 3 Calculate 100101102103104.....498499500 modudo 126.

Problem 4 We define the sum of the first n numbers of geometric progression $\{a_n\}$ S_n. Now we know S_7=7, S_{14}=2014. Calculate $S_7*(S_{21}-S_{14})$.

Problem 5 Calculate the sum of this kind of positive integers n(n>=4): n satisfies that n! can be written as the product of n-3 consecutive positive integers.

Problem 6 Two vertexes of a square are on the beeline y=2x-17, and the other two are on the parabola $y=x^2$. Calculate the sum of two different possible values of the area of this square.

Problem 7 A, B, C, D are four certain points in the space and they are not on the same plane. Calculate the number of different parallelepipeds, which satisfies that 4 vertexes of the parallelepiped are A, B, C and D.

Problem 8 Polynomial x^2-x-1 exactly divides Polynomial $a_1x^{17}+a_2x^{16}+1$. Calculate a_1*a_2.

Problem 9 Suppose x is an acute angle, calculate the minimum possible value of (*sin* x +*cos* x)/(*sin* x +*tan* x) + (*tan* x +*cot* x)/(*cos* x +*tan* x) + (*sin* x +*cos* x)/(*cos* x +*cot* x) + (*tan* x +*cot* x)/(*sin* x +*cot* x).

Problem 10 Suppose $x^4+y^4+z^4$=m/n, x, y, z are all real numbers and satisfy x*y+y*z+z*x=1 and 5*(x+1/x)=12*(y+1/y)=13*(z+1/z), and m, n are positive integers and their greatest common divisor is 1. Calculate m+n.

Input

There is no input.

Output

Ten lines, each contains a single integer denoted the answer to the correspoding problem.

Example

There is no example.

Links

If there is any problem, please contact me.

Added by: Blue Mary
Date: 2007-09-27
Time limit: 1s
Source limit:80B
Languages: TEXT
Resource: CMOP :)

SPOJ Problem Set (classical)

1825. Free tour II

Problem code: FTOUR2

After the success of 2nd anniversary (take a look at problem **FTOUR** for more details), this 3rd year, Travel Agent SPOJ goes on with another discount tour.

The tour will be held on *ICPC* island, a miraculous one on the Pacific Ocean. We list **N** places (indexed from 1 to **N**) where the visitors can have a trip. Each road connecting them has an *interest value*, and this value can be *negative* (if there is nothing interesting to view there). Simply, these **N** places along with the roads connecting them form a *tree structure*. We will choose *two places* as the departure and destination of the tour.

Since September is the festival season of local inhabitants, some places are extremely crowded (we call them *crowded places*). Therefore, the organizer of the excursion hopes the tour will visit *at most **K** crowded places* (too tiring to visit many of them) and of course, the *total number of interesting value* should be maximum.

Briefly, you are given a map of **N** places, an integer **K**, and **M** id numbers of *crowded place*. Please help us to find the optimal tour. Note that we can visit each place only *once* (or our customers easily feel bored), also the departure and destination places *don't need to be different*.

Input

There is exactly one case. First one line, containing 3 integers **N K M**, with 1 <= **N** <= 200000, 0 <= **K** <= **M**, 0 <= **M** <= **N**.

Next **M** lines, each line includes an id number of a *crowded place*.

The last (**N** - 1) lines describe (**N** - 1) two-way roads connected **N** places, form **a b i**, with **a, b** is the id of 2 places, and **i** is its *interest value* (-10000 <= **i** <= 10000).

Output

Only one number, the maximum total interest value we can obtain.

Example

Input:

```
8 2 3
3
5
7
1 3 1
2 3 10
3 4 -2
4 5 -1
5 7 6
5 6 5
```

```
4 8 3
```

Output:
```
12
```

Explanation

We choose 2 and 6 as the departure and destination place, so the tour will be 2 -> 3 -> 4 -> 5 -> 6, total interest value = 10 + (-2) + (-1) + 5 = 12

*** Added some unofficial cases**

Added by: Thanh-Vy Hua
Date: 2007-09-28
Time limit: 1s-5s
Source limit:50000B
Languages: All
Resource: Adapted from Preslav Le's problem, first used in Bulgarian OI 07

SPOJ Problem Set (classical)

1833. Sudoku

Problem code: SUDOKU2

Oh no! Bill just realized that the sudoku puzzle he had spent the last ten minutes trying to solve essentially was last week's puzzle, only rotated counterclockwise. How cheap! Couldn't the magazine afford to make a new one every week? Of course, he had no way of knowing about this before he started to solve it, as the holes to fill with digits were other than last week. Nevertheless, realizing that this week's puzzle was a simple derivative of last week's certainly took the fun out of solving the rest of it.

[IMAGE]

The sudoku board consists of 9×9 cells. These can be grouped into 3×3 *regions* of 3×3 cells each. Some of the cells are filled with a digit 1 through 9 while the rest of them are left empty. The aim of the game is to fill each empty cell with a digit 1...9 so that every row, every column and every region contains each of the numbers 1...9 exactly once. A proper sudoku puzzle always has exactly one solution.

Help Bill avoid unpleasant surprises by creating a program that checks whether an unsolved sudoku puzzle is in fact derived from an earlier puzzle by simple operations.

The allowed operations are:

- Rotating the entire puzzle clockwise or counterclockwise.
- Swapping two columns within a 3×9 column segment.
- Swapping two rows within a 9×3 row segment.
- Swapping entire row or column segments.
- Applying a permutation *f* of the digits 1...9 to every cell (i.e. replace *x* by *f(x)* in every cell).

An operation is considered being performed on the sudoku solution (rather than on the unsolved puzzle) and always guarantees that if the board before the transformation was a solution to a sudoku puzzle, it still is afterwards.

Input

The input starts with the number of test cases $0 <= N <= 50$ on a single line.

Then for every test case follow nine lines describing last week's puzzle solution, from top to bottom. Each line corresponds to a row in the puzzle and consists of nine digits (1...9), describing the contents of the cell from left to right.

Last week's solution is followed by nine lines describing this week's unsolved puzzle. Here, also, every line corresponds to a puzzle row and every digit (0...9) describes the contents of a cell. 0 indicates that the cell is empty. The rows are presented ordered from top to bottom, and within each row, the cells are ordered from left to right.

After every test case except the last one follows a blank line. Every unsolved puzzle is guaranteed to be uniquely solvable and last week's solution is always a proper sudoku solution. /p>

Output

For every test case, output `Yes` if the sudoku puzzle can be derived from the given solved puzzle using the allowed operations, or `No` if this is not possible.

Example

Input:

```
2
963174258
178325649
254689731
821437596
496852317
735961824
589713462
317246985
642598173
060104050
200000001
008305600
800407006
006000300
700901004
500000002
040508070
007206900

534678912
672195348
198342567
859761423
426853791
713924856
961537284
287419635
345286179
010900605
025060070
870000902
702050043
000204000
490010508
107000056
040080210
208001090
```

Output:

```
Yes
No
```

Added by: Robin Nittka
Date: 2007-10-02
Time limit: 10s
Source limit: 50000B
Languages: All
Resource: ACM ICPC NWERC 2006

SPOJ Problem Set (classical)

1835. The SetStack Computer

Problem code: SETSTACK

[IMAGE]

Background from Wikipedia: “Set theory is a branch of mathematics created principally by the German mathematician Georg Cantor at the end of the 19th century. Initially controversial, set theory has come to play the role of a foundational theory in modern mathematics, in the sense of a theory invoked to justify assumptions made in mathematics concerning the existence of mathematical objects (such as numbers or functions) and their properties. Formal versions of set theory also have a foundational role to play as specifying a theoretical ideal of mathematical rigor in proofs.”

Given this importance of sets, being the basis of mathematics, a set of eccentric theorist set off to construct a supercomputer operating on sets instead of numbers. The initial SetStack Alpha is under construction, and they need you to simulate it in order to verify the operation of the prototype.

The computer operates on a single stack of sets, which is initially empty. After each operation, the cardinality of the topmost set on the stack is output. The cardinality of a set S is denoted $|S|$ and is the number of elements in S. The instruction set of the SetStack Alpha is `PUSH`, `DUP`, `UNION`, `INTERSECT`, and `ADD`}.

- `PUSH` will push the empty set {} on the stack.
- `DUP` will duplicate the topmost set (pop the stack, and then push that set on the stack twice).
- `UNION` will pop the stack twice and then push the union of the two sets on the stack.
- `INTERSECT` will pop the stack twice and then push the intersection of the two sets on the stack.
- `ADD` will pop the stack twice, add the first set to the second one, and then push the resulting set on the stack.

For illustration purposes, assume that the topmost element of the stack is

A = { {}, {{}} },

and that the next one is

B = { {}, {{{}}} }.

For these sets, we have $|A| = 2$ and $|B| = 2$. Then:

- `UNION` would result in the set { {}, {{}}, {{{}}} }. The output is 3.
- `INTERSECT` would result in the set { {} }. The output is 1.
- `ADD` would result in the set { {}, {{{}}}, {{},{{}}} }. The output is 3.

Input

An integer 0 <= *T* <= 5 on the first line gives the cardinality of the set of test cases. The first line of each test case contains the number of operations 0 <= *N* <= 2000. Then follow *N* lines each containing one of the five commands. It is guaranteed that the SetStack computer can execute all the commands in the sequence without ever popping an empty stack.

Output

For each operation specified in the input, there will be one line of output consisting of a single integer. This integer is the cardinality of the topmost element of the stack after the corresponding command has executed. After each test case there will be a line with `***` (three asterisks).

Example

Input:

```
2
9
PUSH
DUP
ADD
PUSH
ADD
DUP
ADD
DUP
UNION
5
PUSH
PUSH
ADD
PUSH
INTERSECT
```

Output:

```
0
0
1
0
1
1
2
2
2
***
0
0
1
0
0
***
```

Added by: Robin Nittka
Date: 2007-10-02
Time limit: 12s
Source limit: 50000B
Languages: All
Resource: ACM ICPC NWERC 2006

SPOJ Problem Set (classical)

1837. Pie

Problem code: PIE

[IMAGE]

My birthday is coming up and traditionally I'm serving pie. Not just one pie, no, I have a number *N* of them, of various tastes and of various sizes. *F* of my friends are coming to my party and each of them gets a piece of pie. This should be one piece of one pie, not several small pieces since that looks messy. This piece can be one whole pie though.

My friends are very annoying and if one of them gets a bigger piece than the others, they start complaining. Therefore all of them should get equally sized (but not necessarily equally shaped) pieces, even if this leads to some pie getting spoiled (which is better than spoiling the party). Of course, I want a piece of pie for myself too, and that piece should also be of the same size.

What is the largest possible piece size all of us can get? All the pies are cylindrical in shape and they all have the same height 1, but the radii of the pies can be different.

Input

One line with a positive integer: the number of test cases. Then for each test case:

- One line with two integers *N* and *F* with 1 <= *N, F* <= 10000: the number of pies and the number of friends.
- One line with *N* integers r_i with 1 <= r_i <= 10000: the radii of the pies.

Output

For each test case, output one line with the largest possible volume *V* such that me and my friends can all get a pie piece of size *V*. The answer should be given as a floating point number with an absolute error of at most 10^{-3}.

Example

Input:

```
3
3 3
4 3 3
1 24
5
10 5
1 4 2 3 4 5 6 5 4 2
```

Output:

```
25.1327
3.1416
50.2655
```

Added by: Robin Nittka
Date: 2007-10-02
Time limit: 5s
Source limit: 50000B
Languages: All
Resource: ACM ICPC NWERC 2006

SPOJ Problem Set (classical)

1838. Ticket to Ride

Problem code: TICKET

[IMAGE]

Ticket to Ride is a board game for up to 5 players. The goal of the game is to set up train lines (and to thwart the opponents' attempts at setting up their train lines). At the beginning of play, each player is assigned four train lines. A player may choose to discard as many of these four assignments as she likes. Each assignment has a score, corresponding to its difficulty (so, typically, a train line between e.g. Stockholm and Tokyo would be worth more than a train line between e.g. Stockholm and Utrecht). At the end of the game, each player gets points for the assignments that they have successfully completed, and penalty points for the assignments that they have failed to complete.

An assignment consists of a pair of cities that are to be connected by a series of shorter railway routes. A route can be claimed (for a certain cost associated with the route), but things are complicated by the fact that there is only a limited number of routes, and once a player claims a route, none of the other players can claim it. A player has successfully set up a train line between two cities if there is a path between the two cities using only routes that have been claimed by this player. For simplicity, we will ignore all additional aspects of the game (including the actual process of claiming routes and additional ways to score points).

For instance, if your assignment is to connect Stockholm and Amsterdam in the Figure above, you would probably want to claim the routes between Stockholm and Copenhagen, and between Copenhagen and Amsterdam. But if another player manages to claim the route between Copenhagen and Stockholm before you, your train line would have to use some other routes, e.g. by going to Copenhagen via Oslo.

In this problem, we will consider the rather bold strategy of trying to complete all four assignments (typically, this will be quite hard). As a preliminary assessment of the difficulty of achieving this, we would like to calculate the minimum cost of setting up all four lines assuming that none of the other players interfere with our plans. Your job is to write a program to determine this minimum cost.

Input

The input consists of several (at most 20) games to be analyzed. Each game starts with two integers $1 <= n <= 30$, $0 <= m <= 1000$, giving the number of cities and railway routes in the map, respectively. Then follow n lines, giving the names of the n cities. City names are at most 20 characters long and consist solely of lower case letters (`'a'-'z'`).

After this follow m lines, each containing the names of two different cities and an integer $1 <= c <= 10000$, indicating that there is a railway route with cost c between the two cities. Note that there may be several railway routes between the same pair of cities. You may assume that it is always possible to set up a train line from any city to any other city.

Finally, there will be four lines, each containing the names of two cities, giving the four train line assignments.

The input is terminated by a case where $n = m = 0$. This case should not be processed.

Output

For each game, output a single line containing a single integer, the minimum possible cost to set up all four train lines.

Example

Input:

```
10 15
stockholm
amsterdam
london
berlin
copenhagen
oslo
helsinki
dublin
reykjavik
brussels
oslo stockholm 415
stockholm helsinki 396
oslo london 1153
oslo copenhagen 485
stockholm copenhagen 522
copenhagen berlin 354
copenhagen amsterdam 622
helsinki berlin 1107
london amsterdam 356
berlin amsterdam 575
london dublin 463
reykjavik dublin 1498
reykjavik oslo 1748
london brussels 318
brussels amsterdam 173
stockholm amsterdam
oslo london
reykjavik dublin
brussels helsinki
2 1
first
second
first second 10
first first
first first
second first
first first
0 0
```

Output:

```
3907
10
```

Added by: Robin Nittka
Date: 2007-10-02
Time limit: 20s
Source limit: 50000B
Languages: All
Resource: ACM ICPC NWERC 2006

SPOJ Problem Set (classical)

1839. The Bookcase

Problem code: BOOKCASE

[IMAGE]

No wonder the old bookcase caved under the massive piles of books Tom had stacked on it. He had better build a new one, this time large enough to hold all of his books. Tom finds it practical to have the books close at hand when he works at his desk. Therefore, he is imagining a compact solution with the bookcase standing on the back of the desk. Obviously, this would put some restrictions on the size of the bookcase, it should preferably be as small as possible. In addition, Tom would like the bookcase to have exactly three shelves for aesthetical reasons.

Wondering how small his bookcase could be, he models the problem as follows. He measures the height h_i and thickness t_i of each book i and he seeks a partition of the books in three non-empty sets S_1, S_2, S_3 such that

[IMAGE]

is minimized, i.e. the area of the bookcase as seen when standing in front of it (the depth needed is obviously the largest width of all his books, regardless of the partition). Note that this formula does not give the exact area of the bookcase, since the actual shelves cause a small additional height, and the sides cause a small additional width. For simplicity, we will ignore this small discrepancy.

Thinking a moment on the problem, Tom realizes he will need a computer program to do the job.

Input

The input begins with a positive number on a line of its own telling the number of test cases (at most 20). For each test case there is one line containing a single positive integer N, $3 <= N <= 70$ giving the number of books. Then N lines follow each containing two positive integers h_i, t_i, satisfying $150 <= h_i <= 300$ and $5 <= t_i <= 30$, the height and thickness of book i respectively, in millimeters.

Output

For each test case, output one line containing the minimum area (height times width) of a three-shelf bookcase capable of holding all the books, expressed in square millimeters.

Example

Input:

```
2
4
220 29
195 20
200 9
180 30
6
```

```
256 20
255 30
254 15
253 20
252 15
251 9
```

Output:

```
18000
29796
```

Added by: Robin Nittka
Date: 2007-10-02
Time limit: 40s
Source limit: 50000B
Languages: All
Resource: ACM ICPC NWERC 2006

SPOJ Problem Set (classical)

1840. Printer Queue

Problem code: PQUEUE

[IMAGE]

The only printer in the computer science students' union is experiencing an extremely heavy workload. Sometimes there are a hundred jobs in the printer queue and you may have to wait for hours to get a single page of output.

Because some jobs are more important than others, the Hacker General has invented and implemented a simple priority system for the print job queue. Now, each job is assigned a priority between 1 and 9 (with 9 being the highest priority, and 1 being the lowest), and the printer operates as follows.

- The first job *J* in queue is taken from the queue.
- If there is some job in the queue with a higher priority than job *J*, then move *J* to the end of the queue without printing it.
- Otherwise, print job *J* (and do not put it back in the queue).

In this way, all those important muffin recipes that the Hacker General is printing get printed very quickly. Of course, those annoying term papers that others are printing may have to wait for quite some time to get printed, but that's life.

Your problem with the new policy is that it has become quite tricky to determine when your print job will actually be completed. You decide to write a program to figure this out. The program will be given the current queue (as a list of priorities) as well as the position of your job in the queue, and must then calculate how long it will take until your job is printed, assuming that no additional jobs will be added to the queue. To simplify matters, we assume that printing a job always takes exactly one minute, and that adding and removing jobs from the queue is instantaneous.

Input

One line with a positive integer: the number of test cases (at most 100). Then for each test case:

- One line with two integers *n* and *m*, where *n* is the number of jobs in the queue (1 <= *n* <= 100) and *m* is the position of your job (0 <= m <= n-1). The first position in the queue is number 0, the second is number 1, and so on.
- One line with *n* integers in the range 1 to 9, giving the priorities of the jobs in the queue. The first integer gives the priority of the first job, the second integer the priority of the second job, and so on.

Output

For each test case, print one line with a single integer; the number of minutes until your job is completely printed, assuming that no additional print jobs will arrive.

Example

Input:

```
3
1 0
5
4 2
1 2 3 4
6 0
1 1 9 1 1 1
```

Output:

```
1
2
5
```

Added by:	Robin Nittka
Date:	2007-10-02
Time limit:	2s
Source limit:	50000B
Languages:	All
Resource:	ACM ICPC NWERC 2006

SPOJ Problem Set (classical)

1841. Prime Path

Problem code: PPATH

[IMAGE]

The ministers of the cabinet were quite upset by the message from the Chief of Security stating that they would all have to change the four-digit room numbers on their offices.
-- It is a matter of security to change such things every now and then, to keep the enemy in the dark.
-- But look, I have chosen my number 1033 for good reasons. I am the Prime minister, you know!
-- I know, so therefore your new number 8179 is also a prime. You will just have to paste four new digits over the four old ones on your office door.
-- No, it's not that simple. Suppose that I change the first digit to an 8, then the number will read 8033 which is not a prime!
-- I see, being the prime minister you cannot stand having a non-prime number on your door even for a few seconds.
-- Correct! So I must invent a scheme for going from 1033 to 8179 by a path of prime numbers where only one digit is changed from one prime to the next prime.

Now, the minister of finance, who had been eavesdropping, intervened.
-- No unnecessary expenditure, please! I happen to know that the price of a digit is one pound.
-- Hmm, in that case I need a computer program to minimize the cost. You don't know some very cheap software gurus, do you?
-- In fact, I do. You see, there is this programming contest going on...

Help the prime minister to find the cheapest prime path between any two given four-digit primes! The first digit must be nonzero, of course. Here is a solution in the case above.

```
1033
1733
3733
3739
3779
8779
8179
```

The cost of this solution is 6 pounds. Note that the digit 1 which got pasted over in step 2 can not be reused in the last step - a new 1 must be purchased.

Input

One line with a positive number: the number of test cases (at most 100). Then for each test case, one line with two numbers separated by a blank. Both numbers are four-digit primes (without leading zeros).

Output

One line for each case, either with a number stating the minimal cost or containing the word `Impossible`.

Example

Input:

```
3
1033 8179
1373 8017
1033 1033
```

Output:

```
6
7
0
```

Added by: Robin Nittka
Date: 2007-10-02
Time limit: 3s
Source limit: 50000B
Languages: All
Resource: ACM ICPC NWERC 2006

SPOJ Problem Set (classical)

1842. Lineland Airport

Problem code: LINELAND

Lineland is a strange country. As the name suggests, it's shape (as seen from above) is just a straight line, rather than some two-dimensional shape. The landscape along this line is very mountainous, something which occasionally leads to some problems. One such problem now occurs: in this modern era the king wants to build an airport to stimulate the country's economy. Unfortunately, it's impossible for airplanes to land on steep airstrips, so a horizontal piece of land is needed. To accommodate for the larger airplanes, this strip needs to have length at least *L*.

Over the years, the inhabitants of Lineland have become very proficient in flattening pieces of land. Given a piece a land, they can remove rock quickly. They don't want to add rock for that may lead to an unstable landing strip. To minimize the amount of effort, however, they want to remove the least amount of rock necessary to reach their goal: a flat piece of land of length *L*. What is this minimum amount? Because of the low-dimensional nature of Lineland, the amount of rock that needs to be removed is measured as the total area of land above the place where the landing strip is placed, rather than the volume (so in the Figure below, the amount of land removed is given by the lightly shaded area).

[IMAGE]

Input

One line with a positive number: the number of test cases (at most 25). Then for each test case:

- One line with an integer *N*, 2 <= *N* <= 500, the number of points, and an integer *L*, 1 <= *L* <= 10000, the necessary length to flatten.
- *N* lines with two integers x_i and y_i with 0 <= x_i, y_i <= 10000 describing the landscape of Lineland. The x_i are in (strictly) ascending order. At position x_i the height of the landscape is x_i. Between two x_i the landscape has constant slope. (So the landscape is piecewise linear). The difference between x_N and x_1 is greater than or equal to *L*.

Output

For each test case, output one line with the minimum amount of rock which must be removed in order to build the airport. The answer should be given as a floating point number with an absolute error of at most 10^{-3}.

Example

Input:

```
4
3 5
0 2
4 2
14 0
```

```
4 3
0 2
2 0
4 0
5 3
3 10
10 2
30 2
35 7
2 777
222 333
4444 5555
```

Output:

```
0.9000
0.3750
0.0000
373362.4867
```

Added by: Robin Nittka
Date: 2007-10-02
Time limit: 20s
Source limit: 50000B
Languages: All
Resource: ACM ICPC NWERC 2006

SPOJ Problem Set (classical)

1843. Leonardo Notebook

Problem code: LEONARDO

[IMAGE]

-- I just bought Leonardo's secret notebook!
Rare object collector Stan Ucker was really agitated but his friend, special investigator Sarah Keptic was unimpressed.
-- How do you know it is genuine?
-- Oh, it must be, at that price. And it is written in the da Vinci code.
Sarah browsed a few of the pages. It was obvious to her that the code was a substitution cipher, where each letter of the alphabet had been substituted by another letter.
-- Leonardo would have written the plain-text and left it to his assistant to encrypt, she said. And he must have supplied the substitution alphabet to be used. If we are lucky, we can find it on the back cover!
She turned up the last page and, lo and behold, there was a single line of all 26 letters of the alphabet:

QWERTYUIOPASDFGHJKLZXCVBNM

-- This may be Leonardo's instructions meaning that each A in the plain-text was to be replaced by Q, each B with W, etcetera. Let us see...
To their disappointment, they soon saw that this could not be the substitution that was used in the book. Suddenly, Stan brightened.
-- Maybe Leonardo really wrote the substitution alphabet on the last page, and by mistake his assistant coded that line as he had coded the rest of the book. So the line we have here is the result of applying some permutation TWICE to the ordinary alphabet!
Sarah took out her laptop computer and coded fiercely for a few minutes. Then she turned to Stan with a sympathetic expression.
-- No, that couldn't be it. I am afraid that you have been duped again, my friend. In all probability, the book is a fake.
Write a program that takes a permutation of the English alphabet as input and decides if it may be the result of performing some permutation twice.

Input

The input begins with a positive number on a line of its own telling the number of test cases (at most 500). Then for each test case there is one line containing a permutation of the 26 capital letters of the English alphabet.

Output

For each test case, output one line containing `Yes` if the given permutation can result from applying some permutation twice on the original alphabet string ABC...XYZ, otherwise output `No`.

Example

Input:

```
2
QWERTYUIOPASDFGHJKLZXCVBNM
ABCDEFGHIJKLMNOPQRSTUVWXYZ
```

Output:

```
No
Yes
```

Added by: Robin Nittka
Date: 2007-10-02
Time limit: 2s
Source limit: 50000B
Languages: All
Resource: ACM ICPC NWERC 2006

SPOJ Problem Set (classical)

1845. Mice and Maze

Problem code: MICEMAZE

A set of laboratory mice is being trained to escape a maze. The maze is made up of cells, and each cell is connected to some other cells. However, there are obstacles in the passage between cells and therefore there is a time penalty to overcome the passage Also, some passages allow mice to go one-way, but not the other way round.

Suppose that all mice are now trained and, when placed in an arbitrary cell in the maze, take a path that leads them to the exit cell in minimum time.

We are going to conduct the following experiment: a mouse is placed in each cell of the maze and a count-down timer is started. When the timer stops we count the number of mice out of the maze.

Problem

Write a program that, given a description of the maze and the time limit, predicts the number of mice that will exit the maze. Assume that there are no bottlenecks is the maze, i.e. that all cells have room for an arbitrary number of mice.

Input

The maze cells are numbered $1,2,\ldots,N$, where N is the total number of cells. You can assume that $N\leq 100$.

The first three input lines contain N , the number of cells in the maze, E , the number of the exit cell, and the starting value T for the count-down timer (in some arbitrary time unit).

The fourth line contains the number M of connections in the maze, and is followed by M lines, each specifying a connection with three integer numbers: two cell numbers a and b (in the range $1,\ldots,N$) and the number of time units it takes to travel from a to b .

Notice that each connection is one-way, i.e., the mice can’t travel from b to a unless there is another line specifying that passage. Notice also that the time required to travel in each direction might be different.

Output

The output consists of a single line with the number of mice that reached the exit cell E in at most T time units.

Example

Input:

```
4
2
1
8
1 2 1
1 3 1
2 1 1
2 4 1
3 1 1
3 4 1
4 2 1
4 3 1
```

Output:

```
3
```

Added by:	Robin Nittka
Date:	2007-10-04
Time limit:	2s
Source limit:	50000B
Languages:	All
Resource:	ACM ICPC -- SWERC 2001

SPOJ Problem Set (classical)

1846. Project File Dependencies

Problem code: PFDEP

Project managers, such as the UNIX utility `make`, are used to maintain large software projects made up from many components. Users write a *project file* specifying which components (called *tasks*) depend on others and the project manager can automatically update the components in the correct order.

Problem

Write a program that reads a project file and outputs the order in which the tasks should be performed.

Input

For simplicity we represent each task by an integer number from $1,2,\ldots,N$ (where N is the total number of tasks). The first line of input specifies the number N of tasks and the number M of rules, such that $N \leq 100,\; M\leq 100$.

The rest of the input consists of M *rules*, one in each line, specifying dependencies using the following syntax:

```
$T_0$      $k$      $T_1$      $T_2$      ...      $T_k$
```

This rule means that task number T_0 depends on k tasks $T_1, T_2, \ldots T_k$ (we say that task T_0 is the *target* and $T_1\ldots T_k$ are *dependents*).

Note that tasks numbers are separated by single spaces and that rules end with a newline. Rules can appear in any order, but each task can appear as target only once.

Your program can assume that there are no circular dependencies in the rules, i.e. no task depends directly or indirectly on itself.

Output

The output should be a single line with the permutation of the tasks $1\ldots N$ to be performed, ordered by dependencies (i.e. no task should appear before others that it depends on).

To avoid ambiguity in the output, tasks that do not depend on each other should be ordered by their number (lower numbers first).

Example

Input:

```
5 4
3 2 1 5
2 2 5 3
4 1 3
```

```
5 1 1
```

Output:

```
1 5 3 2 4
```

Added by:	Robin Nittka
Date:	2007-10-04
Time limit:	2s
Source limit:	50000B
Languages:	All
Resource:	ACM ICPC -- SWERC 2001

SPOJ Problem Set (classical)

1847. No Change

Problem code: NOCHANGE

Though it might be hard to imagine, the inhabitants of a small country Additivia do not know of such thing as change, which probably has to do with them not knowing subtraction either. When they buy something, they always need to have the exact amount of addollars, their currency. The only other option, but not a really attractive one, is over-paying.

Professor Adem, one of the Additivian mathematicians came up with an algorithm for keeping a balanced portfolio. The idea is the following. Suppose you have more coins of value v_1 than coins of value v_2 . In this case you should try to spend at least as many coins of value v_1 as those of value v_2 on any buy you make. Of course spending too many v_1 coins is not a good idea either, but to make the algorithm simpler professor Adem decided to ignore the problem. The algorithm became an instant hit and professor Adem is now designing a kind of ‘‘electronic portfolio’’ with built-in Adem’s algorithm. All he needs now is a software for these machines, that will decide whether a given amount of addollars can be paid using a given set of coins according to the rules of Adem’s algorithm. Needless to say, you are his chosen programmer for the task.

Problem

Write a program that reads the description of a set of coins and an amount of addollars to be paid, and determines whether you can pay that amount according to Professor Adem’s rules.

Input

The input starts with the amount of addollars to be paid x , where $1\le x \le 100 000$. The number of different coin values k follows, where $1 \le k \le 5$. The values of the coins $v_1,\ldots,v_k$ follow, where $1 \le v_i \le 10 000$.

Notice that the order among coin values is significant: you need to spend at least as many coins of value v_1 as coins of value v_2 , at least as many coins of value v_2 as those of value v_3 , and so on. You may assume that you have a sufficiently large number of coins of each value.

Output

Your program should output for each test case either a single word ‘‘YES’’, if the given amount can be paid according to the rules, or a single word ‘‘NO’’ otherwise.

Example

Input:
```
13 3 9 2 1
```
Output:
```
NO
```

Added by: Robin Nittka
Date: 2007-10-04
Time limit: 5s
Source limit: 50000B
Languages: All
Resource: ACM ICPC -- SWERC 2001

SPOJ Problem Set (classical)

1865. Making Waves

Problem code: MKWAVES

Suppose we know a signal is generated by the function x sin ($f_1 t$) sin ($f_2 t$). F1 and f2 are two unique frequencies, each having an integral value in the range 400 to 600 Hz (Hz = cycles per second), and t represents time. In this problem, you will be given N samples of this function at equally-spaced time intervals corresponding to t = 1/N, t = 2/N, and so forth. From these samples, you are to determine f1 and f2.

For example, suppose f1 = 400 Hz, f2 = 500 Hz, and N = 100. The first sample, at time t = 1/100 sec, is equal to sin(400 0.01) sin(500 0.01) sin(4) sin(5) 0.156912 . Similarly, the second sample, at time t = 2/100 sec, is equal to sin(400 0.02) sin(500 0.02) sin(8) sin(10) 0.312821.

Input

There will be multiple cases to consider. Each case begins with an integer N, no larger than 1000, that specifies the number of signal samples. The next N data items are real numbers representing the signal samples at time 1/N sec, 2/N sec, and so forth. A single integer 0 follows the last case.

The number of samples for each case is guaranteed to be sufficient to allow the correct result to be obtained.

Output

For each input case, display a single line that is formatted like this:

Case 1, f1 = 400, f2 = 500

Example

Input:

```
100
0.156912 0.312821 0.466731 0.617657 0.764638 0.906737 1.04305 1.17271
1.29489 1.40883 1.51381 1.60917 1.69432 1.76873 1.83195 1.8836 1.92338
1.95106 1.96649 1.96962 1.96045 1.93908 1.9057 1.86055 1.80396 1.73634
1.65816 1.56997 1.47237 1.36603 1.25166 1.13003 1.00196 0.868307 0.729943
0.587785 0.442764 0.295823 0.147918 1.68756e-010 -0.146981 -0.292088
-0.434403 -0.573031 -0.707107 -0.835801 -0.958325 -1.07394 -1.18195
-1.28171 -1.37266 -1.45428 -1.52611 -1.58779 -1.63898 -1.67947 -1.70907
-1.7277 -1.73535 -1.73205 -1.71795 -1.69323 -1.65816 -1.61308 -1.55838
-1.49452 -1.42201 -1.34141 -1.25334 -1.15846 -1.05745 -0.951057 -0.840028
-0.725146 -0.607206 -0.487017 -0.365392 -0.243145 -0.121082 -2.75143e-010
0.119322 0.236125 0.34968 0.459289 0.564288 0.664055 0.758014 0.845635
0.926438 1 1.06595 1.12398 1.17384 1.21533 1.24833 1.27276 1.28862
1.29596 1.29489 1.28558
```

```
0
```

Output:

```
Case 1, f1 = 400, f2 = 500
```

Added by: Camilo Andrés Varela León
Date: 2007-10-07
Time limit: 1s
Source limit: 50000B
Languages: All
Resource: North Central North America Regional Programming Contest - 2003

SPOJ Problem Set (classical)

1866. Making Pals

Problem code: MKPALS

A *palindrome* is a sequence that is the same when read forward or backward. For example, "pop" is a palindrome, as are "Poor Dan is in a droop" (ignoring spaces and case), and "12321".

In this problem, you are to find the "cheapest" way to transform a sequence of decimal digits into a palindrome. There are only two types of modifications you may make to the sequence, but each of these may be repeated as many times as necessary. You may delete a digit from either end of the sequence, or you may add a digit to either end of the sequence. Each of these operations incurs a "cost" of 1. For each input sequence, determine the smallest cost of transforming the sequence into a palindrome, and the length of the resulting palindrome. If two palindromes can be produced with the same cost, the length of the longer palindrome (the one with more digits) is to be reported.

For example, suppose the initial sequence was "911". This can be transformed into a palindrome by deleting the leading "9" (yielding "11") or by adding an additional "9" to the right end of the sequence (yielding "9119"). Since both of these transformations have a cost of 1, and the second transformation yields a longer palindrome, it is this one which would be reported as your result.

Note that the particular palindrome produced by the cheapest sequence of transformations is not necessarily unique, but since you are not required to report the resulting palindrome, any of these will suffice.

Input

There will be multiple cases to consider. Each case has a single line of input that contains one or more decimal digits followed by the end of line. The maximum number of digits in a sequence will be 6. The last case is followed by an empty line (that is, only an end of line).

Output

For each input case, display the case number (1, 2, ...), the input sequence, the cost of the cheapest transformation, and the length of the resulting palindrome. Your output should follow the format shown in the examples below.

Example

Input:

```
911
9118
11234
<-- This line is blank
```

Output:

```
Case 1, sequence = 911, cost = 1, length = 4
Case 2, sequence = 9118, cost = 2, length = 4
Case 3, sequence = 11234, cost = 3, length = 8
```

Added by: Camilo Andrés Varela León
Date: 2007-10-07
Time limit: 1s
Source limit:50000B
Languages: All
Resource: North Central North America Regional Programming Contest - 2003

SPOJ Problem Set (classical)

1868. Making Money

Problem code: MKMONEY

A trick sometimes used by parents to teach their children the value of money is to give then a penny - just a penny! - and the promise that for each day they don't spend it, the parent will double it. All students of computing know that long before a month has elapsed without spending a cent, the parents will not likely be able to make good on their promise.

100-percent compound daily interest on an investment is, of course, unattainable in normal financial dealings, but we are all continually reminded of the power of compound interest, even with the relatively low interest rates available today.

But exactly how much money can be made with compound interest? Assume, for example, an initial investment of $100.00 (US or Canadian :-)), an annual interest rate of 6.00 percent, and that interest is compounded monthly. That is, the interest earned during the preceding month is added to the principal at the end of the month. (For our purposes, we'll assume a month is exactly 1/12th of a year.)

At the end of the first month, the money will have earned 0.5 percent interest (1/12th of 6.00 percent), or $0.50. This is added to the $100.00 invested, so that during the next month, interest is paid on $100.50. During the next month another 0.5 percent interest is earned, which is exactly $0.5025. We will assume that the bank, being conservative, will not pay any interest less than $0.01, so our investment is credited with an additional $0.50 at the end of the second month, for a whopping total of $101.00. Continuing in the same manner, at the end of 12 months our investment will total $106.12, $0.12 more than simple 6.00 percent interest for a year with no compounding.

Given an amount P to be invested for a year with I percent interest, compounded C times during the year at equal intervals, what is total return on the investment?

Input

There will be multiple cases to consider. The input for each case is a single line containing the initial investment amount, P, given in dollars and cents (but no fractional cents, and no larger than $100,000.00), the annual interest rate (I) given as a real number with two fractional digits representing a percentage, greater than zero but less than 100, and the number of compounding intervals per year (C), an integer between 1 and 365. The last case will be followed by a line containing "0.00 0.00 0".

Output

For each input case, display the case number (1, 2, ...), the initial investment (P), the annual interest rate (I), the number of compounding intervals per year(C), and the value of the investment at the end of a year. Your output should follow the format shown in the examples below.

Example

Input:

```
100.00 6.00 1
100.00 6.00 12
1000.00 6.00 12
0.00 0.00 0
```

Output:

```
Case 1. $100.00 at 6.00% APR compounded 1 times yields $106.00
Case 2. $100.00 at 6.00% APR compounded 12 times yields $106.12
Case 3. $1000.00 at 6.00% APR compounded 12 times yields $1061.63
```

Added by: Camilo Andrés Varela León
Date: 2007-10-07
Time limit: 1s
Source limit: 50000B
Languages: All
Resource: North Central North America Regional Programming Contest - 2003

SPOJ Problem Set (classical)

1869. Making Mountains Out Of Molehills

Problem code: MKMOOM

A macro processor is a symbol processing program. It takes a stream of characters as its input, and produces a stream of characters as its output. A "macro" is similar to ordinary function in that it has a definition, and when invoked, that definition is applied to the argument(s) to yield the result. In this problem, you will develop a simple macro processor.

A "macro call" consists of a name and a list of arguments, separated by commas. The name is preceded by '[' (a left bracket) and the last argument is followed by ']' (a right bracket). For example, "[doit,to,it]" calls the macro named "doit" with two arguments, "to" and "it". "[random]" calls the macro named "random" with no arguments.

Before a macro can be called, it must be defined by associating its name with a symbol string. This definition string may contain the special constructions "$1" through "$9" to reference the first through the ninth macro parameters. "$0" references the macro's name itself. When the macro is called, these constructions are literally replaced by the values of the parameters. For example, suppose the definition string for the macro named "321" was "$3-$2-$1". The macro call "[321,This,is,fun]" would yield the output "fun-is-This". A macro call can appear anywhere. For example, the macro call "[321,[321,A,B,C],D,E]" would yield the output "E-D-C-B-A".

Input enclosed in '<' and '>' prevents the evaluation of the text enclosed, allowing special characters like '[', ']', ',' and '$' to be used in other than their usual contexts. Thus the macro call "[321,<$>,<[>,<,>]" would yield ",-[-$".

Macros are defined using the predefined macro named "def", which has two arguments. The first argument is the name of the macro being defined, and the second argument is the defining symbol string for the macro. The "321" macro definition is "[def,321,<$3-$2-$1>]". Note that the definition is enclosed in '<' and '>' to prevent "$1", "$2", and "$3" from being interpreted as parameter references to def. The def macro produces no output. Naturally, the def macro isn't defined using def, but is treated specially by the implementation.

Processing

The input stream is processed character by character and copied to the output until a macro call is encountered, or the input is exhausted (which terminates processing). A macro call is evaluated as described below, with the result (if any) copied to the output.

1. The macro name and the parameters are evaluated in sequence from left to right. This may require evaluating additional macro calls, which must be processed recursively.
2. When the argument list is complete (that is, when the closing ']' is encountered) the definition of the macro being called is scanned in the same manner as the original input stream except that occurrences of "$0", "$1", and so forth are replaced literally by the corresponding arguments. The result of the macro call is the symbol stream produced by this scan.
3. When the macro call is completed, the macro name and the arguments are discarded, and processing resumes at the point where it was interrupted by the macro call.

Limits and Caveats

Macro names and arguments will contain no more than 32 characters each. The defining string for a macro will contain no more than 100 characters. Macros will never be defined more than once (that is, the same macro name will not be used more than once as the first argument to "def"). Macro calls will always provide the correct number of arguments. Character case is significant in comparisons. All input characters, including end of line characters, are to be processed through the macro processor. No output line will contain more than 80 characters, including the end of line character. The input is guaranteed to be correct.

Input

There will be multiple cases to consider. The input for each case begins with a line containing a single integer between 1 and 10, that specifies the number of lines of text immediately following that will be used as input to the macro processor. None of these lines will contain more than 80 characters, so the input to the macro processor will contain at most 810 characters. The last case will be followed by a line containing the integer 0.

Output

For each input case, display the case number (1, 2, ...), a line containing 79 hyphens, the output from the macro processor, another line containing 79 hyphens, and a blank line.

In the sample input shown below, assume that the last visible character on each line is immediately followed by the end of line character. Blank lines in the expected output are shown here as **BLANK** for clarity, but these should actually be totally blank in your output.

Example

Input:

```
1
This is just copied (including end of line).
1
[def,321,<$3-$2-$1>][321,This,is,fun]
1
[def,321,<$3-$2-$1>][321,[321,A,B,C],D,E]
1
[def,321,<$3-$2-$1>][321,This,is,fun][321,[321,A,B,C],D,E]
3
[def,321,<$3-$2-$1>]
[321,This,is,fun]
[321,[321,A,B,C],D,E]
3
[def,A,<$1[B]$2>]
[def,B,*B*]
[A,1,2]
2
[def,#,<[-]$1[-]$2[-]$3[-]>][def,-,<$0>][def,DEF,def][[DEF],X,THIS IS X]
[X][#,DEF,#,X]
0
```

Output:

```
Case 1
--------------------------------------------------------------------------------
This is just copied (including end of line).
```

```
--------------------------------------------------------------------------------
**BLANK**
Case 2
--------------------------------------------------------------------------------
fun-is-This
--------------------------------------------------------------------------------
**BLANK**
Case 3
--------------------------------------------------------------------------------
E-D-C-B-A
--------------------------------------------------------------------------------
**BLANK**
Case 4
--------------------------------------------------------------------------------
fun-is-ThisE-D-C-B-A
--------------------------------------------------------------------------------
**BLANK**
Case 5
--------------------------------------------------------------------------------
**BLANK**
fun-is-This
E-D-C-B-A
--------------------------------------------------------------------------------
**BLANK**
Case 6
--------------------------------------------------------------------------------
**BLANK**
**BLANK**
1*B*2
--------------------------------------------------------------------------------
**BLANK**
Case 7
--------------------------------------------------------------------------------
**BLANK**
THIS IS X-DEF-#-X-
--------------------------------------------------------------------------------
**BLANK**
```

Added by: Camilo Andrés Varela León
Date: 2007-10-07
Time limit: 1s
Source limit:50000B
Languages: All
Resource: North Central North America Regional Programming Contest - 2003

SPOJ Problem Set (classical)

1870. Making Labels

Problem code: MKLABELS

Trees comes in many varieties other than the popular binary tree. In general, a tree is a connected acyclic graph. That is, it consists of some number of vertices N (which we'll assume is at least one in this problem), and N - 1 edges, each of which connects a pair of vertices.

A "labeled tree" is a tree in which each vertex has been given a "label." For simplicity, let us assume these labels are the integers 1 through N. In how many different ways may a tree with N vertices be labeled? By "different" we mean that no rearrangement of two trees with the same number of vertices with different labeling will be identical. (Note that although we commonly associate data with each vertex, and identify one vertex as the root of the tree, that's not significant in this problem.)

Let's consider some examples. The figure below shows all possible arrangements of trees with N = 1, 2, 3, 4, or 5 vertices. The number shown below each tree is the number of different ways in which the vertices in each tree can be labeled.

[IMAGE]

Clearly a tree with only one vertex can be labeled in only one way - by assigning the label "1" to the single vertex. A tree with two vertices can also be labeled in only one way. For example, although the two trees shown on the left below appear to be different, the first can be easily transformed into the second. (Imagine the edges are strings, so the vertices can be easily repositioned without losing their connectivity.)

[IMAGE]

There are, however, three possible ways to label the vertices in a 3-vertex tree, as shown on the right above. No matter how you rearrange the labeled vertices in any of the three trees, you cannot produce any of the other labeled trees.

In a similar manner, the various arrangements of four vertices in a tree yield a total of 16 possible labelings - 12 for the four vertices "in a row," and 4 for the other configuration. There are three possible arrangements of the vertices in a tree with N = 5, with a total of 125 possible

Input

There will be multiple cases to consider. The input for each case is an integer N specifying the number of vertices in a tree, which will always be between 1 and 10. The last case will be followed by a zero.

Output

For each input case, display the case number (1, 2, ...), the input value of N, and the number of different ways in which a tree with N vertices may be labeled. Use the format shown in the examples below.

Example

Input:

```
2
3
4
5
0
```

Output:

```
Case 1, N = 2, # of different labelings = 1
Case 2, N = 3, # of different labelings = 3
Case 3, N = 4, # of different labelings = 16
Case 4, N = 5, # of different labelings = 125
```

Added by: Camilo Andrés Varela León
Date: 2007-10-07
Time limit: 1s
Source limit: 50000B
Languages: All
Resource: North Central North America Regional Programming Contest - 2003

SPOJ Problem Set (classical)

1871. Making A Budget

Problem code: MKBUDGET

A company uses temporary employees ("temps") to handle its varying workloads. By doing so, it avoids having to pay for benefits normally provided to its permanent employees. However, the company must pay an employment agency a fixed fee for each temp they hire, as well as paying the temp a fixed amount of severance pay when they are terminated - in addition, of course, to the monthly salary each temp receives. The company has a good understanding of when it needs temporary workers, and how many such workers it will require each month. Depending on the fee paid to the employment agency, the temporary worker's salary, and the severance pay, it may make sense to retain an unneeded worker for one or more months if it's known that they will be needed again in the future.

Let's consider an example. Suppose we know that in March the company will need 10 temps, in April they'll need 9, and in May they'll need 11. Suppose a temp earns $500 per month, that the employment agency receives $400 for each temp hired, and $600 is paid as severance to each temp that is terminated. If the company employs just the minimum number of temps required, then their payments will be as follows (we ignore the cost of terminating all employees at the end of the last month):

[IMAGE]

The total cost to the company is $20,400. But suppose they did not terminate the unneeded temp at the end of March, but just let that person remain employed. They would then save $400 in employment agency fees (since they'd need to hire just one additional temp for May), $600 in severance pay, and only have to pay the temp worker $500, for an overall savings of $500.

In this problem you are given, as input, the number of months for which the company is to plan its temp worker budget, the cost of hiring and firing a temp worker, the temp worker's monthly salary, and the required minimum number of workers needed each month. You are to determine the minimum cost to the company to have at least the required minimum number of workers on hand each month. Assume there are no temporary workers on hand before the first month, and that the cost of terminating the workers at the end of the last month is not to be included in the cost. You may assume that the planning interval will be no longer than 24 months, and the hiring cost, severance pay, and monthly salary for each temp worker is greater than zero.

Input

There will be multiple cases to consider. The input for each case begins with an integer N, the number of months for which planning is required (never larger than 24). This is followed by three integers giving the cost of hiring a worker, the worker's monthly salary, and the severance pay for a terminated worker. Finally there will appear N integers giving the required minimum number of workers needed in each month. The last case will be followed by a zero.

Output

For each input case, display the case number (1, 2, ...) and the minimum cost to the company. Use the format shown in the examples below.

Example

Input:

```
3 400 500 600 10 9 11
8 400 600 600 11 9 10 14 9 9 13 15
0
```

Output:

```
Case 1, cost = $19900
Case 2, cost = $66600
```

Added by: Camilo Andrés Varela León
Date: 2007-10-07
Time limit: 1s
Source limit: 50000B
Languages: All
Resource: North Central North America Regional Programming Contest - 2003

SPOJ Problem Set (classical)

1873. Accumulate Cargo

Problem code: ACARGO

A cargo shipment containing N ($1 <= \mathbf{N} <= 10^5$) boxes, has just arrived and it requires some regrouping. All the cargo is currently placed on a long circular conveyor belt of length **L** metres ($1 <= \mathbf{L} <= 10^9$), which you can control and perform the following operations.

- Rotate the wheel clock wise or anti-clockwise (free of cost).
- Hold the cargo at some point and not let it move, while the belt is rolling. This causes the cargo behind it to come closer to this cargo by one step. Any consecutive sequence of cargo is grouped together and called as a luggage. The aim of the program is to group all cargo as a single luggage. Now the cost of this holding operation for one second is equal to the weight of the *luggage* that is held fixed. Also please note that you can hold the luggage only at ends of the luggage and never at inbetween points.

Each unit of cargo weighs exactly one Kg. The conveyor belt rotates at a speed of one meters per second.
This cost function directly reflects the human effort required to group the cargo. Workers would be happy if you can write a program that prints the minimal required effort to group the cargo, assuming an intelligent sequence of operations.

Input Format:
The input file consists of multiple testcases.
The first line of each testcase contains two integers, **N** and **L**.
The following **N** lines contain one integer each specifying the position of the $\mathbf{i}^{\mathbf{th}}$ cargo on the belt. The positions will be between 0 & **L-1**.
Input terminates with a line containing N=0 and L=0 which must not be processed.
Output Format:
For each testcase print one integer in a single line, which is the minimal required cost for grouping all the cargo into a single luggage.
Sample Input:

```
3 5
0
1
3
2 3
0
1
5 20
2
7
12
9
13
0 0
```

Sample Output:

```
1
0
10
```

NOTE: Please use 64-bit integers.

Added by: Prasanna
Date: 2007-10-08
Time limit: 1s-10s
Source limit:50000B
Languages: All
Resource: NITT ACM ICPC Local Contest 2007 [Idea from a Topcoder problem]

SPOJ Problem Set (main)

1874. Burrows Wheeler Precompression

Problem code: BWHEELER

The **Burrows-Wheeler transform** (**BWT**, also called **block-sorting compression**), is an algorithm used in data compression techniques such as bzip2. It was invented by Michael Burrows and David Wheeler.

When a character string is transformed by the BWT, none of its characters change value. The transformation permutes the order of the characters. If the original string had several substrings that occurred often, then the transformed string will have several places where a single character is repeated multiple times in a row. This is useful for compression, since it tends to be easy to compress a string that has runs of repeated characters by techniques such as move-to-front transform and run-length encoding.

For example, the string:

```
SIX.MIXED.PIXIES.SIFT.SIXTY.PIXIE.DUST.BOXES
```

could be transformed into this string, which is easier to compress because it has many repeated characters:

```
 TEXYDST.E.IXIXIXXSSMPPS.B..E.S.EUSFXDIIOIIIT
```

Now the Burrows-Wheeler algorithm works as follows:

- Given an input string **S**, eg: "abcba".
- Find all rotations of **S**.

  ```
  eg: "abcba", "bcbaa", "cbaab", "baabc", "aabcb"
  ```

- Now sort the strings hence produced.

  ```
  eg: "aabcb", "abcba", "baabc", "bcbaa", "cbaab"
  ```

- Arrange the strings in a len(S) x len(S) grid.

  ```
  aabcb
  abcba
  baabc
  bcbaa
  cbaab
  ```

- Output the row number (1-based indexing) containing the original input string. Also output the strings formed by characters in the last column.
  eg: 2 bacab

Now given the output of Burrows-Wheeler, can you recover the orginal string?

Input Format:
The input file consists of multiple testcases.
The first line of each testcase contains one integer, **R**, indicating the row number containing the original input string in the sorted matrix.
The second line of each testcase contains one string, **Col**, which is the last column of the grid. (1 <= len(Col) <= 1000)
Col contains only lowercase characters. 1 <= **R** <= len(Col).
Input terminates with a line containing R=0 which must not be processed.
Output Format:
Print the original input string to the burrow wheeler's algorithm.
Testdata:
30 testcases
Sample Input:

```
2
bacab
3
rwlb
11
baaabaaaabbbaba
0
```

Sample Output:

```
abcba
rbwl
baaabbbbaaaaaab
```

Added by: Prasanna
Date: 2007-10-08
Time limit: 1s
Source limit:50000B
Languages: All
Resource: NITT ACM ICPC Local Contest 2007 [Wikipedia/General]

SPOJ Problem Set (classical)

1875. Cool Numbers

Problem code: COOLNUMS

Cool numbers are those, whose digits can be partitioned into two sets such that the sum of the digits in either sets are equal.
Example: 23450 is cool because 3+4+0 = 2+5; So is 91125;
The numbers 567, 34523 are not cool, since there is no such digit partition.
Write a program that prints the number of cool numbers in the inclusive range [A,B].

Input Format:
The input file consists of multiple testcases.
Each case contains one line containing two 32-bit unsigned integers A and B. (1 <= A <= B <= $4*10^9$).
Input terminates with a line containing two zeros and must not be processed.
Output Format:
For each testcase print a single line containing one integer saying the number of cool numbers between A and B, inclusive.
Sample Input:

```
1 11
12 20
1 20
3 100
6354 234363
123456789 234567891
0 0
```

Sample Output:

```
1
0
1
9
82340
54801678
```

Test Data:
About 50 testcases.

Added by: Prasanna
Date: 2007-10-08
Time limit: 30s
Source limit:50000B
Languages: All
Resource: NITT ACM ICPC Local Contest 2007 [Topcoder problem with Constraints raised]

SPOJ Problem Set (classical)

1876. Dragon Curves

Problem code: DRAGONCU

Define **r(s)** to be the complement of the reverse of the binary string **s**. i.e. Reverse **s** and then convert all 1's to 0's and all 0's to 1's.
Further define a sequence of binary string as follow: $s_0 = 1$ and $s_n = s_{n-1}\,1\,r(s_{n-1})$. i.e.

$s_0 = 1$

$s_1 = 110$

$s_2 = 1101100$

$s_3 = 1101100111001000$

...

We then program a robot to move at a steady speed of 1 unit per second and make a right-angle turn according to the characters of s_{10} after every unit of movement. At the kth turn, the robot turns to left if the kth character of s_{10} is a 1, and to right otherwise. The figure below shows the whole path of the robot.
[IMAGE]
The robot is placed at the origin (the small circle) and face east originally. It ends up at the coordinates (-32,32) (the small spot) after 2048 seconds. The path of the robot is known as a dragon curve, a pretty well-known pattern of fractal.
If the robot is now programmed with input string s_{30} (with identical initial conditions as above), it will keep moving and then stop after 2^{31} seconds. We want to know the location of the robot at any given time.

Input Format:
Input consists of multiple problem instances. Each instance consists of a single non-negative integer **n**, where **n** <= 10^9. The input data is terminated by a "-1". There will be less than 5000 test cases.
Output Format:
For each input integer **n**, print out the location of the robot right after **n** second since the robot starts its journey with input string s_{30}. The location should be printed with the format "(x,y)" in a single line.
Sample Input:

```
1
2
3
2048
1000000000
-1
```

Sample Output:

```
(1,0)
(1,1)
(0,1)
(-32,32)
(9648,-31504)
```

Added by: Prasanna
Date: 2007-10-08
Time limit: 1s
Source limit: 50000B
Languages: All
Resource: NITT ACM ICPC Local Contest 2007 [Tsinghua]

SPOJ Problem Set (classical)

1877. Enrich my purse

Problem code: EPURSE

Jack plays this ball game for the first time in his club. Jack has a ball, which bounces with a width of **W**. Coins are arranged on a straight line at regular intervals. If the ball strikes the i-th coin, Jack gains **money**[i] (which could possibly be negative). Jack can take atmost **B** turns, to throw the ball. At each turn, jack can either throw the ball from left to right, or right to left, and choose which ball to start the knock out. If he chooses to knock out from ball i to the right, he will knock out i, i+W, i+2W, ...; Similarly if he chooses to knock out from right to left, starting from ball i he will knock out, i, i-W, i-2W, ...; Please note that once a ball is knocked out, it is removed and it's place contains a void. i.e., you cant gain **money[i]** for the same **i** twice.
Jack wants to maximise his money gained, by carefully choosing his turns. If there is more than one way to gain the same money, jack wishes to minimise the number of times he throws.

Input Format

The input file consists of multiple testcases.
The first line of each testcase contains three integers, **W B N** (1 <= N <= 100; W,B > 0)
The second line of each testcase contains N integers, denoting **money[i]**. (| money[i] | <= 10^6)
Input terminates with a line containing three zeros which must not be processed.

Output Format

For each testcase print one line denoting the maximal money gained and the number of turns taken. Please see the sample output and stick to the output format.
"Case#id: Jack wins $X out of Y throws."
NOTE: You must spell the same way the sample output says. Extra spaces and case insensitivity can cause wrong answer responses.
Testdata:
100 testcases, Timelimit: 10s
Sample Input:

```
2 3 10
-1 3 2 5 1 -2 0 5 1 -3
2 3 14
-1 3 2 5 -5 -5 1 -2 0 5 -5 -5 1 -3
3 3 5
-1 -2 -3 -4 -5
1 2 6
-1 -1 10 10 -1 -1
0 0 0
```

Sample Output:

```
Case#1: Jack wins $15 out of 2 throws.
Case#2: Jack wins $10 out of 3 throws.
Case#3: Jack wins $0 out of 0 throws.
Case#4: Jack wins $18 out of 1 throws.
```

Output Explanation:
We present one of the optimal solutions. We number balls from 1 to N.
TestCase#1: [Jack takes only two throws, though he can take three]
Throw#1: From ball#3 towards right, 2 + 1 + 0 + 1 = 4
Throw#2: From ball#8 towards left, 5 + -2 + 5 + 3 = 11
TestCase#2:
Throw #1: From ball#3 towards left, 2 + -1 = 1
Throw #2: From ball#4 towards left, 5 + 3 = 8
Throw #3: From ball#13 towards right, 1 = 1
TestCase#3:
All numbers are negative. Jack takes no throws.

Added by: Prasanna
Date: 2007-10-08
Time limit: 10s
Source limit:50000B
Languages: All
Resource: NITT ACM ICPC Local Contest 2007 [Self]

SPOJ Problem Set (classical)

1878. Farmers Cattle

Problem code: FCATTLE

Farmer john owns a single cow and he loves it a lot. The cow has a disease and is going to die. To survive, the cow needs medicine of a particular type each day. Let us say the cow needs **medicine**[i] to survive the i^{th} day. (medicine[i] will be terminated by -1, which is an unavailable medicine, and the cow has to invariably die that day).
To help the cow, john has decided to buy pastures of some medical value. Farmer sees a two-dimensional grid of pastures, each cell having exactly one medical herb. Now he needs to buy a sub-rectangular region of the grid, whose area cannot exceed **A** (**A** > 1). With this region the farmer intends to feed his cow, as long as possible.

Input Format:
The input file consists of multiple testcases.
The first line of each testcase contains three integers, **R**, **C** and **A**.
The second line consists of sequence of integers describing **medicine**[i]. This list will be terminated by -1.
The next **R** lines contain **C** integers each, specifying the medicinal type of the herb in that cell. (1 <= **R,C** <= 200). All herbs are specified by non negative integers.
Input terminates with a line containing three zeros and must not be processed.
Output Format:
For each testcase print a single line containing 5 integers:
days r1 c1 r2 c2
(1 <= r1 <= r2 <= R, 1 <= c1 <= c2 <= C)

- ***days*** is the number of days the cow survives. We wish to maximise this.
- If there are more than one solutions print the one with minimal r1.
- If there are more than one solutions still, print the one with minimal c1.
- If there are more than one solutions still, print the one with minimal r2.
- If there are more than one solutions still, print the one with minimal c2.

Sample Input:

```
3 4 6
12 30 12 100 22 -1
30 12 5 3
12 30 100 5
22 3 22 100
3 4 6
2 30 12 100 22 -1
30 12 5 3
12 30 100 5
22 3 22 100
3 4 6
12 30 12 100 22 -1
```

```
30 12 5 3
12 30 100 5
22 12 22 100
0 0 0
```

Sample Output:

```
4 1 1 2 3
0 1 1 1 1
5 1 2 3 3
```

Added by: Prasanna
Date: 2007-10-08
Time limit: 5s-120s
Source limit:50000B
Languages: All
Resource: NITT ACM ICPC Local Contest 2007 [Self]

SPOJ Problem Set (classical)

1879. Game Time

Problem code: GAMETIME

Jack has got a new game of Super Mario recently. There are **n** castles in the game, and Princess Peach has been kidnapped by the King Koopa and enjailed in castle n - 1 (0-based). The King Koopa has Son Koopas, and the Son Koopas has Grandson Koopas, and the Grandson Koopas has Great-Grandson Koopas... The whole Koopa family is taking possession of these n castles, and there is exactly one Koopa in each castle and the King Koopa is in castle n - 1. Super Mario's task is to conquer these castles and save Princess Peach.

Each time, Super Mario can freely choose an unconquered castle and go in to fight with the Koopa. Super Mario has time **T[i]** to conquer castle **i** and beat the Koopa in it. If Super Mario fails to conquer the castle or fails to do that within the time limit, the game is over. Otherwise, the defeated Koopa will go to seek the Father Koopa for help. If any Koopa has two or more Son Koopas beaten by Super Mario, he will get angry. Super Mario has to go to the castle of the angry Koopa and beat him immediately; otherwise the Princess Peach will be killed. If castle n - 1 is conquered and King Koopa is defeated, the game will also end, but with triumph and Princess Peach saved.

Jack loves this game a lot, but he has to prepare for the upcoming ACM/ICPC. So he guarantees to himself that he only plays the game once a month. However, Jack really loves playing game and hates programming, he wants to maximize the time he can spend on the game without breaking his promise.

Input Format:
The input consists of multiple test cases.
Each test case starts with a number **N** (0 < N <= 100,000) in a single line, the number of castles and Koopas. There are two lines following and each contains N numbers.
The first line is N numbers, **T[i]** (0 < **T[i]** <= 100) for Super Mario to conquer castle i.
The second line is N numbers of **P[i]** (-1 <= Pi < N), the castle (0-based) in which the Parent of each Koopa resides.
-1 means that there is no parent for him, and the last number will always be -1, since King Koopa has no Parent Koopa. The input ends with 0(zero), which should not be processed.
Output Format:
Output the maximal time Jack can play in a single line for each test case.
Sample Input:

```
5
1 2 3 4 5
4 4 4 4 -1
5
2 2 2 2 2
1 2 3 4 -1
9
1 1 1 1 1 1 1 1 1
6 6 6 7 7 7 8 8 -1
0
```

Sample Output:

```
12
10
8
```

Added by: Prasanna
Date: 2007-10-08
Time limit: 10s
Source limit:50000B
Languages: All
Resource: NITT Local Contest 07 [ZJU inc constrts]

SPOJ Problem Set (classical)

1880. Hanoi Calls

Problem code: HANOICAL

Theory:

- Towers of hanoi is an arrangement consisting of three pegs and **N** discs of radius 1 to **N**.
- Each peg can hold zero or more discs, but at any point of time, the radius of the discs must be in decreasing order from bottom to top.
- A move consists of moving the topmost disc from one peg to another. After the move, the decending order property of pegs must hold.

Traditional problem is: If all discs are stacked up on peg#1, how many moves will it take to move all the discs to peg#2?
Recursive solution: Noting that for disc N to move, from peg #a to peg #b, all discs of size 1 to N-1 must be in peg #c. Hence there is exactly one minimal way to move the discs. After disc N has moved, all pegs from #c must be moved back to #a. If moves(N) denote the number of moves required to transfer N discs between two pegs (both sorted configuration), then moves(N) = moves(N-1) + 1 + moves(N-1); Solving the recurrence yields moves(N) = 2^N -1; The idea i am trying to share is that, there is exactly one such move sequence.
Now the problem is that given any initial configuration of the discs, and any final configuration, Can you tell me the minimal number of moves required to change it from initial to final configuration?

Input Format:
The input file consists of multiple testcases.
The first line of each testcase contains one integer, **N** (1 <= N <= 30)
The second line of each testcase contains N integers, each one of which will be between 1 and 3. The **i**-th integer tells you the peg number at which disc of radius **i** is present in the initial configuration.
The third line contains a similar specification for the final configuration.
Input terminates with a line containing a single zero, which must not be processed.
Output Format:
For each testcase print one line containing a single integer, which is the minimal number of moves to make the transfer.
Testdata:
100 testcases
Sample Input:

```
4
1 1 1 1
2 2 2 2
3
1 3 3
2 1 1
5
1 3 2 2 2
2 3 2 1 2
0
```

Sample Output:

```
15
6
14
```

Output Explanation:
TestCase#1:
This is the moves(4) = 2^4 -1;
TestCase#2:

```
[peg1, peg2, peg3] =
#0 [ {1}, {}, {3,2} ]
#1 [ {1}, {2}, {3}  ]
#2 [ {}, {2,1}, {3} ]
#3 [ {3}, {2,1}, {} ]
#4 [ {3}, {2}, {1}  ]
#5 [ {3,2}, {}, {1} ]
#6 [ {3,2}, {1}, {} ]
```

Added by: Prasanna
Date: 2007-10-08
Time limit: 1s
Source limit:50000B
Languages: All
Resource: NITT ACM ICPC Local Contest 2007 [Self/Traditional]

SPOJ Problem Set (classical)

1881. Instruction Decoder

Problem code: ICODER

Mathews uses a brand new 16-bit instruction processor. (Yeah i am being sarcastic!). It has one register (say R) and it supports two instructions:

- ADD X; Impact: R = (R + X) mod 65536
- MUL X; Impact: R = (R * X) mod 65536
- [For both instructions 0 <= X <= 65535]

Mathews sees a segment of code, but doesnot know what value the register had before the code was being executed. How many possible values can the register have after the segment completed execution?

Input Format:
The input file consists of multiple testcases.
The first line of each testcase contains one integer, **N**. (1 <= N <= 100,000).
The following **N** lines contain one instructions each.
Input terminates with a line containing N=0, which must not be processed.
Output Format:
For each testcase print one integer in a single line, denoting the number of different values the register can take after code execution.
Sample Input:

```
1
ADD 3
1
MUL 0
5
MUL 3
ADD 4
MUL 5
ADD 3
MUL 2
8
ADD 32
MUL 5312
ADD 7
MUL 7
ADD 32
MUL 5312
ADD 7
MUL 7
0
```

Sample Output:

```
65536
1
32768
16
```

Added by: Prasanna
Date: 2007-10-08
Time limit: 3s
Source limit: 50000B
Languages: All
Resource: NITT ACM ICPC Local Contest 2007 [Self]

SPOJ Problem Set (classical)

1960. Rectangles

Problem code: RECTANGL

You are given a set *S* of *N* points in the plane and must count the number of distinct axis-parallel rectangles whose four vertices all lie in *S* (that is, count those rectangles which have two sides parallel to the **x**-axis, and the other two sides parallel to the **y**-axis).

Input

The first line of the input is *N* (1 <= *N* <= 250000), the number of points in *S*. *N* lines then follow, where the **i**-th line is of the form "x_i y_i", giving the coordinates of a point (x_i, y_i) in *S*. All given points are distinct, and all coordinates fit into a 32-bit signed integer.

Output

Your output should consist of a single number, the number of distinct axis-parallel rectangles whose four vertices all lie in *S*, followed by a newline.

Example

Input:
```
6
-1 0
-1 1
0 0
0 1
1 0
1 1
```

Output:
```
3
```

Added by: Jelani Nelson (Minilek)
Date: 2007-10-25
Time limit: 1s-30s
Source limit:50000B
Languages: All
Resource: MIT Individual Contest 2007

SPOJ Problem Set (classical)

1961. Roman Roads

Problem code: ROMANRDS

Some 2000 years ago the Roman Empire covered a large part of Europe including the entire coast of the Mediterranean. The transportation network of that empire consisted of roads and sea routes (the two are considered equivalent and simply called roads for this problem). Each road connected exactly two cities and the road network was such that every city can be reached from Rome. However, building this network required resources (cobblestone and buoys) proportional to the total length of the network. In order to cut down on building costs and maintainance and spend the rest of the money on wine, the empire built the cheapest possible network.

Additionally, each road had a single signpost that listed all of the other roads it connected to (at any of its two cities). There were *N* roads in the empire labeled *1*, *2*, ..., *N*. It is believed that a traveller once travelled all roads and for each road wrote down the numbers on its signpost, thus making a map.

2000 years later a young archaeologist found something that looks suspiciously like that map. Your job is to write a program that determines if this can really be a map of the Roman empire and for each road output the two cities it connected. Note that roads in a valid map are always between two distinct cities.

(Disclaimer: The description of the transportation network is for this problem only and may not necessarily be what the Roman Empire actually did. Do not cite this in your history papers: I made it up.)

Input

The first line of input contains *N*, *1* <= *N* <= *500*. Each of the next *N* lines contains a space-separated list of integers. The *i*-th of these lines describes the "roads" that connect to "road" *i*. The first number on the line specifies the number of those "roads", d_i, and the following d_i numbers specify their labels.

Note that although at this point we don't know if the map is valid, the input is consistent, i.e. if a road *x* is on the signpost of *y*, then *y* will be on the signpost of *x*. (Otherwise the archaeologist would know this is not a Roman map right away).

Output

If the input ***cannot*** describe a valid map according to the description, output "NO" on the first (and only) line of output.

Otherwise, output "YES" and on each of the next *N* lines, write two integers separated by space, the numbers of the two cities that the road connected. City labeling is up to you with the only restriction that all city labels must be integers between *1* and *M*, where *M* is the total number of cities. Of course, a city can only have one label.

Note that since we don't know the actual locations of the cities or whether the roads were straight, we have no idea what the total cost of the network might have been. The archaeologist is willing to accept any map for which she can't determine if there is a cheaper network without knowing the actual costs. It is assumed that each road had some positive cost.

Example

Input:

```
3
2 3 2
2 1 3
2 2 1
```

Output:

```
YES
1 3
4 1
1 2
```

Added by: Jelani Nelson (Minilek)
Date: 2007-10-25
Time limit: 30s
Source limit: 50000B
Languages: All
Resource: MIT Individual Contest 2007

SPOJ Problem Set (classical)

1962. Circles

Problem code: CIRCLES

Little Gary plays the following video game. Circles pop up on the screen and disappear from it. When the screen flashes, Gary can draw a straight line on the screen and win as many points as there are circles intersected by the line. As a born-to-be-winner, Gary wants to maximize his score. Please, help him, and write a program that will determine the maximum number of points he can win each time the screen flashes.

Input

The first line of the input contains M ($1 <= M <= 1000$), the number of events during the game. The next M lines contain descriptions of the events, one per line. They can be in one of the following three formats:
1 x y r
, representing a circle of radius r popping up with the position of its center at (x, y) in the plane
2 i
, representing a circle i disappearing, where circle i is the ith circle that popped up since the beginning of the game; and
3
, representing the screen flashing.
x, y, and r are real numbers with at most two decimals, $-10^6 < x, y, r < 10^6$, $r > 0$.
Notes:

- A line intersects a circle if it has at least two common points with it.
- At any time, no two Circles on the screen have a common point.
- At any time, there is no line that "touches" more than two circles (a line touches a circle if they have exactly one common point).
- At any time, there are no more than 100 circles on the screen.
- Each i determines a circle that is on the screen at the moment of removal.
- No circle is removed twice.

Output

Each time the screen flashes, write an integer to a separate line, which is the maximum number of circles Gary can intersect.

Example

Input:

```
9
1 3.00 0.00 1.00
1 -2.00 0.00 1.00
3
1 2.00 3.00 1.50
3
```

```
1 2.00 -4.00 1.00
3
2 3
3
```

Output:

```
2
2
3
2
```

Added by: Jelani Nelson (Minilek)
Date: 2007-10-25
Time limit: 30s
Source limit: 50000B
Languages: All

SPOJ Problem Set (main)

1963. Image Projections

Problem code: IMGPROJ

Given an image *I* with *N* columns and *M* rows, a diagonal projection is the vector (d_1, d_2, ..., d_{M+N-1}) where $d_i = S_{x+y-1=i}\ I(x,y)$. Here *I(x,y)*, *1 <= x <= N*, *1 <= y <= M*, is the image intensity (a non-negative integer less than 256) at column *x* and row *y*.

You are given a set of images and you are asked to find the diagonal projection for each of them.

Input

The first line of input contains a positive number, the number of images that follow. For each image there is a line with *N* and *M*. The following *M* lines describe one row each starting from row 1. A row is described in run-length encoding by pairs of numbers separated by spaces. The first number in each pair is the length of the run and the second number is the image intensity. Obviously, for each row, the run lengths add up to *N*. As in the example input, there is a blank line between each two consecutive images and before the first one.

The number of test cases is at most 10. The width of each image is less than 10^9 and the height is less than 10^3. Additionally, the total size of the input does not exceed 4 MB.

Output

For each image you should output one line, the diagonal projection for the image in run length encoding. The number of output lines should be the same as the number of images in the input. All the numbers on a line should be separated by exactly one space.

When encoding the output in run-length encoding, the runs should be as long as possible, i.e. no two consecutive runs should have the same intensity value.

Example

Input:

```
2

3 3
1 1 1 2 1 3
1 1 1 2 1 3
1 3 1 2 1 1

3 2
3 1
3 1
```

Output:

```
1 1 1 3 1 8 1 5 1 1
1 1 2 2 1 1
```

Added by:	Jelani Nelson (Minilek)
Date:	2007-10-25
Time limit:	30s
Source limit:	50000B
Languages:	All
Resource:	MIT Individual Contest 2007

SPOJ Problem Set (main)

1964. Tree cut

Problem code: MMCUT

You are given a tree (a connected, acyclic graph) along with a set of **commodities**, i.e. pairs of vertices, (s_1,t_1),...,(s_m,t_m) (s_i != t_i). A **multicut** is a set of edges that when removed disconnects s_i from t_i for all *i*. There is a unique path $P_{u,v}$ between every pair of vertices *u,v* in a tree, and the **max-cost** of a multicut *S* is $\max_i |S * P_{s_i, t_i}|$. You will be given a rooted tree of height *1* and a set of commodities and must return the minimum possible max-cost over all multicuts.

Input

The first line of the input is "*N M*" (*1 <= N, M <= 100000*), where *N* is the number of vertices in the tree and *M* is the number of commodities. All vertices are numbered *0, ...,N-1*, and the root has label *N - 1*. *M* lines then follow, where the *i*th line is "s_i t_i", representing a commodity (s_i, t_i) where s_i != t_i. Commodities are distinct: neither (s_i, t_i) = (s_j, t_j) nor (s_i, t_i) = (t_j, s_j) will hold when *i* != *j*.

Output

Your output should consist of a single number, the minimum possible max-cost of a multicut, followed by a newline.

Example

Input:
```
10 2
0 5
4 8
```

Output:
```
1
```

Added by: Jelani Nelson (Minilek)
Date: 2007-10-25
Time limit: 30s
Source limit:50000B
Languages: All
Resource: MIT Individual Contest 2007

SPOJ Problem Set (main)

1965. Set Cover

Problem code: SETCOV

In the set cover problem there is a collection $C = \{S_1, ...,S_m\}$ of subsets of the universe $[n] = \{0, ...,n-1\}$, and one must find a minimum-sized subcollection of C that still covers $[n]$ (it may be the case that S_i and S_j contain the exact same elements for some $i != j$). A **path of length r** is a graph on $r+1$ vertices $v_0, ...,v_r$ where v_i has an undirected edge to v_{i+1} for $i = 0, ...,r-1$ (these are the only edges). A set cover instance I is said to be **path-realizable** if there exists a mapping from I to a path of length m where the S_i are mapped to edges in the path and each i in $[n]$ is mapped to a pair of (not-necessarily distinct) vertices s_i, t_i on the path such that the edges lying between s_i and t_i correspond exactly to the sets of C that contain i. Two sets S_i,S_j must be mapped to different edges on the path if $i != j$. You will be given a set cover instance that is guaranteed to be path-realizable and should output the size of a minimum-sized subcollection of C still covering $[n]$.

Input

The first line of the input is "N M" (1 <= N, M <= 300), where N is the size of the universe and M is the number of sets S_i in the collection of subsets of $\{0, ...,N - 1\}$. What follows are M groups of lines. The ith group starts with one line containing $|S_i|$, the size of the ith subset. If $|S_i| = 0$, the current group of lines ends. Otherwise the next line is a space-separated list of the elements contained in S_i.

Output

If $[n]$ cannot be covered by a subcollection of C then you should output -1, followed by a newline. Otherwise, your output should consist of two lines. The first line is the size of a minimum-sized set cover. The second line is a space-separated list of the 0-based indices of the sets in an optimal set cover.

Example

Input:

```
3 4
0
2
2 1
2
1 0
0
```

Output:

```
2
1 2
```

Added by:	Jelani Nelson (Minilek)
Date:	2007-10-25
Time limit:	30s
Source limit:	50000B
Languages:	All
Resource:	MIT Individual Contest 2007

SPOJ Problem Set (main)

1966. Ski Valley

Problem code: SKIVALL

The Society of Sport of New Hampshire has decided to build a new attraction in White Mountains. For the first time, the world will see a ski-valley, a ski path that goes downhill then uphill. They believe that skiers can gain enough speed from going down in the first part in order to climb up the second part. To maximize the joy of visitors, they want to find the longest such path.

To simplify calculations, they approximate the mountain terrain with a matrix of square fields and obtain the height of each field from the New Hampshire Geographical Institute. A ski-valley is a sequence of neighboring fields, such that height of fields only decreases along the sequence until some point, and then it only increases until its end. No field appears more than once in a ski-valley. Two fields are neighbors if they share a common edge. The length of a ski-valley is the number of fields in its sequence.

More technically, the terrain is an M x N matrix of fields, where (i, j) denotes a field in the ith row and jth column, and $h(i, j)$ denotes its height. A ski-valley is a sequence $(x_1, y_1), (x_2, y_2), ..., (x_l, y_l)$, such that:

1. for any i ($1 <= i <= l-1$), either $x_i = x_{i+1}$ and $|y_i - y_{i+1}| = 1$, or $y_i = y_{i+1}$ and $|x_i - x_{i+1}| = 1$ (neighbors rule)
2. if $i != j$ ($1 <= i, j <= l$), then either $x_i != x_j$ or $y_i != y_j$ (no repeating rule), and
3. There exists a k ($1 <= k <= l$), such that $h(x_1, y_1) > h(x_2, y_2) > ... > h(x_{-k-1}, y_{-k-1}) > h(x_k, y_k) < h(x_{-k+1}, y_{-k+1}) < ... < h(x_l, y_l)$ (down-up rule).

The length of such ski-valley is l.

They hire you, a reputable programmer, to write a program that will find a ski-valley of maximum length. If there are multiple ski-valleys with the same (maximum) length, you can choose any of them.

Note: Yes, they were not cautious and also allowed a ski-valley to bo only downhill or only uphill, but your job is only to adhere to the specification they gave you!

Input

The first line of the input contains M and N (1 <= M, N <= 60), respectively, separated by a space character. Each of the next M lines contain N numbers, such that the jth number in the ith line represents $h(i, j)$ (-10^6 <= h(i, j) <= 10^6). No two fields in the terrain are of the same height. Numbers on a line are separated by a space character.

Output

In the first line of the output, write a single number l_{max}, which is the maximum length of a ski-valley. In the next l_{max} lines write a description of any ski-valley of that length. In each of the lines, write two integers separated by a space character, such that numbers x_i and y_i in the *i*th line represent (x_i, y_i), the *i*th field in the ski-valley.

Example

Input:

```
3 4
2 6 7 16
1 4 3 20
9 8 17 12
```

Output:

```
9
3 1
3 2
2 2
2 1
1 1
1 2
1 3
1 4
2 4
```

Added by: Jelani Nelson (Minilek)
Date: 2007-10-25
Time limit: 30s
Source limit: 50000B
Languages: All
Resource: MIT Individual Contest 2007

SPOJ Problem Set (classical)

1991. Another Continuous Fractions Problem

Problem code: ACFRAC

The problem description is the same as the problem CFRAC and CFRAC2.

Input & Output

Multiple test cases, a single line with a single uppercase character C indicates the end of the input. The number of test cases will be less than 1000.

For each test case:

The first line of the input contains a single uppercase character A or B.A denotes that the input following character A and output format of this test case is the same as problem CFRAC, otherwise the input following character B and output format of this test case is the same as problem CFRAC2. But please pay attention that: the width and the height of the image after the character B will not appear in the input; the original fraction will not appear in the output of the test case of type A.

The example will make everything clear.

Example

Input:

```
A
75 34
B
.........1......
2.+.-------------
............1....
....4.+.---------
..............1..
........1.+.-----
...............1
............5.+.-
...............1
C
```

Output:

```
Case 1:
.........1......
2.+.-------------
............1....
....4.+.---------
..............1..
........1.+.-----
...............1
............5.+.-
...............1
Case 2:
75 34
```

Added by:	Blue Mary
Date:	2007-11-01
Time limit:	30s
Source limit:	50000B
Languages:	All except: C99 strict
Resource:	XX Colombian National Programming ACM 2006, test data by Blue Mary

SPOJ Problem Set (classical)

2000. Boxes (Hard)

Problem code: BOX

There are n boxes on the circle. The boxes are numbered from 1 to n in clock wise order. There are balls in the boxes, and the number of all the balls in the boxes is not greater than n.

The balls should be displaced in such a way that in each box there remains no more than one ball. In one move we can shift a ball from one box to one of it's neighboring boxes.

Write a program that: reads from the standard input the number of boxes n and the arrangement of balls in the boxes, computes the minimal number of moves necessary to displace the balls in such a way that in each box there remains no more than one ball, writes the result in the standard output.

Input

The first line of the input file contains an integer t representing the number of test cases. Then t test cases follows. Each test case has the following form:

- The first line contains one positive integer n - the number of boxes
- The second line contains n nonnegative integer separated by single spaces. The i-th number is the number of balls in the i-th box.

Output

For each test case, output one nonnegative integer - the number of moves necessary to displace the balls in such a way that in each box there remains no more than one ball.

Example

Input:
```
1
12
0 0 2 4 3 1 0 0 0 0 0 1
```

Output:
```
19
```

Note

There are two input files.

In the first input file, t=19, n<=1000, time limit=0.5 second;

In the second input file, t=3, n<=200000, time limit=10 seconds.

Warning: large input/output data, be careful with certain languages

---

Added by: Blue Mary
Date: 2007-11-02
Time limit: 0.5s-10s
Source limit: 50000B
Languages: All except: C99 strict
Resource: POI III, stage 3; Special thanks to Lei Huang

SPOJ Problem Set (classical)

2002. Random Number Generator

Problem code: RNG

LoadingTime got a RNG (*Random Number Generator*) from his classmate several weeks ago. And he spent a lot of time study it. He found that RNG can generate a real number in range [-**S**,**S**] by executing following steps. First RNG generates n integer $\mathbf{X}_1..\mathbf{X}_n$, the sum of which is equal to **S**. Then for each $\mathbf{X}_i$, it generates a real number in range [$-\mathbf{X}_i$,$\mathbf{X}_i$] randomly. The output (a real number) of RNG will be the sum of the **N** generated real numbers. LoadingTime noticed that the distribution of the output was very interesting, and he wanted to know: for given **N** and **X**, what's the probability that the generated number is in range [**A**,**B**]. Could you help him?

Input

The first line contains an integer T representing the number of test cases.

For each test case, the first line contains three integers **N**, **A**, **B**(1 <= **N** <= 10, -100 <= **A** <= **B** <= 100) In the second line of the test case, you are given $\mathbf{X}_1...\mathbf{X}_n$ (1 <= $\mathbf{X}_i$ <= 10).

Output

For each test case, print a line contains a real number representing the probablity as the problem required. It must be printed with exactly nine decimal places.

Example

Input:

```
5
1 -100 100
10
1 10 90
10
1 -20 5
10
2 -20 5
5 5
5 -5 10
1 2 3 4 5
```

Output:

```
1.000000000
0.000000000
0.750000000
0.875000000
0.864720052
```

Added by:	Jin Bin
Date:	2007-11-03
Time limit:	1s
Source limit:	10000B
Languages:	All except: C99 strict
Resource:	www.test-the-best.by

SPOJ Problem Set (classical)

2005. Minus Operation

Problem code: MINUS

There are *n* integer numbers listed in one line. Every time you can arbitrarily choose two neighboring integers, kick them out and write down the result of the first number subtract the second number instead. Now, you want to get number *m* after you perform this operation *n-1* times.

Input

Multiple test cases, the number of them is given in the very first line.

For each test case:

The first line contains two space-separated integers *n*(1<= *n* <=100) and *m*(-500<= *m* <=500). *n* lines follow, each contains a single integer (in the range [0,100]) denotes the original numbers.

Output

For each test case:

You should output *n-1* lines, each contains a single integer p_i, which denotes that you are to wipe the p_i-th and (p_i+1)-th number in the current sequence and use their substraction instead. Each line of your output should not have any leading or trailing white spaces.

You may assume that there is always a valid solution to each test case in the input file. If there are multiple solutions, any of them will be accepted.

Print a blank line after each test case.

Example

Input:

```
1
5 4
12
10
4
3
5
```

Output:

```
2
3
2
1
```

Added by:	Blue Mary
Date:	2007-11-03
Time limit:	10s
Source limit:	50000B
Languages:	All except: C99 strict
Resource:	description by Blue Mary; standard program and test data by Zhou Yisu

SPOJ Problem Set (classical)

2006. Load Balancing

Problem code: BALIFE

SuperComputer Inc. have built a super-fast computer server consisting of N hyper-scalar lightning-fast processors Beta 007. These processors are numbered from 1 to N and are used to process independent jobs. Every new incoming job is assigned to an arbitrary processor. Sometimes, a processor may be assigned too many jobs while other processors have a relatively light load (or even wait idly). In that case, the whole system undergoes rebalancing.

Rebalancing proceeds in rounds. In each round, every processor can transfer at most one job to each of its neighbors on the bus. Neighbors of the processor i are the processors i-1 and i+1 (processors 1 and N have only one neighbor each, 2 and N-1 respectively). The goal of rebalancing is to achieve that all processors have the same number of jobs.

Given the number of jobs initially assigned to each processor, you are asked to determine the minimal number of rounds needed to achieve the state when every processor has the same number of jobs, or to determine that such rebalancing is not possible.

Input file specification

The input file consists of several blocks. Each block begins with a line containing a single number N(1<= N <=9000) - the number of processors. N numbers follow, separated by spaces and/or end of line characters. The i-th number denotes the number of jobs assigned to the i-th processor before rebalancing. There is a blank line after each block. The last block is followed by a single number -1 on a separate line (which should not be processed).

Output file specification

For each block in the input file, output the minimal number of rounds needed to rebalance loads for all the processors. If it is not possible to rebalance jobs so that each processor has the same number of jobs, output -1.

Example

Input file:

```
3
0 99 3

2
49 50

8
16 17 15 0 20 1 1 2

10
0 0 100 0 0 0 0 0 0 0

-1
```

Output file:

```
34
-1
23
70
```

Added by: Blue Mary
Date: 2007-11-03
Time limit: 1s
Source limit: 50000B
Languages: All except: C99 strict
Resource: IPSC 2002

SPOJ Problem Set (classical)

2007. Another Very Easy Problem! WOW!!!

Problem code: COUNT

Background

This problem is somewhat easier than the problem A Very Easy Problem! because of the super long time limit...

Description

Assurance Company of Moving (ACM) is a company of moving things for people. Recently, some schools want to move their computers to another place. So they ask ACM to help them. One school reserves *K* trucks for moving, and it has *N* computers to move. In order not to waste the trucks, the school ask ACM to use all the trucks. That is to say, there must be some computers in each truck, and there are no empty trucks. ACM wants to know how many partition shemes exists with moving N computers by K trucks, the ACM ask you to compute the number of different shemes with given N and K. You needn't care with the order. For example *N*=7,*K*=3, the the following 3 partition instances are regarded as the same one and should be counted as one sheme: "1 1 5","1 5 1","5 1 1". Each truck can carry almost unlimited computers!!

Input

Each line of the input contains two postisive integer N (1<=N<=5000) and K(1<=K<=N).Input is terminated by a line with N=K=0(this case should not be processed).

Output

For each line, output the number of different partition sheme. To avoid big integers, you may output the answer modudo 1988.

Example

Input:

```
1 1
7 3
0 0
```

Output:

```
1
4
```

Added by:	Blue Mary
Date:	2007-11-03
Time limit:	21s
Source limit:	50000B
Languages:	All except: C99 strict
Resource:	Time limit: 1000 Years! Memory limit: 2000 GB! Acc%: 100%!

SPOJ Problem Set (classical)

2008. Dab of Backpack

Problem code: BACKPACK

One day Blue Mary goes to a nearby supermarket to buy some goods. She has a backpack, whose capacity is *V-Max*. She finds that there are many goods in the market, each has a volume V_i(it will always be a multiple of 10 and less than 10000) and an importance C_i(1<= C_i <=5). Since she has almost unlimited money, the only problem she is to solve is how to choose goods such that the total volume won't exceed the capacity of the backpack and the sum of the product of the volume and the importance of each good is maximum. To be an excellent mathematician, she comes up with the answer quickly, and now she wants you to do a harder task. There are two kinds of goods: main goods and attachments. If you want to buy an attachment you must buy its main good before.

Input

Multiple test cases, the number of them is given in the very first line.

For each test case:

The first line contains two space-separated integers *V-Max* (1<=*V-Max*<=32000) and the number of the goods *N* (1<=*N*<=60). N lines follow, each contains three space-separated integers V_i, C_i and a integer *u*. If *u* is not 0, this good is an attachment of good *u*(as the order in the input file).

To make the problem not too difficult, Blue Mary tells you that:

(A) An attachment won't have any attachments which belong to it.

(B) A main good will always have less than 3 attachments.

Output

For each test case:

The first and the only line contains a single integer denoted the answer.

Example

Input:

```
1
1000 5
800 2 0
400 5 1
300 5 1
400 3 0
500 2 0
```

Output:

```
2200
```

Added by: Blue Mary
Date: 2007-11-03
Time limit: 3s
Source limit:50000B
Languages: All except: C99 strict
Resource: Based on a Problem from Chinese National Olympiad in Informatics in Province 2006

SPOJ Problem Set (classical)

2009. Cryptography

Problem code: CRYPTO

Your task is to work as a cryptographer for some time, the reason is ...

Blue Mary has set a problem using English. Since the problem is too easy and it will be boring when solving it, she has deleted all the whitespaces and punctuations in the original problem description, and lowercased all the capital latin letters. Then, she randomly chose a permutation of the English lowercase letter alphabet, and then used the corresponding letters in place of the letters in the original text.

The encrypted text can be downloaded here.

There is no example for this problem.

Blue Mary's note: some tricky test cases were added on Nov. 25th, 2007 and the time limit has been changed. Programs have been rejudged and some "accepted" solutions got Wrong Answer.

Added by:	Blue Mary
Date:	2007-11-03
Time limit:	1s-11s
Source limit:	50000B
Languages:	All except: C99 strict
Resource:	Sadly, the ability to make a simple problem difficult to understand is seldom considered a talent.

SPOJ Problem Set (classical)

2019. The Rolling Ball

Problem code: ROLLBALL

Rolling Ball

Problem

A solid spherical ball of radius **R** rolls without slipping on the inside surface of a fixed cone, whose tip points downward. The half-angle at the vertex of the cone is **u**. Initial conditions have been set up so that the ball travels around the cone in a horizontal circle of radius **l>R**, with the points on the ball that touch the cone tracing out a circle on the ball. Determine the radius of the circle of these contact points, if you want the sphere to travel around the cone as fast as possible.

The Input

Each line of input has integers **l** (R < l <= 1000000) **R** (0 < R <= 1000) and **u** (0 < u < 90) given in degrees.

The Output

For each line of input, output the radius of the circle of the contact points, round to integer.

Sample Input

```
220000 100 29
```

Sample Output

```
46
```

Problemsetter --- Wu, Xiaogang

Added by: Chen Xiaohong
Date: 2007-11-06
Time limit: 1s
Source limit:50000B
Languages: All

SPOJ Problem Set (classical)

2021. Moving Pebbles

Problem code: PEBBMOV

A Game of Moving Pebbles

Problem

Two players play the following game. At the beginning of the game they start with **n** (1<=**n**<=100000) piles of stones. At each step of the game, the player chooses a pile and remove at least one stone from this pile and move zero or more stones from this pile to any other pile that still has stones. A player loses if he has no more possible moves. Given the initial piles, determine who wins: the first player, or the second player, if both play perfectly.

The Input

Each line of input has integers **0 < n <= 100000**, followed by **n** positive integers denoting the initial piles.

The Output

For each line of input, output "first player" if first player can force a win, or "second player", if the second player can force a win.

Sample Input

```
3 2 1 3
```

Sample Output

```
first player
```

Problemsetter --- Chen, Jiahong

Added by: Chen Xiaohong
Date: 2007-11-06
Time limit: 5s
Source limit:50000B
Languages: All

SPOJ Problem Set (main)

2022. Truth Or Lie

Problem code: TRUTHORL

Truth Or Lie

Problem

Suppose you have **m** yes or no questions that you want to ask **n** people. You are allowed to ask each person exactly two different questions. He/she will answer exactly one of them correctly and one of them incorrectly, you don't know which is a correct answer and which is an incorrect one. Given their answers, determine the number of combinations of answers to the **m** questions that can still be correct (i.e., no contradictions).

The Input

First line is the number of inputs. For each set of input, start out with a line of **n<=10000** and **m<=200**, followed by **n** lines. The i-th line has four integers **a b c d**. It means that the answer given by the i-th person for question a is b, and for question c is d. Moreover, the answer "1" means yes, and "0" means no.

The Output

For each line of input, output "No Inference" if the answers do not help you eliminate any wrong combination of answers or the number of combinations of possible answers is 0, otherwise output the size of the set of combinations of answers still possible.

Sample Input

```
2
2 2
1 1 2 0
1 1 2 1
4 4
1 1 2 1
1 1 3 0
2 1 4 1
3 1 4 0
```

Sample Output

```
No Inference
2
```

Problemsetter --- sy

Added by: Chen Xiaohong
Date: 2007-11-06
Time limit: 1s
Source limit:50000B
Languages: All

SPOJ Problem Set (classical)

2023. One Instruction Computer Simulator

Problem code: ONEINSTR

A computer with only one instruction! The instruction is:

```
SUBLEQ A B C
```

This means: subtract the value in M(A) from M(B) and store it in M(B); if the result is non-positive jump to the instruction in position C. M(i) represents the value stored in memory position i. The computer has a memory of 9999 integer positions, numbered from 0 to 9998. C > 9996, indicates the end of the program. Also, if A is negative, then the value of A is directly subtracted from M(B).

Since there is only one instruction, it is unnecessary to represent its opcode explicitely in memory. Therefore, an instruction is stored in main memory using three consecutive memory positions, which correspond to the three instruction parameters. The memory is organized as follows:

```
Position Content
0-8      input/output variables (M0 to M8)
9-9998   program memory (instructions+data)
```

The following pseudo-code shows the one instruction computer simulator:

```
simulate(integer M[0..9998])
   integer pc,A,B,C
   pc = 9
   while (pc<9997)
      A = M[pc]; B = M[pc+1]; C = M[pc+2]
      if(A>=0)
         M[B] = M[B] - M[A]
      else
         M[B] = M[B] - A
      if (M[B]>0)
         pc = pc + 3
      else
         pc = C
      end_if
   end_while
end_simulate
```

Each iteration of the above while instruction is called a simulation cycle. You are to translate postfix instructions into this machine language. There are at most 100 arithematic terms and 99 operators. Numerical constants are non-negative and less than or equal to 10000.

Input

The input has several test cases, one test case per line. Each test case corresponds to an arithmetic expression in postfix notation. An expression may contain constants (integer values), input variables (M0 to M8) and arithmetic operators (+, -, *, /).

Output

For each test case, a program must be printed using the following format: First line indicates m, the number of instructions of the program; and the following m lines contain the program, one instruction per line, where each instruction is represented by 3 integer values separated by one blank space. Your outputed program must finish within 10^7 simlation cycles for each test case.

Example

Input:

```
100
M1 M2 -
```

Output:

```
4
0 0 12
-100 0 0
19 19 10000

4
0 0 12
1 2 15
2 0 18
21 21 10000
```

Added by: Chen Xiaohong
Date: 2007-11-06
Time limit: 1s
Source limit:50000B
Languages: All
Resource: Changed and Enhanced from Columbian National Contest

SPOJ Problem Set (classical)

2031. Please help You-Know-Who

Problem code: YKH

Background

Poor You-Know-Who was out of business for quite some time, being neither dead, nor alive. He would like to get rich so that he can run for the post of Minister of Magic. From there he could unfold at least some of his evil plots even if he lost most of his magical powers. You are fed up with all this Harry Potter business and decide to help You-Know-Who make some dough. You-Know-Who inherited lots of magic powder that increases one's sense of smell for a short period of time if inhaled. He will sell it and you'll help him maximize the profit.

Problem

You-Know-Who has `C` clients. Client `i` will buy `a[i]-b[i]*p` powder if the unit price is `p`. That is, if You-Know-Who shows a certificate issued by the Minister saying that the asked price is the correct price. Getting one such certificate involves paying a bribe `B`. To maximize the profit he may buy several such certificates and use for each client the one that maximizes his profit. You must tell You-Know-Who what certificates he should get. With some luck, Harry Potter is out!

Input

The first line contains the number of testcases, which shall be less than 20. The testcases follow. (Possible empty lines between testcases should be ignored.)

The first line of each testcase input contains two non-negative integers: the bribe `B` and the number of clients `C`. The next `C` lines describe each client by two positive integers `a[i]` and `b[i]` (on line `i`). You are guaranteed that all numbers in the input are at most 2000.

Output

For each testcase the output is a single number, the maximum profit of You-Know-Who, and it should be written on a line by itself. The answer should have either an absolute or a relative error less than 1e-6.

Example

Input:

```
2
10 2
10 1
20 3
100 1
5 1
```

Output:

```
46.25
0
```

First testcase: If YKW would have only the first client then he should choose to get a certificate for the price 5. This way he will sell 5 grams of powder for a profit of 5x5-10=15. Analogously, for the second client the optimal price would be 3.(3), which would make the client buy 10 grams of magic powder. Choosing the optimal price for each client means that two bribes must be payed, which leads to a profit of 5x5+10x3.(3)-2x10=38.(3). Nevertheless, YKW can win more money by getting only one certificate for the price 3.75. This way the profit is 6.25x3.75+8.75x3.75-10=46.25.

The second testcase shows that sometimes it's better to not sell magic powder to some clients.

Added by:	Radu Grigore
Date:	2007-11-07
Time limit:	10s
Source limit:	10000B
Languages:	All
Resource:	CSIC 2007

SPOJ Problem Set (classical)

2038. Rectangle Tiling

Problem code: TILING

We say that a 2-dimensional, rectangular word w of size n×m (imagine it as a board with letter written in the squares) can be tiled with a rectangular pattern p if there are such occurrences of p in w (but not necessarily all of them) that no two of them overlap and each symbol (square) of w is covered by one of them. Given such word w, find a rectangular pattern p of smallest size (area) wich the word w can be tiled with.

Input

The first line of input contains a number t (1<=t<=100) that indicates the number of test cases to follow. Each test case begins with a line consisting of two positive integers n and m (1<=n,m<=1000) indicating dimensions of the board. n lines follow, each of them containing m small letters of the English alphabet (a,b,...,z).

Output

For each test case output the smallest possible area of a pattern p that can be used to tile the given board.

Example

Input:

```
3
4 3
aaa
aaa
aaa
aaa
4 4
abab
cdcd
abab
cdcd
3 4
aaaa
aaaa
aaab
```

Output:

```
1
4
12
```

Added by: Pawel Gawrychowski
Date: 2007-11-10
Time limit: 1s
Source limit:50000B
Languages: All

SPOJ Problem Set (classical)

2047. Stone Removing Game

Problem code: REMGAME

Consider the following game. The game is played on a 5 x 5 board. Initially every array cell has a piece in it. Two players remove pieces alternatively from the board. The player can remove any number of consecutive pieces in a row or column. For example, in the configuration depicted below where one indicates a piece, the player can either remove one piece (**A1**, **A2**, or **B1**), or remove two pieces (**A1** and **A2**, or **A1** and **B1**) simultaneously. The game ends when one player is forced to take the last piece, and the other player wins the game.

	1	**2**	**3**	**4**	**5**
A	1	1	0	0	0
B	1	0	0	0	0
C	0	0	0	0	0
D	0	0	0	0	0
E	0	0	0	0	0

Write a program that evaluates board configurations from this game. The program must output "winning" when there exists a winning move that no matter how the opponent responds, it will force the opponent to take the last piece. Otherwise, the program must output "losing".

Input

The first line contains **n**, the number of test cases. For each test case, a 5x5 grid of an initial game configuration is shown.

Output

For each case, output "winning" or "losing".

Example

Input:
```
1
1 0 0 0 0
1 1 0 0 0
1 1 1 0 0
1 1 1 1 0
1 1 1 1 1
```

Output:
```
winning
```

Added by: Chen Xiaohong
Date: 2007-11-13
Time limit: 1s
Source limit:50000B
Languages: All
Resource: Adapted from Taiwan TPC 1999, Harder datasets.

SPOJ Problem Set (classical)

2050. Strange Billboard

Problem code: CERC07B

The marketing and public-relations department of the Czech Technical University has designed a new reconfigurable mechanical Flip-Flop Bill-Board (FFBB). The billboard is a regular two- dimensional grid of R × C square tiles made of plastic. Each plastic tile is white on one side and black on the other. The idea of the billboard is that you can create various pictures by flipping individual tiles over. Such billboards will hang above all entrances to the university and will be used to display simple pictures and advertise upcoming academic events. To change pictures, each billboard is equipped with a "reconfiguration device". The device is just an ordinary long wooden stick that is used to tap the tiles. If you tap a tile, it flips over to the other side, i.e., it changes from white to black or vice versa. Do you agree this idea is very clever?

Unfortunately, the billboard makers did not realize one thing. The tiles are very close to each other and their sides touch. Whenever a tile is tapped, it takes all neighboring tiles with it and all of them flip over together. Therefore, if you want to change the color of a tile, all neighboring tiles change their color too. Neighboring tiles are those that touch each other with the whole side. All inner tiles have 4 neighbors, which means 5 tiles are flipped over when tapped. Border tiles have less neighbors, of course.

example

For example, if you have the billboard configuration shown in the left picture above and tap the tile marked with the cross, you will get the picture on the right. As you can see, the billboard reconfiguration is not so easy under these conditions. Your task is to find the fastest way to "clear" the billboard, i.e., to flip all tiles to their white side.

Input

The input consists of several billboard descriptions. Each description begins with a line containing two integer numbers R and C (1 <= R,C <= 16) specifying the billboard size. Then there are R lines, each containing C characters. The characters can be either an uppercase letter "X" (black) or a dot "." (white). There is one empty line after each map. The input is terminated by two zeros in place of the board size.

Output

For each billboard, print one line containing the sentence "You have to tap T tiles.", where T is the minimal possible number of taps needed to make all squares white. If the situation cannot be solved, output the string "Damaged billboard."instead.

Example

Input:

```
5 5
XX.XX
X.X.X
.XXX.
X.X.X
XX.XX

5 5
.XX.X
.....
..XXX
..X.X
..X..

1 5
...XX

5 5
...X.
...XX
.XX..
..X..
.....

8 9
..XXXXX..
.X.....X.
X..X.X..X
X.......X
X.X...X.X
X..XXX..X
.X.....X.
..XXXXX..

0 0
```

Output:

```
You have to tap 5 tiles.
Damaged billboard.
You have to tap 1 tiles.
You have to tap 2 tiles.
You have to tap 25 tiles.
```

Added by: Rafal
Date: 2007-11-15
Time limit: 1s-20s
Source limit: 50000B
Languages: All
Resource: Central European Programming Contest, Prague 2007

SPOJ Problem Set (classical)

2051. Cell Phone

Problem code: CERC07C

Nowadays, everyone has a cellphone, or even two or three. You probably know where their name comes from. Do you. Cellphones can be moved (they are "mobile") and they use wireless connection to static stations called BTS (Base Transceiver Station). Each BTS covers an area around it and that area is called a cell.

The Czech Technical University runs an experimental private GSM network with a BTS right on top of the building you are in just now. Since the placement of base stations is very important for the network coverage, your task is to create a program that will find the optimal position for a BTS. The program will be given coordinates of "points of interest". The goal is to find a position that will cover the maximal number of these points. It is supposed that a BTS can cover all points that are no further than some given distance R. Therefore, the cell has a circular shape.

example

The picture above shows eight points of interest (little circles) and one of the possible optimal BTS positions (small triangle). For the given distance R, it is not possible to cover more than four points. Notice that the BTS does not need to be placed in an existing point of interest.

Input

The input consists of several scenarios. Each scenario begins with a line containing two integer numbers N and R. N is the number of points of interest,1 <= N <= 2 000. R is the maximal distance the BTS is able to cover, 0 <= R < 10 000. Then there are N lines, each containing two integer numbers X_i, Y_i giving coordinates of the i-th point, |X_i|, |Y_i| < 10 000. All points are distinct, i.e., no two of them will have the same coordinates.

The scenario is followed by one empty line and then the next scenario begins. The last one is followed by a line containing two zeros.

A point lying at the circle boundary (exactly in the distance R) is considered covered. To avoid floating-point inaccuracies, the input points will be selected in such a way that for any possible subset of points S that can be covered by a circle with the radius R +0.001, there will always exist a circle with the radius R that also covers them.

Output

For each scenario, print one line containing the sentence "It is possible to cover M points.", where M is the maximal number of points of interest that may be covered by a single BTS.

Example

Input:

```
8 2
1 2
5 3
5 4
1 4
8 2
4 5
7 5
3 3

2 100
0 100
0 -100

0 0
```

Output:

```
It is possible to cover 4 points.
It is possible to cover 2 points.
```

The first sample input scenario corresponds to the picture, providing that the X axis aims right and Y axis down.

Added by: Rafal
Date: 2007-11-16
Time limit: 20s
Source limit: 50000B
Languages: All
Resource: Central European Programming Contest, Prague 2007

SPOJ Problem Set (classical)

2052. Hexagonal Parcels

Problem code: CERC07H

A civil engineer that has recently graduated from the Czech Technical University encountered an interesting problem and asked us for a help. The problem is more of economical than engineering nature. The engineer needs to connect several buildings with an infrastructure. Unfortunately, the investor is not the owner of all the land between these places. Therefore, some properties have to be bought first.

The land is divided into a regular "grid" of hexagonal parcels, each of them forms an independent unit and has the same value. Some of the parcels belong to the investor. These parcels form four connected areas, each containing one building to be connected with the others. Your task is to find the minimal number of parcels that must be acquired to connect the four given areas.

example

The whole land also has a hexagonal shape with six sides, each consisting of exactly H parcels. The above picture shows a land with H = 4, parcels with letters represent the four areas to be connected. In this case, it is necessary to buy four additional parcels. One of the possible solutions is marked by crosses.

Input

The input contains several scenarios. Each scenario begins with an integer number H,which specifies the size of the land, 2 <= H <= 20. Then there are 2.H - 1 lines representing individual "rows" of the land (always oriented as in the picture). The lines contain one non-space character for each parcel. It means the first line will contain H characters, the second line H +1, and so on. The longest line will be the middle one, with 2.H - 1 characters. Then the "length" descends and the last line contains H parcels, again.

The character representing a parcel will be either a dot (".") for the land that is not owned by the investor, or one of the uppercase letters "A", "B", "C", or "D". The areas of parcels occupied by the same letter will always be connected. It means that between any two parcels in the same area, there exists a path leading only through that area.

Beside the characters representing parcels, the lines may contain any number of spaces at any positions to improve "human readability" of the input. There is always at least one space between two letters (or the dots). After the land description, there will be one empty line and then the next scenario begins. The last scenario is followed by a line containing zero.

Output

For each scenario, output one line with the sentence "You have to buy P parcels.", where P is the minimal number of parcels that must be acquired to make all four areas connected together.

Areas are considered connected, if it is possible to find a path between them that leads only through parcels that have been bought.

Example

Input:

```
4
   B . . C
  . . . . C
 . A . . C .
. A A . . . .
 . A . . . .
  . . . D D
   . . . .

0
```

Output:

```
You have to buy 4 parcels.
```

Added by: Rafal
Date: 2007-11-16
Time limit: 5s
Source limit: 50000B
Languages: All
Resource: Central European Programming Contest, Prague 2007

SPOJ Problem Set (classical)

2053. Key Task

Problem code: CERC07K

The Czech Technical University is rather old - you already know that it celebrates 300 years of its existence in 2007. Some of the university buildings are old as well. And the navigation in old buildings can sometimes be a little bit tricky, because of strange long corridors that fork and join at absolutely unexpected places.

The result is that some first-graders have often difficulties finding the right way to their classes. Therefore, the Student Union has developed a computer game to help the students to practice their orientation skills. The goal of the game is to find the way out of a labyrinth. Your task is to write a verification software that solves this game.

The labyrinth is a 2-dimensional grid of squares, each square is either free or filled with a wall. Some of the free squares may contain doors or keys. There are four different types of keys and doors: blue, yellow, red, and green. Each key can open only doors of the same color. You can move between adjacent free squares vertically or horizontally, diagonal movement is not allowed. You may not go across walls and you cannot leave the labyrinth area. If a square contains a door, you may go there only if you have stepped on a square with an appropriate key before.

Input

The input consists of several maps. Each map begins with a line containing two integer numbers R and C (1 <= R, C <= 100) specifying the map size. Then there are R lines each containing C characters. Each character is one of the following:
Hash mark '#' Wall
Dot '.' Free square
Asterisk '*' Your position
Uppercase letter 'B','Y','R','G' Blue, yellow, red or green door
Uppercase X 'X' Exit

Note that it is allowed to have * more than one exit,
* no exit at all,
* more doors and/or keys of the same color, and
* keys without corresponding doors and vice versa.

You may assume that the marker of your position ("*") will appear exactly once in every map. There is one blank line after each map. The input is terminated by two zeros in place of the map size.

Output

For each map, print one line containing the sentence "Escape possible in S steps.", where S is the smallest possible number of step to reach any of the exits. If no exit can be reached, output the string "The poor student is trapped!" instead. One step is defined as a movement between two adjacent cells. Grabbing a key or unlocking a door does not count as a step.

Example

Input:

```
1 10
*........X

1 3
*#X

3 20
####################
#XY.gBr.*.Rb.G.GG.y#
####################

0 0
```

Output

```
Escape possible in 9 steps.
The poor student is trapped!
Escape possible in 45 steps.
```

Added by: Rafal
Date: 2007-11-16
Time limit: 2s
Source limit:50000B
Languages: All
Resource: Central European Programming Contest, Prague 2007

SPOJ Problem Set (classical)

2054. Gates of Logic

Problem code: CERC07L

The Department of Computer Science and Engineering runs courses dealing not only with algo- rithms but also with computer hardware. One such introductory course explains basic principles of integrated circuits ("chips"), binary logic, boolean algebra, etc. As you may know, the very basic units of logical circuits are called gates. A gate is an element performing one simple logical operation. It can be connected to other gates using lines.

Logical circuits may be drawn as pictures with the gates represented as squares with inputs on the left and outputs on the right. In each square, there is a symbol that determines the gate type: Number 1 denotes an OR gate (its outputs are 0 if and only if there is no input with the value of 1), & is an AND gate (outputs are 1 if and only if there is no 0 input), and = is a XOR gate (outputs are 1 if and only if there is an odd number inputs that have the value of 1).

Your task is to scan such a "picture" and compute values of all named circuit outputs. The lines may split and join again but you may assume that each "value consumer" (input port of a gate or a named output) will be connected to exactly one "value source" (output port of a gate or an input value). There will be no feedback loops, i.e., there exists no cycle that would lead through the same gate twice.

Input

The input contains several pictures. Each picture consists of at least one and at most 200 rows composed of the following characters:

* Space (" "). Empty space in the picture. Spaces are used to indent other characters to appropriate locations, because the exact position of characters is often important. Trailing spaces at the end of input rows may be present but may also be left out.

* Dash ("-"). Horizontal line. It connects characters on its left and right together, those characters will always exist and be able to "accept" the connection.

* Pipe ("|"). Vertical line, connects characters that are directly above and below. Like with the horizontal line, those characters will always accept the connection.

* Plus sign ("+"). Line connection or a bend. Connects characters on all four sides. All characters that are able to accept the connection are considered connected (there will always be at least two). However, there may be sides that contain a non-empty character that is not connected. For example, if a dash is present on a position directly below the plus sign, they are not considered connected.

* Lowercase letter x ("x"). Crossing of two lines without a connection. All four neigh- boring characters will accept the connection. The character above is connected to the one below and the character to the left with the one on the right, but there is no mutual connection between these two pairs.

* Equal sign ("="). Represents an input or output port. It always connects characters on its left and right, at least one of these characters is the port. If there is a port on the left, it may only be a value source. If there is a port on the right, it may only be a value consumer.

* Lowercase letter o ("o"). Negation. There will always be a gate on the left and a port on the right of this character. It makes the particular gate output negated.

* Hash mark ("#"). Gate, which has always a rectangular shape with two vertical and two horizontal sides. The left vertical side may be connected to input ports, the right side to output ports (possibly negated). No two gates will touch each other's side, which means that any two vertically or horizontally neighboring hash marks are always parts of the same gate.

The rectangle size will always be at least 3 characters in both directions, which means there is at least one character inside. All inner characters are empty (spaces), with exactly one exception. That single non-empty character denotes the gate type (note that it may have different meaning than outside the gate area) and will be a digit "one" ("1"), ampersand ("&"), or an equal sign ("=").

* Binary digit ("0"and"1"). Input value of the circuit. It is connected to the character on its right, which is always an equal sign.

* Uppercase letter ("A" through "Z"). Named output of the circuit. It accepts connection from its left, which is always an equal sign. Each letter will appear at most once, which means the number of circuit outputs is between 0 and 26, inclusive. Each picture will be terminated by a row consisting solely of asterisk ("*") characters (at least one). The last picture will be followed by two such rows. No row in the input will be longer than 200 characters.

Output

For each picture, print the values of all named outputs, sorted alphabetically. Each output row should contain three characters: output name (one uppercase letter), equals sign, and a binary value (zero or one). Print one empty line after each test case.

Example

Input:

```
        0=+
          |
          | #######
          +=#      #
            # &    #o=--+
  1=-------=#      #     |
            #      #     |
            #######     +--=### ###
                        |    #=#=#1#o==X
          1=------------x--=# # ###
                    1=---x--=###
                         +------------=Y
**********
1=A
***
*
```

Output:

```
X=0
Y=1

A=1
```

Added by:	Rafal
Date:	2007-11-16
Time limit:	1s
Source limit:	50000B
Languages:	All
Resource:	Central European Programming Contest, Prague 2007

SPOJ Problem Set (classical)

2055. Weird Numbers

Problem code: CERC07N

Binary numbers form the principal basis of computer science. Most of you have heard of other systems, such as ternary, octal, or hexadecimal. You probably know how to use these systems and how to convert numbers between them. But did you know that the system base (radix) could also be negative. One assistant professor at the Czech Technical University has recently met negabinary numbers and other systems with a negative base. Will you help him to convert numbers to and from these systems.

A number N written in the system with a positive base R will always appear as a string of digits between 0 and R - 1, inclusive. A digit at the position P (positions are counted from right to left and starting with zero) represents a value of R^P. This means the value of the digit is multiplied by R^P and values of all positions are summed together. For example, if we use the octal system (radix R = 8), a number written as 17024 has the following value:

1.8^4 + 7.8^3 + 0.8^2 + 2.8^1 + 4.8^0 = 1.4096 + 7.512 + 2.8+4.1 = 7700

With a negative radix -R, the principle remains the same: each digit will have a value of (-R)^P. For example, a negaoctal (radix R = -8) number 17024 counts as:

1.(-8)^4 + 7.(-8)^3 + 0.(-8)^2 + 2.(-8)^1 + 4.(-8)^0 = 1.4096 - 7.512 - 2.8+4.1 = 500

One big advantage of systems with a negative base is that we do not need a minus sign to express negative numbers. A couple of examples for the negabinary system (R = -2):

decimal	negabinary	decimal	negabinary	decimal	negabinary
-10	1010	-3	1101	4	100
-9	1011	-2	10	5	101
-8	1000	-1	11	6	11010
-7	1001	0	0	7	11011
-6	1110	1	1	8	11000
-5	1111	2	110	9	11001
-4	1100	3	111	10	11110

You may notice that the negabinary representation of any integer number is unique, if no "leading zeros" are allowed. The only number that can start with the digit "0", is the zero itself.

Input

The input will contain several conversions, each of them specified on one line. A conversion from the decimal system to some negative-base system will start with a lowercase word "to" followed by a minus sign (with no space before it), the requested base (radix) R, one space, and a decimal number N.

A conversion to the decimal system will start with a lowercase word "from", followed by a minus sign, radix R, one space, and a number written in the system with a base of -R.

The input will be terminated by a line containing a lowercase word "end". All numbers will satisfy the following conditions: 2 <= R <= 10, -1 000 000 <= N <= 1 000 000 (decimal).

Output

For each conversion, print one number on a separate line. If the input used a decimal format, output the same number written in the system with a base -R. If the input contained such a number, output its decimal value.

Both input and output numbers must not contain any leading zeros. The minus sign "-"may only be present with negative numbers written in the decimal system. Any non-negative number or a number written in a negative-base system must not start with it.

Example

Input

```
to-2 10
from-2 1010
to-10 10
to-10 -10
from-10 10
end
```

Output

```
11110
-10
190
10
-10
```

Added by: Rafal
Date: 2007-11-16
Time limit: 2s
Source limit:50000B
Languages: All
Resource: Central European Programming Contest, Prague 2007

SPOJ Problem Set (classical)

2056. Rectangular Polygon

Problem code: CERC07P

In this problem, we will help the Faculty of Civil Engineering. They need a software to analyze ground plans of buildings. Specifically, your task is to detect outlines of a building when all of its corners are given.

example

You may assume that each building is a rectangular polygon with each of its sides being parallel either with X or Y axis. Therefore, each of its vertex angles is exactly either 90 or 270 degrees.

Input

The input contains several buildings. The description of each building starts with a single positive integer N, the number of corners (polygon vertices), 1 <= N <= 1000. Then there are N pairs of integer numbers X_i,Y_i giving coordinates of individual corners, |X_i|, |Y_i| <= 10 000.

You may assume that all corners are listed and no two of them have the same coordinates. The polygon does always exist, it is closed, its sides do not intersect or touch (except neighboring sides, of course), and it contains no "holes" inside. In other words, the outline is formed by one closed line. The order of corners in the input file may be arbitrary.

There is an empty line after each building, then the next one is described. After the last building, there is a single zero that signals the end of input.

Output

For each building, output one line containing N characters without any whitespace between them. The characters should be uppercase letters that specify directions of individual walls (sides) when the building outline is followed. "N" stands for North (the positive direction of the Yaxis), "E" for East (the positive direction of the X axis), "W" for West, and "S" for South. The "walk" should start in the vertex that has been given first in the input and always proceed in the clockwise direction.

Example

Input

```
4
0 0
2 2
0 2
2 0

6
1 1
2 2
0 1
1 0
```

```
0 2
2 0

0
```

Output

```
NESW
WNESWN
```

The second sample input corresponds to the picture.

Added by:	Rafal
Date:	2007-11-16
Time limit:	1s
Source limit:	50000B
Languages:	All
Resource:	Central European Programming Contest, Prague 2007

SPOJ Problem Set (classical)

2058. Reaux! Sham! Beaux!

Problem code: CERC07R

Roshambo - this simple game is known all around the world. In German, it is called "Schnick, Schnack, Schnuck", in Japanese "Janken", in Spanish "Cachipún",in Polish "Papier, kamień, nożyce". The Czechs call it "Kámen, nůžky, papír".

Whatever is the name of the game, its principles remain the same. Two players simultaneously form their hand into one of three possible shapes (symbols): Rock (closed fist), Paper (open hand), or Scissors (two fingers extended). If both of them show the same symbol, it is a tie and no points are given. Otherwise, one of the symbols wins: Rock blunts Scissors, Scissors cut Paper, and Paper covers Rock.

Czech Technical University students also know the game very well and use it to resolve small disputes. Imagine, for example, two students living together in one room. Yesterday evening, there was a small celebration, and in the morning, no one wants to go to the lectures. They agreed that one person would be enough to take notices for both, but who will be the poor one. Roshambo is a very effective way to decide.

Did you know there are even the World Series of Roshambo. Our organizing team would like to host the World Championships in 2009. Your task is to help us in developing a Roshambo scoring system and write a program that evaluates one game between two players.

Since the participants will come from different countries, the system must accept input in various languages. The following table shows names of three Roshambo symbols. Note that in some languages, there may be two different words for the same symbol.

Language	Code	Rock	Scissors	Paper
Czech	cs	Kamen	Nuzky	Papir
English	en	Rock	Scissors	Paper
French	fr	Pierre	Ciseaux	Feuille
German	de	Stein	Schere	Papier
Hungarian	hu	Ko | Koe	Ollo | Olloo	Papir
Italian	it	Sasso | Roccia	Forbice	Carta | Rete
Japanese	jp	Guu	Choki	Paa
Polish	pl	Kamien	Nozyce	Papier
Spanish	es	Piedra	Tijera	Papel

Input

The input contains several games. Each game starts with two lines describing players. Each of these two lines contains two lowercase letters specifying the language used by the player (see the language code in the table above), one space, and a player name. The name will consist from at most twenty upper- or lower-case letters.

After the players description, there are at most 100 lines containing individual rounds. Each round is described by two words separated with one space. The words name the symbol shown by the first and second player, respectively. All symbols are named in the mother tongue of the concerned player. All allowed words are shown in the table above, the first letter will be always in uppercase, all other letters in lowercase.

The last round is followed by a line containing one single dash character ("-") and then the next game begins. The only exception is the last game in the input, which is terminated by a dot (".") instead of the dash.

Output

For each game, print five lines of output. The first line should contain the string "Game #G:", where G is the number of the game, starting with one.

The second line will contain the first player name followed by a colon (":"), one space and the number of rounds won by that player. The number should be followed by one space and the word "points". Use the singular form "point" if (and only if) the number of points of the player equals one.

The third line has the same format and shows the second player's name and points.

The fourth line displays the outcome of the game. It must contain the word "WINNER" followed by a colon, space and the name of the player who gained more points. If both players have the same number of points, the fourth line will contain words "TIED GAME" instead.

The fifth line is left empty to visually separate individual games.

Example

Input

```
cs Pepik
en Johnny
Nuzky Scissors
Papir Rock
Papir Scissors
-
de Gertruda
cs Lenka
Stein Papir
Schere Kamen
.
```

Output

```
Game #1:
Pepik: 1 point
Johnny: 1 point
TIED GAME

Game #2:
Gertruda: 0 points
Lenka: 2 points
WINNER: Lenka
```

Added by: Rafal
Date: 2007-11-16
Time limit: 1s
Source limit: 50000B
Languages: All
Resource: Central European Programming Contest, Prague 2007

SPOJ Problem Set (classical)

2059. Robotic Sort

Problem code: CERC07S

Somewhere deep in the Czech Technical University buildings, there are laboratories for examin- ing mechanical and electrical properties of various materials. In one of yesterday's presentations, you have seen how was one of the laboratories changed into a new multimedia lab. But there are still others, serving to their original purposes.

In this task, you are to write software for a robot that handles samples in such a laboratory. Imagine there are material samples lined up on a running belt. The samples have different heights, which may cause troubles to the next processing unit. To eliminate such troubles, we need to sort the samples by their height into the ascending order.

Reordering is done by a mechanical robot arm, which is able to pick up any number of consecutive samples and turn them round, such that their mutual order is reversed. In other words, one robot operation can reverse the order of samples on positions between A and B.

A possible way to sort the samples is to find the position of the smallest one (P_1) and reverse the order between positions 1 and P_1, which causes the smallest sample to become first. Then we find the second one on position P_2 and reverse the order between 2 and P_2. Then the third sample is located etc.

example

The picture shows a simple example of 6 samples. The smallest one is on the 4th position, therefore, the robot arm reverses the first 4 samples. The second smallest sample is the last one, so the next robot operation will reverse the order of five samples on positions 2-6. The third step will be to reverse the samples 3-4, etc.

Your task is to find the correct sequence of reversal operations that will sort the samples using the above algorithm. If there are more samples with the same height, their mutual order must be preserved: the one that was given first in the initial order must be placed before the others in the final order too.

Input

The input consists of several scenarios. Each scenario is described by two lines. The first line contains one integer number N, the number of samples, 1 <= N <= 100 000. The second line lists exactly N space-separated positive integers, they specify the heights of individual samples and their initial order.

The last scenario is followed by a line containing zero.

Output

For each scenario, output one line with exactly N integers P_1,P_2,...P_N, separated by a space. Each P_i must be an integer (1 <= P_i <= N) giving the position of the i-th sample just before the i-th reversal operation.

Note that if a sample is already on its correct position P_i, you should output the number P_i anyway, indicating that the "interval between P_i and P_i" (a single sample) should be reversed.

Example

Input

```
6
3 4 5 1 6 2
4
3 3 2 1
0
```

Output

```
4 6 4 5 6 6
4 2 4 4
```

Added by: Rafal
Date: 2007-11-16
Time limit: 2s
Source limit: 50000B
Languages: All
Resource: Central European Programming Contest, Prague 2007

SPOJ Problem Set (classical)

2060. Tough Water Level

Problem code: CERC07W

Czech Technical University has the word "technical" in its name. Beside others, this means that lectures in physics are important here. Do you still remember some of the basic physical principles.

For example, imagine a simple glass of water. Or, we will rather call it a cup to avoid ambiguity of this word. So, imagine a simple cup (made of glass) that is partially filled with water. You might have noticed that the stability of such a cup depends on the amount of water inside. If you brush against a full cup, it is relatively easy to knock it down and spill its contents. If the cup is empty, there is fortunately nothing to be spilled, but other than that, the situation does not improve much - it is still easy to knock the cup down with only a little force. The best stability is usually achieved with a "half-full" cup.

In this problem, your task is to determine the water level that makes a cup as much stable as possible. For the purpose of this problem, we will make a simple assumption that the "stability" of a cup is higher, if its center of mass (sometimes also called the center of gravity) is lower (closer to the bottom).

The center of mass can be informally defined as follows: Imagine that glass and water consist of a very large number of very small particles. Then the center of mass is an average of the position of all these particles. The average is weighted by particle masses. Since the density of glass is approximately 2 500 kg.m-3 and the density of water only 1 000 kg.m-3, we will suppose that the mass of a glass particle is 2.5 times higher than the mass of a water particle of the same size.

All cups considered in this problem will have an exact rotary shape. But their radius may vary with the height - some cups are wider at the top, others are wider at the bottom. Also, the thickness of the glass may not be constant.

example

The left figure shows a typical cup considered in this problem. It can be fully described by its height (H), thickness of the bottom (B), and two functions R and T . Both of these functions take a current height as their argument and they give the outer radius (R) and glass thickness (T) in the appropriate height. Please note that the thickness is always measured strictly horizontally and may therefore not reflect the "real thickness" of the glass in its usual meaning.

Input

The input contains several cup descriptions. Each description consists of three lines. The first line contains two numbers: H (cup height) and B (bottom thickness), $0 < B < H <= 100$. The second line contains an expression R(x) (radius), the third line an expression T (x) (glass thickness). All data are given in centimeters. The last description is followed by a line with two zeros.

The expressions will contain only digits ("0" through "9"), decimal points ("."), four basic operators ("+", "-", "*", and "/"), parentheses ("("and")"), and the lowercase letters "x" denoting the input variable (height measured from the cup bottom).

Vx, 0 <= x <= H, the radius will satisfy: 0.1 < R(x) <= 100

Vx, B <= x <= H, the thickness will satisfy: 0.1 <= T (x) < R(x)

Arithmetical operators have their usual meaning and priorities, i.e., multiplication and division have a higher priority than addition and subtraction.

Output

For each cup, output the sentence "Pour L litres / W cm of water.", where L is the amount of water that must be poured into the cup to reach the maximal stability (in litres). W is the water level (in centimeters) measured from the bottom of the cup. Both numbers must be given with exactly three decimal digits.

Example

Input

```
9 1
3+x/6
17/8-x/8
10 1
(x+10)
10/(x+10)
0 0
```

Output

```
Pour 0.030 litres / 3.365 cm of water.
Pour 0.878 litres / 2.193 cm of water.
```

Added by: Rafal
Date: 2007-11-16
Time limit: 5s
Source limit:50000B
Languages: All
Resource: Central European Programming Contest, Prague 2007

SPOJ Problem Set (classical)

2070. Minimum Distance

Problem code: MINDIST

Given an weighted tree, you are to find two nodes A and B of the tree(A and B needn't to be different), such that the length of the path between A and B is less than or equals to a given integer S, and the maximum distance from each node of the tree to this path is minimum.

Input

The first line of the input contains a single integer T, the number of test cases. T blocks follow.

For each test case, the first line contains two space-separated integer N (1<=N<=100000) and S(0<=S<=100000000).N-1 lines follow, each contains three integers X(1<=X<=N), Y(1<=Y<=N) and Z(1<=Z<=1000), denotes that there is an (undirected) edge weighted Z between node X and Y. The input is correct.

Output

T lines, each contains a single integer denoted the minimum distance.

Example

Input:

```
2
5 2
1 2 5
2 3 2
2 4 4
2 5 3
8 6
1 3 2
2 3 2
3 4 6
4 5 3
4 6 4
4 7 2
7 8 3
```

Output:

```
5
5
```

Warning: large input/output data, be careful with certain languages

---

Added by:	Blue Mary
Date:	2007-11-19
Time limit:	15s
Source limit:	50000B
Languages:	All except: C99 strict
Resource:	description by Blue Mary; standard program and test data by g201513

SPOJ Problem Set (classical)

2123. Candy I

Problem code: CANDY

Jennifer is a teacher in the first year of a primary school. She has gone for a trip with her class today. She has taken a packet of candies for each child. Unfortunatelly, the sizes of the packets are not the same.

Jennifer is afraid that each child will want to have the biggest packet of candies and this will lead to quarrels or even fights among children. She wants to avoid this. Therefore, she has decided to open all the packets, count the candies in each packet and move some candies from bigger packets to smaller ones so that each packet will contain the same number of candies. The question is how many candies she has to move.

Input specification

The input file consists of several blocks of data. Each block starts with the number of candy packets **N**(1<= **N** <=10000) followed by **N** integers (each less than 1000) in separate lines, giving the number of candies in each packet. After the last block of data there is the number -1.

Output specification

The output file should contain one line with the smallest number of moves for each block of data. One move consists of taking one candy from a packet and putting it into another one. If it is not possible to have the same number of candies in each packet, output the number -1.

Example

Input file:

```
5
1
1
1
1
6
2
3
4
-1
```

Output file:

```
4
-1
```

Added by: Blue Mary
Date: 2007-12-01
Time limit: 1s
Source limit: 50000B
Languages: All except: C99 strict
Resource: IPSC 1999

SPOJ Problem Set (classical)

2124. Last Non-Zero Digit of Factorials

Problem code: FCTRL4

Tom is fascinated by big numbers. Most of all he likes factorials. First, he computed **N!** for some small values of **N**. But factorials were getting longer and longer very quickly and each of them had a long sequence of zeroes at its end. Since he was unable to consider so many digits in his research and he thought that the research on zeroes would be boring, he decided to focus on the last non-zero digit of factorials.

For example 4!=24 so the last non-zero digit is 4. For **N**=5 we have 5!=120 and the last non-zero digit 2.

Tom needs to know the last non-zero digit of **N!** for several specific values of **N**. Help him please.

Input specification

The input file consists of several positive integers (less than 10^{100}) delimited by whitespace.

Output specification

The output file contains the last non-zero digit of N! for each integer N from the input file. Digits should be delimited by whitespace.

Example

Input file:

```
1
2
3
4
5
6
7
8
```

Output file:

```
1
2
6
4
2
2
4
2
```

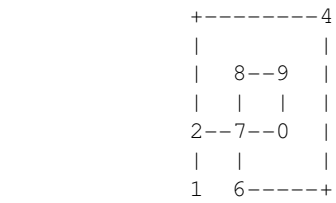
Added by: Blue Mary
Date: 2007-12-01
Time limit: 1s
Source limit: 50000B
Languages: All except: C99 strict
Resource: IPSC 1999

SPOJ Problem Set (classical)

2125. Number Labyrinth

Problem code: LABYR2

Fred is a robotic mouse built by a group of students of artificial intelligence. Fred can move around in the labyrinth shown in the picture below. Whenever Fred comes to a place marked by a number, he has to choose one of the possible directions. Behavior of the mouse should look chaotic and complex enough so that it will impress students' supervisor.



Numbered places in the labyrinth are called nodes. Fred has one integer X stored in its memory and can perform some calculations. In each node (except node 1) he chooses a direction according to X, decreases X by 1 and goes to the chosen node. The direction is chosen according to this rules:

```
Node 2: Compute X mod 3.
        If the result is 0, go to 7
                         1, go to 1
                         2, go to 4.

Node 4: Let Y be X written backwards (in decimal system).
        If Y>X then go to 6 otherwise go to 2.

Node 6: Compute the number of digits of X (in decimal system).
        If the result is even then go to 4 otherwise go to 7.

Node 7: Compute (X*X) mod 7.
        If the result is 0 go to 2
                         1 go to 6
                         2 go to 8
                         4 go to 0.

Node 8: Compute X mod 5.
        If the result is 2 or 3 then go to 7 otherwise go to 9.

Node 9: If you have come from 8 then go to 0.
        If you have come from 0 then go to 8.

Node 0: Let Y be the third least significant digit of X in decimal system
        (if X<100 then Y=0). If Y<=7 then go to 7 otherwise go to 9.
```

At the beginning of each experiment, the experimenter puts the mouse in the node 0 and initializes value X by voice. After that, the mouse starts to move. The mouse displays current value of X on its digital display. The experiment finishes when the mouse enters the node 1, the result of the experiment is the number displayed. If the value of X decreases to zero, the experiment fails and its result is -1.

Input file description

The input file contains several initial values of X(less than two-million) as they were told by the experimenter.

Output file description

For each value of X in the input file write to a separate line of the output file the result of the corresponding experiment (see example output).

Example

Input file:

```
thirteen
fourteen
one-thousand
one-million-three-hundred-and-twenty-five-thousand-nine-hundred-and-seventy-nine
```

Output file:

```
-1
9
789
1325784
```

Blue Mary's Note: new test cases were added on Jun.6, 2008.

Another Note: One of the test cases has something wrong & it has been fixed in Jul. 19, 2008. Thanks to Robert Gerbicz and Stephen Merriman's discussion in the forum.

Added by: Blue Mary
Date: 2007-12-01
Time limit: 1s
Source limit:50000B
Languages: All except: C99 strict
Resource: IPSC 1999

SPOJ Problem Set (classical)

2126. Panel

Problem code: PANEL

Bulbville is a small town. However, its main square resembles a scene from Las Vegas. It is full of colorful blinking pipes and panels advertising various local companies. Nevertheless, a particular influential family residing at the main square has started to complain that a certain advertising panel shines into the windows of their residence. Therefore, they demand that the panel be switched off during the night.

The panel is a square consisting of N rows of bulbs with N bulbs in each row. However, it has only 2N switches - one for each row and one for each column. The switch for a given row (or a column) turns off all the bulbs in a given row (or a column) which are on and turns on all the bulbs which are off. It might not be possible to turn off all the bulbs of the panel. Your task is to determine a configuration of the panel with the least number of shining bulbs that can be achieved using the switches.

Input specification

The input describes the initial configuration of the advertising panel. The first line consists of a single number N(N<=50). Each of the next N lines describes one row of the panel starting from the top. It contains N numbers separated by spaces - each number standing for one bulb, 1 if the bulb is on, 0 if the bulb is off, starting from the left.

Output specification

The first and the only line of the output contains one integer M, giving the minimum number of shining bulbs. N lines follow, each contains a single integer 1 or 0 denoting whether you wants to use the switch for the *i*-th row. N lines follow, each contains a single integer 1 or 0 denoting whether you wants to use the switch for the *i*-th column.

Example

Input file:

```
4
1 0 1 1
1 0 1 0
1 1 1 0
0 1 0 0
```

Output file:

```
3
0
0
0
1
1
0
1
1
```

Added by: Blue Mary
Date: 2007-12-01
Time limit: 2s
Source limit: 50000B
Languages: All except: C99 strict
Resource: IPSC 1999

SPOJ Problem Set (classical)

2127. Rain

Problem code: RAIN3

Doctor Jones is a famous archeologist. He did some research on the Tiribaki Islands recently. His most famous discovery was the Meteoronome - a machine with a yellow button used by the Tiribakian highest priest to predict the weather. The Meteoronome had been set up by the gods at the Beginning of Time. Tiribakians pressed the button every day. As a result, the Meteoronome produced a number - the expected rainfall in millimetres for the next day. More precisely, after *i* button hits (counted since the Beginning of Time) Meteoronome gives the expected rainfall for the day *i* since the Beginning of Time.

Unfortunately, the Meteoronome has not been used for several thousands of years and nobody knows how many steps should be performed to reach the current date. Researchers have spent a lot of effort to find out how the Meteoronome works. A mathematical model has been proposed: The Meteoronome is initialized by a pair of integers, s[0] and t[0]. For the *i*-th step, the Meteoronome computes the values

```
s[i] = (78901 + 31*s[i-1]) mod  699037
t[i] = (23456 + 64*t[i-1]) mod 2097151
```

The output of the i-th step is the number

```
a[i]=(s[i] mod 100 + 1) * (t[i] mod 100 + 1)
```

Doctor Jones's friend, Ms. Linda Watson, is now planning a holiday on Tiribaki Islands. She would like to stay there as long as possible but she hates the rain. She can stand no more than M millimetres of rainfall during her entire stay on Tiribaki.

Doctor Jones wants to help his friend and to compute the longest period which she can safely stay on Tiribaki. He simulated *N* steps of the Meteoronome. This way, he obtained a sequence of numbers a[1],a[2],...,a[N] which represent predictions for *N* subsequent days. Now he wants to find the largest *K* such that for each period of at most *K* subsequent days from day *i* to day *j* the sum of the predictions a[i]+a[i+1]+...+a[j] is less than or equal to *M*. Linda can be sure that if she stays on Tiribaki for at most *K* days, she can endure the rain (provided that *N* is large enough).

Input specification

The input file consists of several blocks of data. The first line of the input file contains the number of blocks. Each block contains four integers delimited by whitespace: s[0], t[0] - the initial values for the Meteoronome, *N* (1<= *N* <=1500000)- the length of the sequence and *M* - the maximum sum of a subsequence. All the input data fits into 32-bit signed integer.

Output specification

The output file contains one line for each block of input data. In this line there is a single integer *K* as specified above.

Example

Input file:

```
1
123456 123456 10 10000
```

Output file:

```
2
```

Note, that the sequence produced by Meteoronome for this input file is 4664,1248,267,4900,837,4048,990,6935,1155,490. No subsequence of length 2 has sum greater than 10000 and there are subsequences of length 3 with greater sum.

Added by: Blue Mary
Date: 2007-12-01
Time limit: 2s
Source limit: 50000B
Languages: All except: C99 strict
Resource: IPSC 1999

SPOJ Problem Set (classical)

2128. K-In-A-Row

Problem code: KROW

Hansel and Gretel do not listen to their teacher at school. Instead they secretly play a game called K-In-A-Row. One day in the evening they started to argue who had won more games that day. They collected all the papers they had used for playing the game and they started to count how many times each of them had won. But it was very tedious and they were sleepy. Help them to count how many games each of them had won.

K-In-A-Row is played in a square grid with M times N squares. Two players alternate in their moves. A player chooses an empty square and fills in his/her sign (Hansel uses cross 'x' and Gretel uses circle 'o'). The game is won by the player who first places at least K his/her own signs in a row (either horizontally, vertically or in one of the two diagonal directions). The game stops immediately after one of the players completes K of his/her signs in a row; thus it may never happen that both players have completed K of their signs in a row. If no player creates such a row, nobody wins.

Input file specification

The first line contains the number of games L. It is followed by L blocks, each describing one game. Each block starts with a line containing 3 numbers M, N and K. Numbers M and N give the size of the grid (M,N<200) and K is the length of the required row. The following N lines each containing M characters describe the situation after the end of the game. Character '.' denotes an empty field, characters 'x' and 'o' denote fields marked by Hansel and Gretel respectively. You may assume that the input is correct.

Output file specification

The output file consists of two numbers separated by a colon ':'. The first number denotes the number of the games won by Hansel, the second one gives the number of the games won by Gretel.

Example

Input file:

```
2
3 3 3
.x.
.xo
oox
4 7 4
....
..x.
ooox
oxx.
oox.
o.ox
```

```
..xx
```

Output file:

```
0:1
```

Added by: Blue Mary
Date: 2007-12-01
Time limit: 2s
Source limit: 50000B
Languages: All except: C99 strict
Resource: IPSC 2000

SPOJ Problem Set (classical)

2129. Cake

Problem code: CAKE2

Some time ago a VERY huge cake was made in the village called Nalomena Trieska. Well, it was infinitely large and infinitely thin. For our needs it looked exactly like an infinite plane. It was not very tasty, so nobody wanted to eat it. Instead, local children started to play with it. Each of them drew one straight line on the plane. These lines divided the plane into many parts. For a few hours the children were happy, they jumped from one part into another and played other similar games. But then little Tommy suddenly asked: "How many parts does the cake have?" "1999." answered Martin. "No, 2000 !" replied Richard. "Well, I think it's only 1748." stated Michael. And they started to argue. Now their parents need your help, because the children spend all their time counting the parts of the cake.

Input file specification

The first line of the input file contains the number of straight lines N(N<=3000). Each of the next N lines contains four integers x1,y1,x2,y2 (the absolute value of each number is at most 10000). These integers are the coordinates of two different points in the plane [x1,y1] and [x2,y2]. These two points determine one straight line in the plane. You can assume that no two straight lines are the same.

Output file specification

The output file contains a single integer giving the number of parts into which given N lines divide the plane.

Example

Input file:

```
4
5 0 0 5
4 0 4 5
2 4 3 4
1 1 1 5
```

Output file:

```
9
```

Added some unofficial test cases.

Added by: Blue Mary
Date: 2007-12-01
Time limit: 1s-7s
Source limit:50000B
Languages: All except: C99 strict
Resource: IPSC 2000

SPOJ Problem Set (classical)

2130. Trolls

Problem code: TROLLS

In a secret forest, there are many trolls. They are intelligent and most of them even knows programming languages C or PASCAL. They have written many programs. With their super computers, their programs will run for a very short time and they can get the correct answer. Being an excellent programmer, you can even come up with the answer without computers!!!

Input

There's no real input file. The four programs can be downloaded here.

Output

For **program1.txt** and **program2.txt** in the zip file:

Your output should contain a single integer denoting the number of "*" the program will output.

For **program3.txt** and **program4.txt** in the zip file:

Your output should contain a single integer denoting the number the program will output.

You can merge the 4 output lines together to get the real output file and submit it.

Example

If **program1.txt** is

```
var i:integer;
begin
  for i:=2 to 8 do write('*');
end.
```

```
#include <stdio.h>
void main() {
  int i;
  for(i=2; i<=8; i++) printf("*");
}
```

The corresponding line should be:

```
7
```

If **program3.txt** is

```
var a, i: integer;

begin
  a := 0;
```

```
  for i:= 1 to 9 do
    a := a * 10 + i;
  writeln(a);
end.
```

```
#include <stdio.h>

void main() {
  int i, a;
  a = 0;
  for(i=1; i<=9; i++)
    a = a * 10 + i;
  printf("%d\n", a);
}
```

The correspoding line should be

```
123456789
```

Note

The numbers fit into the type "int" in C program or "integer" in PASCAL program in the zip file can be arbitrarily large because the computers used by trolls are **super computers**.

Added by: Blue Mary
Date: 2007-12-01
Time limit: 1s
Source limit:50000B
Languages: TEXT
Resource: IPSC 2000 and IPSC 2001

SPOJ Problem Set (classical)

2131. Get Back!

Problem code: GETBACK

Somewhere deep in a desert lives a small, completely forgotten tribe N'Gubara. All the men, women and children of the tribe live with a few camels in the N'Gubara oasis (the N'Gubara people don't have great creativity when it comes to geography names). These poor people have only the N'Gubara well, a few acres of irrigated land and the N'Gubara cave. Yes, the cave.

You do not remember exactly how you have come to N'Gubara. Maybe your car broke when you travelled across the desert. Maybe you jumped out of an airplane. Maybe you were brought here by Martians. But this does not matter now. You know that you are here, pretty far from any civilization. And you desperately want to go home.

The only possibility how to get home is to walk across the desert to Desertville, the closest city. You have to use paths in the desert shown in your map. Each such path connects two restpoints. The N'Gubara oasis and Desertville are also considered to be restpoints. You can walk, but you need water. For each mile travelled, you have to drink one unit of water. You are able to carry at most C units of water at once. Thus, you can never walk more than C miles without replenishing your water supply. The shortest way to Desertville is probably much longer than C miles. It would thus seem that you will stay in N'Gubara forever, but there's a trick: at the end of each path, there is a restpoint with an empty water reservoir. You can transport water into a reservoir from N'Gubara or from other reservoirs where you stored some water previously. You can then use the stored water later on (the water doesn't evaporate). Of course, you can take only as much water from any reservoir as you brought into it. You may use as much water from the N'Gubara oasis as you need, but since water is very valuable in the desert, you promised that you will use only the minimal amount needed for your return to Desertville. In this task, we want you to compute the minimal amount of water you need.

Input file specification

Input contains several test cases, the number of them is given in the very first line.

The first line contains three integers N, M and C, where N is the number of the restpoints, M is the number of the paths and C is your carrying capacity. M lines follow, each describing one path on the map. Each line contains three numbers x, y and l, where x and y are the restpoints joined by a path and l is the length of that path in miles. The restpoints are numbered from 1 to N, where restpoint 1 is the N'Gubara oasis and restpoint N is Desertville.

You may assume all the numbers in the input file are non-negative integers less than 100.

Note: For both input files you may assume that every cycle in the map passes through the restpoint N (i.e. Desertville). A cycle is a sequence of distinct restpoints r1, r2, ..., rk (k>2) such that there is a path from r1 to r2, from r2 to r3, ..., from rk to r1.

Output file specification

For each test case:

Your output file should contain a single integer - the minimum amount of water needed for you to get from N'Gubara to Desertville. If there is no possibility how to get to Desertville with your current carrying capacity C, you should output the number -1.

Example

Input file:

```
1
9 10 25
1 2 3
2 3 12
3 4 4
3 5 9
4 9 13
5 9 5
2 6 10
6 7 10
7 8 10
8 9 10
```

Output file:

```
65
```

Note: You can get to Desertville as follows: first you take 25 units of water from N'Gubara, go to restpoint 2, leave there 19 units in the reservoir and go back to N'Gubara. Then you repeat this trip, bringing additional 19 units to restpoint 2. Last, you take 15 units from N'Gubara to restpoint 2. Now you have 19+19+12 = 50 units of water here. You take a round trip to point 3 and back, leaving there 1 unit of water. Now you take all water left from point 2, go to point 3, take the one unit from point 3 (now you have 25-12+1=14 units) and go via restpoint 5 to Desertville.

Added by: Blue Mary
Date: 2007-12-01
Time limit: 6s
Source limit:50000B
Languages: All except: C99 strict
Resource: IPSC 2000

SPOJ Problem Set (classical)

2132. Puzzle

Problem code: PUZZLE2

Sam loves all kinds of puzzles. Recently he received a very special puzzle - a grid of N x M squares, each square is occupied either by a red or by a blue pebble. The puzzle is controlled by several buttons.

Each row of the grid has its corresponding black button. When you press a black button, a complicated mechanism of the puzzle flips the colors of all pebbles in the row corresponding to that button (all red pebbles in this row become blue and vice versa).

Each column of the grid has its corresponding white button. When you press exactly two white buttons simultaneously, the puzzle mechanism exchanges the contents of the two columns corresponding to those buttons without changing the order of the pebbles in the columns.

Sam has found the puzzle very interesting. Unfortunately, he lent it to his daughter Ann yesterday. She was able to understand the idea of the puzzle very quickly and she gave him this teasing task. She drew two arrangements - initial and final and she changed the colors of the pebbles according to the initial arrangement. Sam's task is to decide whether there exists a sequence of puzzle operations which transforms the initial arrangement of the puzzle to the final arrangement.

Input file specification

The input file contains several blocks of input data. The first line of the input file contains K - the number of blocks. The first line of each block contains integers N(1<=N<=100) and M(1<=M<=100). Each of the next N lines contain M words (either RED or BLUE), describing the initial Ann's arrangement of the colors of the pebbles. One blank line follows. The next N lines contain M words corresponding to the final arrangement of the colors.

Output file specification

The output file contains K lines, the i-th line corresponds to the i-th block of the input data. Each line contains either YES or NO, the answer to the Ann's question.

Example

Input file:

```
2
3 4
BLUE RED BLUE RED
RED BLUE BLUE RED
BLUE BLUE BLUE BLUE

BLUE RED BLUE RED
RED RED BLUE BLUE
BLUE BLUE BLUE BLUE
2 2
BLUE BLUE
```

```
BLUE RED

RED RED
RED RED
```

Output file

```
YES
NO
```

Note: In the first block of the input data it suffices to use the black button for the second row and then to press the white buttons for the first and the third columns simultaneously.

Added by: Blue Mary
Date: 2007-12-01
Time limit: 2s
Source limit: 50000B
Languages: All except: C99 strict
Resource: IPSC 2000

SPOJ Problem Set (classical)

2136. Candy II

Problem code: CANDY2

Little Michael loves candies. Most of all, he likes chocolate, strawberry and banana flavored ones. No wonder that he has candy bags everywhere - there are at least four bags on his table, one or two in the pockets of his jeans, and one under his bed (just in case). Each bag contains some candies of all three flavors. Whenever he wants to eat a candy, he finds the nearest bag (which is usually is not very far because he has really A LOT of them) and eats the candy he wants.

Yesterday, he wanted a strawberry one, so he opened one of his bags and... It is almost impossible to describe how great his disappointment was when he found out that there were no strawberry candies left in that bag. To make the matters worse, there were also none in the second bag he found. He was sure that he had lots of strawberry candies left, but he didn't know in which bags they were. Therefore, he decided to reorganize his candies, and keep the candies of the three different flavors in three distinct bags. He brought all his bags into the center of his room and realized, that there are really an awful lot of them.

Michael has N bags full of candies. He knows the number of candies of each flavor in each bag. He wants to put all chocolate ones into one bag, all strawberry ones into another bag and all banana ones into yet another bag. He has to move the candies one-by-one, because he always has to look at it to determine its flavor. Moving one candy from one bag into another takes 1 second. Your task is to select the bag for each flavor, so that the total time required for Michael to move all the candies into their bags would be minimal.

Input file specification

The first line of the input file contains a single integer N - the number of bags (N<=5000). Each of the following N lines consists of three numbers ci, si, bi - the numbers of chocolate, strawberry and banana candies in the i-th bag. The bags are numbered from 0 to N-1 in the order in which they appear in the input.

Output file specification

Output file should contain three lines with the following text:

```
C[Bag for chocolate candies]
S[Bag for strawberry candies]
B[Bag for banana candies]
```

The numbers C, S, B have to be such that the total number of the required moves is minimal. If there are more solutions, you may choose any of them.

Example

Input file:

```
5
10 10 10
40 39 40
10 20 30
30 20 10
1 2 27
```

Output file:

```
3
1
2
```

Note: In this case Michael has to move 200 candies. If the bags for the different flavors were chosen in any other way, he would have to move more than 200 candies.

Note: the test data is very naive for this problem.

Added by: Blue Mary
Date: 2007-12-01
Time limit: 1s
Source limit:50000B
Languages: All except: C99 strict
Resource: IPSC 2001

SPOJ Problem Set (classical)

2138. Pibonacci

Problem code: PIB

You might have heard of the Fibonacci numbers and of the number **pi**. If you let these two ideas merge, a new and esoteric concept comes into being: **the Pibonacci numbers**. These can be defined for real **x>=0** by:

P(x) = 1	for **0<=x<4**
P(x) = P(x-1) + P(x-pi)	for **4<=x**,

where **pi = 3.1415926535...** In this problem, you are asked to compute **P(x)** for a given **x**.

Input file specification

The input file contains several non-negative integer numbers (less than 30000) each on a separate line. The last line contains -1 marking the end of the input file.

Output file specification

For each non-negative real number in the input file, output the corresponding Pibonacci number. If the Pibonacci number be more than 50 digits long, divide it on the consecutive lines, outputting 50 digits on each (the last line may contain less digits).

Example

Input file:

```
0
4
11
-1
```

Output file:

```
1
2
20
```

Added by:	Blue Mary
Date:	2007-12-01
Time limit:	42s
Source limit:	50000B
Languages:	All except: C99 strict
Resource:	IPSC 2001; standard program and test data by Blue Mary

SPOJ Problem Set (classical)

2139. Gossipers

Problem code: GOSSIPER

Doulnee Keltchow is a small town in the middle of nowhere; what makes it so famous is the number of gossipers who live there. Every morning, each gossiper finds out a new gossip, a gossip so unique that nobody else in the town knows it. The gossipers talk, gossip and exchange rumors all day long. What happens when two gossipers meet? Of course, they exchange all the gossips they have heard so far. Your task is to determine whether every gossiper will know all the gossips by the end of the day.

Input file specification

The input file consists of multiple test cases separated by blank lines. On the first line of every test case there are two positive integers N(1<=N<=2100) and M(1<=M<=12000), where N is the number of gossipers and M is the number of meetings. On the next N lines there are the names of the gossipers. The name of each gossiper is a single word consisting of lower- and uppercase letters. The following M lines describe the meetings in the order they happened. Each meeting is described by two distinct names of the gossipers separated by a single space. The values M=N=0 indicate the end of the input file.

Output file specification

The output file should contain for each test case one line containing a single word "YES" if every gossiper knows all the gossips, or "NO", otherwise.

Example

Input file:

```
3 3
Alice
Bob
Cindy
Alice Bob
Bob Cindy
Cindy Alice

4 4
Kirk
Lucy
Mike
Nancy
Kirk Lucy
Lucy Mike
Mike Nancy
Nancy Lucy

0 0
```

Output file:
YES
NO

Added by: Blue Mary
Date: 2007-12-01
Time limit: 2s
Source limit: 50000B
Languages: All except: C99 strict
Resource: IPSC 2001

SPOJ Problem Set (classical)

2140. (un)Fair Play

Problem code: FAIRONOT

It is not an easy job to be a coach of a football team. Especially if you do not coach great teams like Ajax, Inter, Dynamo (ok, fill in the name of your dream team), but only a mediocre one like FC Dead Horse, playing in the second league. The season is almost over, only a few matches are left to play. All of sudden the team manager comes to you and tells you bad news: the main sponsor of your club is not happy with your results and decided to stop sponsoring your team, which probably means the end of your club. The sponsor's decision is final and there is no way to change it unless... unless your team miraculously wins the league.

The manager left you in deep thought. If you increase the number of practices and offer players a generous bonus for each match, you may be able to win all the remaining matches. Is that enough? You also have to make sure that teams with many points lose against teams with few points so that in the end, your team will have more points than any other team. You know some of the referees and can bribe them to manipulate the result of each match. But first you need to figure out how to manipulate the results and whether it is possible at all.

Problem Description

There are N teams numbered 1 through N, your team has the number N. The current number of points of each team and the list of remaining matches are given. Your task is to find out whether it is possible to manipulate each remaining match so that the team N will finish with strictly more points than any other team. In every match, the winning team gets 2 points, the losing team gets 0. If the match ends with a draw, both teams get 1 point.

Input file specification

The input file consists of several blocks. Each block has the following form: The first line contains two numbers N(1<= N <=100) and M(0<= M <=1000). The next line contains N numbers separated by spaces giving the current number of points of teams 1, 2, ..., N respectively. The following M lines describe the remaining matches. Each line corresponds to one match and contains two numbers a and b (a not equal to b) identifying the teams that will play in the given match. The last block is followed by a "-1"(without quotes) on a separate line.

Output file specification

For each block in the input file, output YES if you can manipulate the remaining matches so that the team N would win the league, or NO otherwise.

Example

Input file:

```
5 8
2 1 0 0 1
1 2
3 4
2 3
4 5
3 1
2 4
1 4
3 5
5 4
4 4 1 0 3
1 3
2 3
3 4
4 5
-1
```

Output file:

```
YES
NO
```

Added by: Blue Mary
Date: 2007-12-01
Time limit: 3s
Source limit: 50000B
Languages: All except: C99 strict
Resource: IPSC 2002

SPOJ Problem Set (classical)

2141. Golden Garden

Problem code: GARDEN

Little Jelly is playing in the Golden Garden alone. She is such a pretty girl that the Evil Uncle wants to catch her. The Evil Uncle drives his flying saucer over the Golden Garden and trys to inhale Little Jelly.

It is known that the Golden Garden is a perfect rectangle and the Evil Uncle's flying saucer can inhale her if and only if she is in a circle with a certian radius around the flying saucer.

Your task is to avoid the Evil Uncle catching Little Jelly. First of all, you should find out the common area of the gardan and the inhaling circle.

Input

There are multiple test cases in the input file. First line of each test case are three integers x, y and r, For the coordinate of the center of the inhaling circle.

The second line contains four integers x1, y1, x2 and y2. Which are the coordinate of the Gold Garden. The Gold Garden is such a perfect rectangle so that its sides are parallel to the axis.

All numbers in the input are integers and do not reach 1000 by their absolute values. Sides of the Golden Garden are non-zero.

Output

For each test cases, output only one line with one real number which is the common area of the garden and the circle. Your answer must be accurate up to 0.000001.

Note

There are two input files for this problem. If you get Wrong Answer, click on the "wrong answer" to get more details. For each test file, my judge will give you the id of the test case for which your program gives a wrong response.

Example

Input:
```
0 0 5
3 3 7 7
```

Output:
```
0.547426365104
```

Warning: enormous input/output data, be careful with certain languages

---

Added by:	Blue Mary
Date:	2007-12-01
Time limit:	7s-13s
Source limit:	50000B
Languages:	All except: C99 strict
Resource:	Description, standard program & test data by g201513

SPOJ Problem Set (classical)

2143. Dependency Problems

Problem code: DEPEND

We bought a brand new computer and now we would like to install an operating system. The only problem is that our chosen operating system consists of many packages and they cannot be installed in an arbitrary order. E.g. you cannot install the package tuxracer, which depends on the package libSDL, before you install libSDL. But libSDL can depend on another packages and so on. The packages may only be installed one at a time. You may install a package only if you already installed all packages it depends on. Your task is to determine how many packages can be installed on our computer.

Input file specification

The input file contains a single line for each available package. The line for each package P begins with the name of the package. The name of each package is a non-empty string of printable characters containing no spaces. Following the name of the package P is the dependency list of P. The dependency list is simply a list of names of packages that P depends on, separated by spaces. A whitespace followed by a single 0 (zero) is at the end of each line. You may assume that no package has the name '0'.

The dependency list of a package P may be empty; in that case, P does not depend on any packages and may be installed immediately. It is possible that a package Q occurs in the dependency list of a package P more than once; this merely means that P depends on Q, nothing more. Only the packages that have a dependency list in the input file are available and may be installed. It is possible that a package P depends on a package that is not available. Such a package cannot be installed.

The number of lines in the input file will be less than 9000.

Output file specification

The output consists of one number -- the maximum number of packages that may be installed on the computer.

Example

Input file

```
a b c b 0
b c 0
c 0
d e f 0
e f 0
f e 0
g h 0
```

Output file

```
3
```

Note

Package c can be installed immediately. Package b depends only on c, and hence can also be installed once we installed c. Finally, package a depends only on b and c and can now also be installed. It is easy to verify that no other packages can be installed.

Added by: Blue Mary
Date: 2007-12-01
Time limit: 1s-5s
Source limit: 50000B
Languages: All except: C99 strict
Resource: IPSC 2003

SPOJ Problem Set (classical)

2147. Root of a Linear Equation

Problem code: ROOT

Given two nonnegative integers *a* and *b*, you are to generate the solution of the equation **b*x=a**.

Input

The number of test cases is given in the very first line. For each test case there is a single line containing two integers *a* and *b* without any leading zeroes, separated by a single space.

Tip: For more than 95% of test cases, **int** in C/C++/Java or **longint** in Pascal is enough.

The input file is about 1.4 KB.

Output

For each test case, output a single line containing the root in decimal cyclic notations, or "Invalid Input!!!"(without quotes) if the solution is not unique or the solution doesn't exist. See the example for more details.

The output file is about 1.3MB.

Example

Input:

```
8
1 1
1 2
1 3
1 4
1 6
24 2
15 7
1 89
```

Output:

```
1.0
0.5
0.(3)
0.25
0.1(6)
12.0
2.(142857)
0.(01123595505617977528089887640449438202247191)
```

Added by: Blue Mary
Date: 2007-12-01
Time limit: 15s
Source limit: 50000B
Languages: All except: C99 strict
Resource: UVA 942; USACO 79; test data by Blue Mary

SPOJ Problem Set (classical)

2148. Candy III

Problem code: CANDY3

A class went to a school trip. And, as usually, all **N** kids have got their backpacks stuffed with candy. But soon quarrels started all over the place, as some of the kids had more candies than others. Soon, the teacher realized that he has to step in: "Everybody, listen! Put all the candies you have on this table here!"

Soon, there was quite a large heap of candies on the teacher’s table. "Now, I will divide the candies into **N** equal heaps and everyone will get one of them." announced the teacher.

"Wait, is this really possible?" wondered some of the smarter kids.

Problem specification

You are given the number of candies each child brought. Find out whether the teacher can divide the candies into **N** exactly equal heaps. (For the purpose of this task, all candies are of the same type.)

Input specification

The first line of the input file contains an integer **T** specifying the number of test cases. Each test case is preceded by a blank line.

Each test case looks as follows: The first line contains **N** : the number of children. Each of the next **N** lines contains the number of candies one child brought.

Output specification

For each of the test cases output a single line with a single word "YES" if the candies can be distributed equally, or "NO" otherwise.

Example

Input:

```
2

5
5
2
7
3
8

6
7
11
2
7
```

```
3
4
```

Output:

```
YES
NO
```

Note: the input file will not exceed 1MB.

Added by: Blue Mary
Date: 2007-12-01
Time limit: 1s-2s
Source limit: 50000B
Languages: All except: C99 strict
Resource: IPSC 2006

SPOJ Problem Set (classical)

2149. Biased Standings

Problem code: BAISED

Usually, results of competitions are based on the scores of participants. However, we are planning a change for the next year of IPSC. During the registration each team will be able to enter a single positive integer : their preferred place in the ranklist. We would take all these preferences into account, and at the end of the competition we will simply announce a ranklist that would please all of you.

But wait... How would that ranklist look like if it won't be possible to satisfy all the requests?

Suppose that we already have a ranklist. For each team, compute the distance between their preferred place and their place in the ranklist. The sum of these distances will be called the badness of this ranklist.

Problem specification

Given team names and their preferred placements find one ranklist with the minimal possible badness.

Input specification

The first line of the input file contains an integer **T** specifying the number of test cases. Each test case is preceded by a blank line.

Each test case looks as follows: The first line contains **N** : the number of teams participating in the competition. Each of the next **N** lines contains a team name (a string of letters and numbers) and its preferred place (an integer between 1 and **N**, inclusive). No two team names will be equal.

Output specification

For each of the test cases output a single line with a single integer : the badness of the best ranklist for the given teams.

Example

Input:

```
2

7
noobz 1
llamas 2
Winn3rz 2
5thwheel 1
NotoricCoders 5
StrangeCase 7
WhoKnows 7

3
ThreeHeadedMonkey 1
```

```
MoscowSUx13 1
NeedForSuccess 1
```

Output:

```
5
3
```

Explanation:

In the first test case, one possible ranklist with the minimal badness is:

```
1. noobz
2. llamas
3. Winn3rz
4. 5thwheel
5. NotoricCoders
6. WhoKnows
7. StrangeCase
```

In the second test case all ranklists are equally good.

Note: the input file will not exceed 5MB.

Added by: Blue Mary
Date: 2007-12-01
Time limit: 1s-3s
Source limit: 50000B
Languages: All except: C99 strict
Resource: IPSC 2006

SPOJ Problem Set (classical)

2150. Counting Subsequences

Problem code: SUBSEQ

"47 is the quintessential random number," states the 47 society. And there might be a grain of truth in that.

For example, the first ten digits of the Euler's constant are:

```
2 7 1 8 2 8 1 8 2 8
```

And what's their sum? Of course, it is 47.

Try walking around with your eyes open. You may be sure that soon you will start discovering occurences of the number 47 everywhere.

Problem specification

You are given a sequence **S** of integers we saw somewhere in the nature. Your task will be to compute how strongly does this sequence support the above claims.

We will call a **continuous** subsequence of **S** interesting if the sum of its terms is equal to 47.

E.g., consider the sequence **S** = (24, 17, 23, 24, 5, 47). Here we have two interesting continuous subsequences: the sequence (23, 24) and the sequence (47).

Given a sequence **S**, find the count of its interesting subsequences.

Input specification

The first line of the input file contains an integer **T** specifying the number of test cases. Each test case is preceded by a blank line.

The first line of each test case contains the length of a sequence **N**. The second line contains **N** space-separated integers : the elements of the sequence.

Output specification

For each test case output a single line containing a single integer : the count of interesting subsequences of the given sentence.

Example

Input:

```
2

13
2 7 1 8 2 8 1 8 2 8 4 5 9
```

```
7
2 47 10047 47 1047 47 47
```

Output:

```
3
4
```

Note: the input file will not exceed 4MB.

Added by: Blue Mary
Date: 2007-12-01
Time limit: 3s-5s
Source limit: 50000B
Languages: All except: C99 strict
Resource: IPSC 2006

SPOJ Problem Set (classical)

2151. Digital Calculator

Problem code: CALCULAT

Dan likes playing with his pocket calculator during those long, boring math classes. Just now the teacher started to talk about the factorial function.

N factorial, denoted by **N!**, is the product of all the integers between 1 and **N**, inclusive. For example 6! = 6*5*4*3*2*1 = 720.

Dan took out his calculator out of his pocket to play around with this new function. Unfortunately his calculator quickly ran out of digits and only showed overflow errors. Soon, Dan realized that the factorial function grows very quickly. Still, he would like to know at least some of its digits.

Problem specification

Given three positive integers **N**(1<= **N** <=10^8), **K**(1<= **K** <=50), **L**(1<= **L** <=100), compute the first **K** digits and the last **L** digits of **N** factorial.

Input specification

The first line of the input file contains an integer **T** specifying the number of test cases. Each test case is preceded by a blank line.

Each test case consists of one line containing three positive integers **N**, **K** and **L** separated by single spaces. Neither **K** nor **L** will exceed the number of digits in **N!**.

Output specification

For each test case output one line containing two strings **A** and **B** separated by a single space. Here, **A** is the string composed of the first **K** digits of **N!** and **B** is the string composed of the last **L** digits of **N!**.

Example

Input:

```
3

6 2 1

10 3 2

8 5 5
```

Output:

```
72 0
362 00
40320 40320
```

Note : for all test cases whose N>=100, its K<=15.

Added by: Blue Mary
Date: 2007-12-01
Time limit: 1s-10s
Source limit:50000B
Languages: All except: C99 strict
Resource: IPSC 2006

SPOJ Problem Set (classical)

2152. Hilbert Curve

Problem code: FRACTAL

The Hilbert Mole is a small and very rare mole. The first and only specimen was found by David Hilbert at his backyard. This mole lives in a huge burrow under the ground, and the border of this burrow forms a Hilbert curve of n-th order (H_n).

[IMAGE]

Figure 1. Hilbert curves, order 1 to 4.

Hilbert curves can be defined as follows. H_1 is a unit square with open top side (figure 1(a)), H_n consists of four copies of H_{n-1}: bottom left and bottom right are copied without changes, top left is rotated 90^o counter-clockwise and top right is rotated 90^o clockwise. These small copies are connected by three segments of unit length (figure 1(b),(c),(d)).

[IMAGE]

Figure 2. Burrow, filled with water.

Trying to exterminate the mole, Mr. Hilbert fills the burrow with water (figure 2). But air inside the burrow prevents water from filling it entirely. In this problem we suppose that air and water are incompressible and cannot leak throw the borders of the burrow. Your task is to find the total area of the burrow, filled with water.

Note that water can flow over the obstacle only when its level is strictly higher. See examples on figure 3 for further clarification.

[IMAGE]

Figure 3. More examples of filled burrows.

Input

Multiple test cases. For each test case:

The first line of the input file contains two integer numbers: n and alpha - order of Hilbert curve and slope angle of surface in degrees ($1 <= n <= 12$, $0 <= alpha < 90$).

Input terminates by EOF.

Output

For each test case:

The first line of the output file must contain a single real number - the total area of the burrow, filled with water. The relative error of the answer must not exceed 10^{-6}.

Example

Input:

```
5 30
3 45
4 10
3 0
```

Output:

```
190.803847577293
15.5
91.573591766702
26.0
```

Added by:	Blue Mary
Date:	2007-12-01
Time limit:	17s
Source limit:	50000B
Languages:	All except: C99 strict
Resource:	ACM/ICPC Northeastern Europe Programming Contest(Northern Subregional) 2008, with description modified

SPOJ Problem Set (classical)

2153. Internet is Faulty

Problem code: IMATCH

David has a problem. He wants to transfer a big file through the internet from his home computer to his computer at work. The size of his file is S kilobytes.

The internet consists of N computers, numbered from 1 to N. David's home computer has the number 1 and his computer at work has the number 2. Some pairs of computers are connected by different types of links (such as network cables or wi-fi). Some of these links (e.g., a satellite dish) may be unidirectional, thus for simplicity we will assume that all links are unidirectional.

The data are sent across the network in packets, each packet contains exactly one kilobyte of data. For each link David knows how faulty it is, i.e. the probability that a network packet gets from one computer to the other through the link. We assume that all transfers are independent from each other. That is, regardless of whether the previous packet was transferred successfully or not, the probability that the next one will pass through remains the same.

Since the network is faulty and the work computer might be many links away from the home computer, the transfer of David's file along even the best route between the two computers might take too long. Luckily, David has an account on some of the machines in the network. He may use these machines as temporary storage, and thus shorten the time of the transfer.

The file transfer will consist of several steps. In each step, David selects a series of links starting with a computer that already has the file and ending with a computer David has an account on. Prior to the transfer, the file is split into S packets. Then the packets are sent one after another along the chosen route. The probability that a packet successfully arrives at the destination computer is the product of probabilities that it passes all the links. If the packet is lost it is resent immediately. Each attempt to send a packet, successful or unsuccessful, takes exactly one millisecond, regardless of the number of links on the route. After the entire file is transferred, David may start another transfer from the new machine, and so on.

Problem specification

You are given the number of computers N and the file size S. For each pair of computers u, v we know the probability p(u, v) that a network packet passes successfully through the direct link from computer u to v. (The value zero means that there is no direct link from u to v.) Finally, you are given a list of servers where David has an account.

Find a way how to send David's file so that the expected transfer time is minimized, and output this expected time in milliseconds. You may assume that the expected transfer time is less than 1 000 000 000 (1 billion) milliseconds.

Input specification

The first line of the input file contains an integer T specifying the number of test cases. Each test case is preceded by a blank line.

Each test case looks as follows: The first line contains a positive integer 2<= N <=200 giving the number of computers on the internet.

N lines follow. The i-th of these lines contains N integers, each of them between 0 and 100, inclusive. These integers give the probabilities p(i, 1), p(i, 2), ..., p(i, N) in percents.

The next line contains a non-negative integer M - the number of computers on which David has an account. The following line contains M integers - the numbers of these computers. This list will always contain the integers 1 and 2 (corresponding to David's home and work computer).

The last line contains an integer S - the size of David's file in kilobytes.

Output specification

For each test case output a single line with a real number - the expected time of the transfer in milliseconds when using the best possible strategy.

Each number in the output file should have sufficiently many decimal places. We recommend printing all results rounded to seven decimal places. Your output will be considered correct if each number has an absolute or relative error less than 10^{-6}.

Example

Input:

```
2

4
0 0 40 66
0 0 0 30
40 47 0 66
0 30 66 0
4
1 2 3 4
47

5
0 1 20 0 0
0 0 0 0 0
0 0 0 50 90
0 20 0 0 0
0 0 0 90 0
3
1 2 5
10
```

Output:

```
207.897153
111.111111
```

Hint

In the second case, we have four possible strategies.

The first one is to use the direct link from 1 to 2. This link is really really faulty, and the expected time for this solution is 1000 milliseconds. The second strategy is to transfer the file from computer 1 to computer 2 using the route 1-3-4-1. The probability that a packet passes this route is 20% * 50% * 20% = 2%, and thus the expected transfer time is 500 milliseconds. The third strategy is to use the route 1-3-5-4-2. Here the probability of successfully transfering a packet is 3.24% and the expected transfer time is roughly 308.6 milliseconds.

The optimal solution is to use the computer 5 as temporary storage. We will first transfer the file along the route 1-3-5, and then along the route 5-4-2. For each of the transfers the probability is 20% * 90% = 18%, and thus the expected time of each of the transfers is 55.5555 milliseconds.

A Note: to get round of the cheating submissions, the time limit is very strict for this problem.

Added by: Blue Mary
Date: 2007-12-01
Time limit: 1s-3s
Source limit:50000B
Languages: All except: C99 strict
Resource: IPSC 2007

SPOJ Problem Set (classical)

2154. Kruskal

Problem code: KRUSKAL

A three-headed monkey was on his (theirs?) peaceful way from his dorm to the university. He decided to use the subway. But as soon as he descended into the station, he was stopped by a strange geek with a flashlight, saying strange words...

"I am a servant of the Secret Group Order, wielder of the flame of Primes. Your limited knowledge of partial derivatives will not avail you, flame of Riemann! You shall not pass! You can't beat Kruskal in his game!"

The three-headed monkey shook his head. The left one. But there was no way out. If he wanted to get to the university in time, he had to play.

(Many others in his situation would use the distract-and-run tactics to get past the evil Kruskal into the subway. However, this was not possible in this case : nobody will turn around upon hearing *"Hey! Look behind you! A three-headed monkey!"* when he already sees the monkey in front of him...)

So, what was the game about? It is a two-player game. At the beginning there are **N** (not necessarily equal) heaps of matches. On each turn, a player may only remove matches from one heap only, and he has to remove between 1 and **K** matches, inclusive. A player wins if after his move the size of some heap is a prime number. The three-headed monkey moves first.

Problem specification

You will be given several starting positions. For each of them, determine whether the three-headed monkey can win this game. You may assume that Kruskal (the monkey's opponent) plays optimally.

Input specification

The first line of the input file contains an integer **T** specifying the number of test cases.

Each test case looks as follows: on the first line there are the two integers **N**(1<= **N** <=200) and **K**(1<= **K** <=100), separated by a single space. **N** lines follow, one for each heap of matches. The **i**-th of these lines contains a single integer $\mathbf{a_i}$ (3<= $\mathbf{a_i}$ < 2^{32}) giving the number of matches on the **i**-th heap.

Output specification

For each test case output one line. If the monkey can win the game, output the string "YES", otherwise output the string "NO".

Example

Input:

```
2

3 3
48
15
4

2 3
51
51
```

Output:

```
YES
NO
```

Note: a somewhat hard test has been removed.

Added by: Blue Mary
Date: 2007-12-01
Time limit: 3s-9s
Source limit: 50000B
Languages: All except: C99 strict
Resource: IPSC 2006

SPOJ Problem Set (classical)

2157. Anti-Blot System

Problem code: ABSYS

Jimmy is a hard-working pupil in his second year at primary school. Recently he decided to convert all his notes into an electronic version. Sadly, he found that his math notes were full of ink blots.

He scanned the notes and sent them through his own OCR package (yes, he coded it all by himself at the age of 8). The OCR package replaced all ink blots by the string "`machula`".

Problem specification

You are given Jimmy's notes, processed by the OCR. They contain simple math exercises, which were used to practice addition on positive integers. Your task is to recover the damaged part of the notes.

Input specification

The first line of the input file contains an integer **T** specifying the number of test cases. Each test case is preceded by a blank line.

Each test case consists of exactly one line. The line represents an equation of the form "`number + number = number`", where each `number` is a positive integer. One part of the equation will be replaced by the string "`machula`". The string always covers a contiguous non-empty sequence of digits, possibly even an entire number. You may assume that for each equation in the input there will be exactly one way to fill in the missing digits.

Output specification

For each test case, the output shall contain one line of the form "`number + number = number`". The line must represent the equation from that test case with all missing digits filled in.

Example

Input:
>

```
3

23 + 47 = machula

3247 + 5machula2 = 3749

machula13 + 75425 = 77038
<pre>
```

Output:>

```
23 + 47 = 70
3247 + 502 = 3749
1613 + 75425 = 77038
<pre>

Note: int in C++/C/Java or longint in Pascal is enough.
```

Added by: Blue Mary
Date: 2007-12-01
Time limit: 1s
Source limit: 50000B
Languages: All except: C99 strict
Resource: IPSC 2007

SPOJ Problem Set (classical)

2159. Delicious Cake

Problem code: CAKE3

Lenka likes to bake cakes since her childhood, when she has learned to bake from her mom. She soon became a cake expert able to bake chocolate cakes, apple pies, muffins, cookies, cheese cakes, tortes and many other cakes.

Recently, she has started her studies of math at Comenius University in Bratislava. In the first year she is taking combinatorics class. Today she is studying for the final exam. Since the brain needs a lot of sugar to study math, she has baked, just for herself, her favorite, very delicious, strawberry cake.

The cake, still hot, is lying on an **N**×**M** inch sheet pan. Hungrily waiting for the cake to cool off Lenka came up with an interesting combinatorial question: How many different possibilities to cut the cake are there so that every connected piece consists of some number of 1×1 inch unit squares?

Problem specification

The cake can be viewed as a grid consisting of **N**×**M** unit squares. We are allowed to cut the cake along the grid lines. As a result the cake splits into several connected pieces. (Two unit squares remain connected if they share a side which was not cut.) How many different ways are there to cut the cake? We consider two cuttings of the cake to be the same if the resulting connected pieces of both cuttings have the same shape and are at the same positions within the cake. In other words, we are only counting those cuttings where no cut leads between two unit squares that are in the same connected piece.

The following picture ilustrates all the 12 different possible ways how to cut a 2×2 inch cake: 12 possible ways of cutting a 2×2 cake
Note that cutting, for example, as on following picture
Not a good way of cutting
is the same as not cutting at all.

Input specification

The first line of the input file contains an integer **T** specifying the number of test cases. Each test case is preceded by a blank line.

Each test case consists of a single line with two positive integers **N** and **M** - dimensions of the cake.

Output specification

For each test case output a line with a single positive integer - the number of different possibilities how to cut the cake.

Example

Input:

>

```
2

1 2

2 2
<pre>
```

Output:>

```
2
12
<pre>
```

Note

For all the test cases, *min*(**N**,**M**)<=5, *max*(**N**,**M**)<=130.

Blue Mary's Note: The data has been enhanced on Feb.28, 2008 to avoid precalculated tables. Sorry to some users.

Added by: Blue Mary
Date: 2007-12-01
Time limit: 8s
Source limit:50000B
Languages: All except: C99 strict
Resource: IPSC 2007

SPOJ Problem Set (classical)

2160. Here-There

Problem code: HERE

Do you know the game Here-There? I presume you don't. It's a virtual board game, so you should first learn how this virtual board looks like.

The process of making the board is remarkably simple. You start by taking a square with side of length 3^N, divide it into nine smaller squares of equal size and remove the central one. Then, you repeat the same divide-and-remove-the-centre process with each of the eight smaller squares over and over (N times in total), until you are left with a grid that consists of many little squares with side length 1 - and of many holes. By the way, the number N is called the degree of the board.

The game itself consists of two steps. First, your opponent chooses two squares on the board, one of them will be "Here" and the other one "There". Your task is to estimate the least number of steps you have to take if you started Here and wanted to get to There. One step consists of moving to another square, which has a common side with the one you're standing on. Obviously, you cannot move over the removed parts of the board. If you guess the number of steps correctly, you get a point.

You would really like to become a master of this game, so you have written down the sizes of the boards and the positions of the Here and There squares from several games in the past. Now, you'd like to find the exact number of steps you need to take to get from Here to There on each of the boards. Each square is described by two numbers between 1 and 3^N, the first of them denoting the column and the second one the row the square is in. The square in the upper left corner of the board has coordinates (1, 1), as you can see on the picture below.

[IMAGE]

You can see one of the shortest paths between squares (1, 1) and (4, 8) on the picture, consisting of 10 steps.

[IMAGE]

Problem specification

You are given several boards and pairs of squares on them and your task is to find the steps-distances between the squares in each pair.

Input specification

The first line of the input file contains an integer T specifying the number of test cases. Each test case is preceded by a blank line.

Each test case consists of five integers D(1<= D <=39), Hc, Tc, Hr, Tr, specifying the degree of the board D, the coordinates of Here (Hc, Hr) and the coordinates of There (Tc, Tr).

Output specification

For each of the test cases, output a single line with one integer - the steps-distance between Here and There.

Example

Input:

```
2

1 1 2 2 1

2 1 1 4 8
```

Output:

```
2
10
```

Added by:	Blue Mary
Date:	2007-12-01
Time limit:	3s
Source limit:	50000B
Languages:	All except: C99 strict
Resource:	IPSC 2007

SPOJ Problem Set (classical)

2161. Pixel Shuffle

Problem code: JPIX

[IMAGE]

Shuffling the pixels in a bitmap image sometimes yields random looking images. However, by repeating the shuffling enough times, one finally recovers the original images. This should be no surprise, since "shuffling" means applying a one-to-one mapping (or permutation) over the cells of the image, which come in finite number.

Your program should read a number n , and a series of elementary transformations that define a "shuffling" [IMAGE] of n * n images. Then, your program should compute the minimal number m (m > 0) , such that m applications of [IMAGE] always yield the original n * n image.

For instance if [IMAGE] is counter-clockwise 90^o rotation then m = 4.

[IMAGE]

Input

Test cases are given one after another, and a single 0 denotes the end of the input. For each test case:

Input is made of two lines, the first line is number n ($2 <= n <= 2^{10}$, n even). The number n is the size of images, one image is represented internally by a n * n pixel matrix ($a^j{}_i$) , where i is the row number and j is the column number. The pixel at the upper left corner is at row 0 and column 0.

The second line is a non-empty list of at most 32 words, separated by spaces. Valid words are the keywords **id**, **rot**, **sym**, **bhsym**, **bvsym**, **div** and **mix**, or a keyword followed by **-**. Each keyword **key** designates an elementary transform (as defined by Figure 1), and **key-** designates the inverse of transform **key**. For instance, **rot-** is the inverse of counter-clockwise 90^o rotation, that is clockwise 90^o rotation. Finally, the list $k_1, k_2, ..., k_p$ designates the compound transform [IMAGE] $= k_1 o k_2 o ... o k_p$. For instance, "bvsym rot-" is the transform that first performs clockwise 90o rotation and then vertical symmetry on the lower half of the image.

[IMAGE]

Figure 1: Transformations of image ($a^j{}_i$) into image ($b^j{}_i$)

[IMAGE]

Output

For each test case:

Your program should output a single line whose contents is the minimal number m (m > 0) such that [IMAGE] is the identity. You may assume that, for all test input, you have $m < 2^{31}$.

Example

Input:

```
256
rot- div rot div
256
bvsym div mix
0
```

Output:

```
8
63457
```

Added by: Blue Mary
Date: 2007-12-01
Time limit: 13s
Source limit: 50000B
Languages: All except: C99 strict
Resource: ACM Central European Programming Contest, Budapest 2005

SPOJ Problem Set (classical)

2162. Towers of Powers

Problem code: TOWER

One of the many problems in computer-generated graphics is realistically modeling the "orderly randomness" of things like mountain ranges and city skylines. A new student intern at a graphics company had an idea - use fluctuations in number representations to model height. In this problem you will compute several such number representations and show the "skylines" they produce.

Let **n** be any positive integer, and let **b** be an integer greater than or equal to 2. The *complete base - **b** expansion of **n*** is obtained as follows. First write the usual base - **b** expansion of **n**, which is just a sum of powers of **b**, each multiplied by a coefficient between 1 and **b** - 1, omitting terms with zero coefficients. For example, if **n** = 20000 and **b** = 3, the base - 3 expansion of 20000 is given by

$20000 = 3^9 + 3^5 + 2\times3^3 + 2\times3^2 + 2$

To obtain the complete base - **b** expansion, we apply the same procedure to the exponents until all numbers are represented in base **b**. For **n** = 20000 and **b** = 3 we would have

$20000 = 3^{3^2} + 3^{3+2} + 2\times3^3 + 2\times3^2 + 2$

As another example, consider **n** = 16647 and **b** = 2. The resulting expansion is

$16647 = 2^{2^{2+1}+2^2+2} + 2^{2^{2+1}} + 2^2 + 2 + 1$

The rising and falling heights of the numbers form the number's "skyline".

For each pair of integers **n** and **b** in the input, display the complete base - **b** representation of **n**. Your display should use multiple output lines for different exponent heights. The display must begin with ***n*** = , followed by the expansion. Answers should use an asterisk as the multiplication symbol between coefficients and powers of **b**. Zero terms must not be printed, and unnecessary coefficients and exponents must not be shown (for example, display 1 instead of b^0, b^2 instead of $1*b^2$ and b instead of b^1). To assist in accurately viewing the skyline of the number, the display must show one character (either a digit, +, or *) per column of the multi-line display; there must be no unnecessary spaces. The correct format is illustrated in the sample output shown below.

Answers must be displayed using no more than 80 columns. Expansions requiring more than 80 columns must be split between terms, into two or more sets of display lines to show the remaining portion of the expansion. The second and following parts of the answer must begin in the same column as the first part of the answer and should contain the same number of (possibly blank) lines. The split may only occur between terms of the number itself (the bottom line), not between terms in an exponent. See the sample output for an example. Note that each set of display lines starts with a blank line.

Input

Input is a sequence of pairs of integers, **n** and **b**, followed by a pair of zeroes. Each value for **n** will be positive, and each value for **b** will be greater than or equal to 2. All values will fit into 64 bits unsigned integers (the maximum is therefore 18446744073709551615).

Output

For each input pair print the complete base - **b** expansion of **n** as described above. Print a line containing

```
n in complete base b:
```

preceding each expansion. Separate the output for consecutive pairs by a line of exactly 80 hyphens. All coefficients, bases, and exponents are to be displayed as standard base 10 integers.

Example

Input:

```
20000 3
16647 2
1000 12
85026244 3
0 0
```

Output:

```
20000 in complete base 3:

          2
         3   3+2    3    2
20000 = 3  +3   +2*3 +2*3 +2
--------------------------------------------------------------------------------
16647 in complete base 2:

          2+1  2     2+1
         2   +2 +2  2     2
16647 = 2         +2    +2 +2+1
--------------------------------------------------------------------------------
1000 in complete base 12:

           2
1000 = 6*12 +11*12+4
--------------------------------------------------------------------------------
85026244 in complete base 3:

             2           2          2          2          2       2       2
            3 +2*3+1    3 +2*3    3 +3+2    3 +3+1    3 +2    3 +1    3
85026244 = 3        +2*3      +2*3      +2*3      +2*3    +2*3    +2*3


               2*3+2  2*3+1    3
           +2*3     +3     +2*3 +3+1
```

Note

The number of test cases will be no more than 100. The judge of this problem compares your output and the expected output, an extra whitespace will cause Wrong Answer.

Added by: Blue Mary
Date: 2007-12-01
Time limit: 1s
Source limit: 50000B
Languages: All except: C99 strict
Resource: ACM ICPC World Final 1998

SPOJ Problem Set (classical)

2171. Ambiguous Codes

Problem code: AMCODES

An extensive area of research in computer science is the field of communications. With computer networks being part of everyday life of many people, the development of ways for making networks faster, more reliable and secure is constantly needed. This practical need motivates an extensive research activity in the theory behind communications.

The very first thing needed to establish any kind of communication is a common code. A code is a way of changing the form of a piece of information into some other form, in general to make it possible to convey that piece of information from one place to another. Flag codes used by boats and the Morse code used in telegraphy are examples of codes for translating letters into different forms to enable communication over different media.

More formally, a code is a set of strings composed of symbols from one alphabet. Each string defined in the code is called a code word. A message is then composed concatenating a set of code words to convey the information needed. For example, in Morse code the alphabet is composed of the symbols hyphen and dot; letter "S" is represented by the code word "...", letter "O" is represented by the code word "---", and therefore the distress message "SOS" in Morse code is "...---...".

Codes for communication can have many desirable and undesirable properties such as ambiguity, entropy, redundancy, and many more. In this problem we will focus on ambiguity as a key property.

A code is ambiguous when there exists a message using that code that can be partitioned into different sequences of code words. In other words, in an ambiguous code a message may have more than one meaning. For example, consider the binary alphabet, composed of symbols {0,1}. For the code composed of the words {10, 01, 101} the message 10101 can be understood as 10-101 or 101-01 and therefore the code is ambiguous. On the other hand, for the code composed of the words {01, 10, 011} no ambiguous message exists and therefore the code is unambiguous.

As a part of the computer science community, you are required to develop a tester that checks if codes are ambiguous. In case a code is indeed ambiguous, you are also required to report the length (i.e. the number of symbols) of the shortest ambiguous message for that code.

Input

Each test case will consist on several lines. In all test cases the alphabet will be the set of hexadecimal digits (decimal digits plus the uppercase letters "A" to "F"). The first line of a test case will contain an integer N (1 <= N <= 100), the number of code words in the code. Each of the next N lines describes a code word and contains a different and non-empty string of at most 50 hexadecimal digits.

Input is terminated by N = 0.

Output

For each test case, output a single line with the length of the shortest ambiguous message for the provided code or -1 if the code is unambiguous.

Example

Input:

```
3
10
01
101
3
AB
BA
ABB
0
```

Output:

```
5
-1
```

Added by: Camilo Andrés Varela León
Date: 2007-12-02
Time limit: 3s
Source limit: 50000B
Languages: All
Resource: The 2007 ACM South American Programming Contest

SPOJ Problem Set (classical)

2175. Emoticons

Problem code: EMOTICON

Emoticons are used in chat and e-mail conversations to try to express the emotions that printed words cannot. This may seem like a nice feature for many, but a lot of people find it really annoying and wants to get rid of emoticons.

George is one of those people. He hates emoticons so bad, that he is preparing a plan to remove all emoticons from all e-mails in the world. Since you share his visionary plans, you are preparing a special program to help him.

Your program will receive the list of emoticons to proscribe. Each emoticon will be a string of characters not including any whitespace. You will also receive several lines of text. What you need to do is change some characters of the text into spaces to ensure no emoticon is left on the text. For an emoticon to be considered to appear in the text it has to appear in a single line and be made of consecutive characters.

To help George's plan remain secret as long as possible, you need to do your job with the minimum possible amount of character changes.

Input

The input file contains several test cases. Each test case consists of several lines. The first line of each test case will contain two integers separated by a single space: N, the number of emoticons to proscribe, and M, the number of lines the text has. The next N lines contain one emoticon each, a non-empty string of at most 15 characters. Each of the last M lines of the test case contains a line of text of at most 80 characters. You can assume 1 <= N,M <= 100.

Valid input characters for emoticons are uppercase and lowercase letters, digits and the symbols "!?.,:;-_'#$%&/=*+(){}[]" (quotes for clarity). Each line of the text may contain the same characters with the addition of the space character.

The input is terminated by N = M = 0.

Output

For each test case, output exactly one line containing a single integer that indicates the minimum number of changes you need to make to the entire text to ensure no emoticon on the list appears in it.

Example

```
Input:
4 6
:-)
:-(
(-:
)-:
```

```
Hello uncle John! :-) :-D
I am sad or happy? (-:-(?
I feel so happy, my head spins
(-:-)(-:-)(-:-)(-:-) :-) (-: :-)
but then sadness comes :-(
Loves you, Joanna :-)))))
3 1
:)
):
))
:):)):)):)):(:((:(((:):)
0 0
```

Output:

```
11
8
```

Added by: Camilo Andrés Varela León
Date: 2007-12-02
Time limit: 4s
Source limit: 50000B
Languages: All
Resource: The 2007 ACM South American Programming Contest

SPOJ Problem Set (classical)

2185. Musical Optimization

Problem code: MUSIC

Bessie the cow used to write musical melody. A musical melody is represented as a sequence of N (1 <= N <= 100,000) notes numbered 1..N. Note i is represented by the integer A_i (-10,000 <= A_i <= 10,000).

To Bessie's cow-like mind, a musical melody is called 'perfect' if and only if the sum of all the notes in any of its consecutive subsequences is strictly positive.

For a given musical melody, Bessie wants to make it perfect, but she wants to change the melody as little as possible.

Thus, to perfect the melody, she repeatedly chooses a consecutive subsequence of the melody, [x, y] (1 < x <= y < N), whose sum S is negative. Then she adds S to both A_{x-1} and A_{y+1}, while subtracting S from both A_x and A_y. (It is possible to subtract from the same note twice if x = y.)

Given a musical melody, compute the minimum number of steps to make the melody perfect.

Input

* Line 1: The single integer N.

* Lines 2..N+1: Line i+1 contains the single integer A_i.

Output

* Line 1: A single integer that represents the minimum number of steps needed to make the given musical melody perfect. If there are no solutions, output -1 instead.

Example

Input:
```
5
13
-3
-4
-5
62
```

Output:
```
2
```

Explanation

There is a musical melody with length of 5. The notes are (13, -3, -4, -5, 62).

First, we choose the range [2, 4]; its sum is (-3) + (-4) + (-5) = -12. After the first step, the melody becomes (1, 9, -4, 7, 50). Second, we choose the range [3, 3], whose sum is -4, and the melody after the second step becomes (1, 5, 4, 3, 50). The melody is perfect now.

Warning: large input/output data, be careful with certain languages

---

Added by: Blue Mary
Date: 2007-12-03
Time limit: 1s
Source limit: 50000B
Languages: All except: C99 strict
Resource: Lei HUANG [g201513] , used in USACO

SPOJ Problem Set (classical)

2189. Making Pairs

Problem code: MKPAIRS

The 2*N (3 <= N <= 1,000) cows have assembled the Bovine Accordion and Banjo Orchestra! They possess various levels of skill on their respective instruments: accordionist i has an associated talent level A_i (0 <= A_i <= 1,000); banjoist j has an associated talent level B_j (0 <= B_j <= 1,000).

The combined 'awesomeness' of a pairing between cows with talents A_i and B_j is directly proportional to the talents of each cow in the pair so a concert with those two cows will earn FJ precisely A_i * B_j dollars in "charitable donations". FJ wishes to maximize the sum of all revenue obtained by his cows by pairing them up in the most profitable way.

Unfortunately, FJ's accordionists are a bit stuck up and stubborn. If accordionist i is paired with banjoist j, then accordionists i+1..N refuse to be paired with banjoists 1..j-1. This creates restrictions on which pairs FJ can form. FJ thus realizes that in order to maximize his profits, he may have to leave some cows unpaired.

To make matters worse, when one or more of the musicians is skipped, they will be greatly upset at their wasted talent and will engage in massive binge drinking to wash away their sorrows.

After all pairings are made, a list is constructed of the groups of each of the consecutive skipped musicians (of either instrument). Every group of one or more consecutive skipped cows will gather together to consume kegs of ice cold orange soda in an amount proportional to the square of the sum of their wasted talent.

Specifically, FJ has calculated that if the x-th to y-th accordionists are skipped, they will consume precisely $(A_x + A_{x+1} + A_{x+2} + ... + A_y)^2$ dollars worth of orange soda in the process of drinking themselves into oblivion. An identical relationship holds for the banjoists. FJ realizes that he'll end up getting stuck with the bill for his cows' drinking, and thus takes this into account when choosing which pairings to make.

Find the maximum amount of total profit that FJ can earn after the contributions are collected and the orange soda is paid for.

Input

* Line 1: A single integer: N

* Lines 2..N+1: Line i+1 contains the single integer: A_i

* Lines N+2..2*N+1: Line i+N+1 contains the single integer: B_i

Output

* Line 1: A single integer that represents the maximum amount of cash that FJ can earn.

Example

Input:
```
3
1
1
5
5
1
1
```

Output:
```
17
```

Explanation

There are 6 cows: 3 accordionists and 3 banjoists. The accordionists have talent levels (1, 1, 5), and the banjoists have talent levels (5, 1, 1).

FJ pairs accordionist 3 with banjoist 1 to get earn A_3 * B_1 = 5 * 5 = 25 in profit. He loses a total of $(1 + 1)^2 + (1 + 1)^2 = 8$ dollars due to the cost of soda for his remaining cows. Thus his final (net) profit is 25 - 8 = 17.

Time limit has been doubled on Aug.8, 2008, enjoy :)

Added by: Blue Mary
Date: 2007-12-04
Time limit: 2s
Source limit:50000B
Languages: All except: C99 strict
Resource: Lei HUANG [g201513] , used in USACO CHN07

SPOJ Problem Set (classical)

2202. Tan and His Interesting Game

Problem code: TAN1

Background

Tan always creates some interesting and strange games to kill time,and the Pick-Number Game on Tree is his favorite one.He got the idea from his another game(Pick-Number Game on sequence): there is an integer sequence, he picks a number from the head or the tail of the sequence each turn.When the sequence gets empty,he gets another sequence A,in which A[i] is the *i*-th integer he picks, then he calculates:

$S=A[0]*5^0+A[1]*5^1+...+A[n-1]*5^{n-1}$, while n is the length of the sequence.If S modudo 8 equals to 3,he wins, otherwise he loses(Tan is such a strange person that he likes games with strange rules).

Tan got tired of generating sequence randomly before playing a game, and he changed the rule to avoid it.This time he plays the game on trees. He generates a big tree. Every time he wants to play, he chooses two nodes (A,B) randomly and he finds the path connected A,B (including A,B).In this way he gets a sequence and he can play games.He calls this game "Game(A,B)".He can play many times on a big tree without generating a new one.If he can win in Game(A,B),he says that Game (A,B) is a good game,otherwise Game(A,B) is a bad game.

If a game is a bad game,he can never win,so he has to find a way to identify if a game is bad or good.

He played this game for a long time,and he thought he found a great law: if Game(A,B) is a good game and Game(B,C) is a good game,then Game(A,C) is a good game.And if Game(A,B) is a bad game and Game(B,C) is a bad game,the (A,C) is a bad game.But soon he found it was wrong,but he wanted to know in how many cases it is right.

P.S:"Tan" in Chinese means funny and droll. And Mr.Tan in the story is a real person.

Task

The input data describes a tree with integer numbers on each of its nodes.You should count the number of triple (A,B,C) (A,B,C are distinct nodes) that (A,B),(B,C),(A,C) are all good games or all bad games((A,B,C) and (B,C,A) are supposed to be counted once).

Input

The first line of the test data is the number of test case t,then t test case follow.

For each test case:

The first line contains a single integer M,the number of nodes in the tree(M<=100000).

M lines follow, each contains two integers F_i and V_i. F_i is the father of node i (F_i=0 if node i is the root).V_i is the number on the node i.(0<=V_i<=40000)

Output

For each test case:

The first and only line contains a single integer S, which means there are S triples(A,B,C) that (A,B),(B,C),(A,C) are all good games or all bad games.

Example

Input:

```
1
3
0 3
1 5
1 7
```

Output:

```
0
```

Warning: large input/output data, be careful with certain languages

---

Added by: Blue Mary
Date: 2007-12-07
Time limit: 2s
Source limit:50000B
Languages: All except: C99 strict
Resource: description and test data by LoLitter; standard program by rendezvous

SPOJ Problem Set (classical)

2270. Balloons in a Box

Problem code: BALLOON

You must write a program that simulates placing spherical balloons into a rectangular box.

The simulation scenario is as follows. Imagine that you are given a rectangular box and a set of points. Each point represents a position where you might place a balloon. To place a balloon at a point, center it at the point and inflate the balloon until it touches a side of the box or a previously placed balloon. You may not use a point that is outside the box or inside a previously placed balloon. However, you may use the points in any order you like, and you need not use every point. Your objective is to place balloons in the box in an order that maximizes the total volume occupied by the balloons.

You are required to calculate the volume within the box that is not enclosed by the balloons.

Input

The input consists of several test cases. The first line of each test case contains a single integer n that indicates the number of points in the set (1<=n<=6). The second line contains three integers that represent the (x, y, z) integer coordinates of a corner of the box, and the third line contains the (x, y, z) integer coordinates of the opposite corner of the box. The next n lines of the test case contain three integers each, representing the (x, y, z) coordinates of the points in the set. The box has non-zero length in each dimension and its sides are parallel to the coordinate axes.

The input is terminated by the number zero on a line by itself.

Output

For each test case print one line of output consisting of the test case number followed by the volume of the box not occupied by balloons. Round the volume to the nearest integer. Follow the format in the sample output given below.

Place a blank line after the output of each test case.

Example

Input:

```
2
0 0 0
10 10 10
3 3 3
7 7 7
0
```

Output:

```
Box 1: 774
```

Added by:	Blue Mary
Date:	2008-01-03
Time limit:	1s
Source limit:	50000B
Languages:	All except: C99 strict
Resource:	ACM/ICPC World Final 2002 (unofficial testdata)

SPOJ Problem Set (classical)

2271. Undecodable Codes

Problem code: UCODES

Phil Oracle has a unique ability that makes him indispensable at the National Spying Agency. His colleagues can bring him any new binary code and he can tell them immediately whether the code is uniquely decodable or not. A *code* is the assignment of a unique sequence of characters (a *codeword*) to each character in an *alphabet*. A binary code is one in which the codewords contain only zeroes and ones. For example, here are two possible binary codes for the alphabet {**a,c,j,l,p,s,v**}.

[IMAGE]

The *encoding* of a string of characters from an alphabet (the *cleartext*) is the concatenation of the codewords corresponding to the characters of the cleartext, in order, from left to right. A code is *uniquely decodable* if the encoding of every possible cleartext using that code is unique. In the example above, Code 1 is uniquely decodable, but Code 2 is not. For example, the encodings of the cleartexts "**pascal**" and "**java**" are both **001010101010**. Even shorter encodings that are not uniquely decodable are **01** and **10**.

While the agency is very proud of Phil, he unfortunately gives only "yes" or "no" answers. Some members of the agency would prefer more tangible proof, especially in the case of codes that are not uniquely decodable. For this problem you will deal only with codes that are *not* uniquely decodable. For each of these codes you must determine the single encoding having the minimum length (measured in bits) that is ambiguous because it can result from encoding each of two or more different cleartexts. In the case of a tie, choose the encoding which comes first lexicographically.

Input

One or more codes are to be tested. The input for each code begins with an integer m, 1<=m<=20, on a line by itself, where m is the number of binary codewords in the code. This is followed by m lines each containing one binary codeword string, with optional leading and trailing whitespace. No codeword will contain more than 20 bits.

The input is terminated by the number zero on a line by itself.

Output

For each code, display the sequential code number (starting with 1), the length of the shortest encoding that is not uniquely decodable, and the shortest encoding itself, with ties broken as previously described. The encoding must be displayed with 20 bits on each line except the last, which may contain fewer than 20 bits. Place a blank line after the output for each code. Use the format shown in the samples below.

Example

Input:

```
3
0
01
10
5
0110
00
111
001100
110
5
1
001
0001
00000000000000000001
10000000000000000000
0
```

Output:

```
Code 1: 3 bits
010

Code 2: 9 bits
001100110

Code 3: 21 bits
10000000000000000000
1
```

Added by: Blue Mary
Date: 2008-01-03
Time limit: 5s
Source limit: 50000B
Languages: All except: C99 strict
Resource: ACM/ICPC World Final 2002 (unofficial testdata)

SPOJ Problem Set (classical)

2272. Crossing the Desert

Problem code: DESERT

In this problem, you will compute how much food you need to purchase for a trip across the desert on foot.

At your starting location, you can purchase food at the general store and you can collect an unlimited amount of free water. The desert may contain oases at various locations. At each oasis, you can collect as much water as you like and you can store food for later use, but you cannot purchase any additional food. You can also store food for later use at the starting location. You will be given the coordinates of the starting location, all the oases, and your destination in a two-dimensional coordinate system where the unit distance is one mile.

For each mile that you walk, you must consume one unit of food and one unit of water. Assume that these supplies are consumed continuously, so if you walk for a partial mile you will consume partial units of food and water. You are not able to walk at all unless you have supplies of both food and water. You must consume the supplies while you are walking, not while you are resting at an oasis. Of course, there is a limit to the total amount of food and water that you can carry. This limit is expressed as a carrying capacity in total units. At no time can the sum of the food units and the water units that you are carrying exceed this capacity.

You must decide how much food you need to purchase at the starting location in order to make it to the destination. You need not have any food or water left when you arrive at the destination. Since the general store sells food only in whole units and has only one million food units available, the amount of food you should buy will be an integer greater than zero and less than or equal to one million.

Input

The first line of input in each trial data set contains n (2 <= n <= 20), which is the total number of significant locations in the desert, followed by an integer that is your total carrying capacity in units of food and water. The next n lines contain pairs of integers that represent the coordinates of the n significant locations. The first significant location is the starting point, where your food supply must be purchased; the last significant location is the destination; and the intervening significant locations (if any) are oases. You need not visit any oasis unless you find it helpful in reaching your destination, and you need not visit the oases in any particular order.

The input is terminated by a pair of zeroes.

Output

For each trial, print the trial number followed by an integer that represents the number of units of food needed for your journey. Use the format shown in the example. If you cannot make it to the destination under the given conditions, print the trial number followed by the word "Impossible."

Place a blank line after the output of each test case.

Example

Input:

```
4 100
10 -20
-10 5
30 15
15 35
2 100
0 0
100 100
0 0
```

Output:

```
Trial 1: 136 units of food

Trial 2: Impossible
```

Added by: Blue Mary
Date: 2008-01-03
Time limit: 1s
Source limit: 50000B
Languages: All except: C99 strict
Resource: ACM/ICPC World Final 2002 (unofficial testdata)

SPOJ Problem Set (classical)

2273. Ferries

Problem code: FERRY

Millions of years ago massive fields of ice carved deep grooves in the mountains of Norway. The sea filled these grooves with water. The Norwegian people call them fjords. This landscape of mountains and water is beautiful, but it makes traveling difficult. The usual scheme is: drive some kilometers, wait for a ferry, cross a fjord with the ferry, drive some more kilometers, and so on until the destination has been reached. To reach a destination as early as possible, most people have the following strategy: drive as fast as allowed (the maximum speed is 80 km/h) to the next ferry, and wait until it goes. Repeat until the destination has been reached.

Since driving fast requires more fuel than driving slow, this strategy is both expensive and harmful to the environment. The new generation of cruise control systems is designed to help. Given the route you want to go, these systems will gather information about the ferries involved, calculate the earliest possible time of arrival at the final destination, and calculate a driving scheme that avoids driving faster than needed. The systems will calculate your road speed so that you board the next ferry the moment it leaves.

Given a route (a sequence of road-pieces and crossings with ferries), you must write a program to calculate the minimal time it takes to complete this route. Moreover, your program must find a driving scheme such that the maximal driving speed at any point during the trip is as small as possible.

Input

The input file contains one or more test cases. Each test case describes a route. A route consists of several sections, each section being either a piece of road or a crossing. The first line in the description contains a single number s ($s > 0$), which is the number of sections in the route. The next s lines contain the descriptions of the sections. Every line describing a section starts with two names: the place of departure and the place of arrival, followed by either the word ‘road’ or the word ‘ferry’ indicating what kind of section it is. If the section is a road, its length (a positive integer) is given in km. For example:

Dryna Solholmen road 32

Lines describing ferry sections have more information. Following the word "ferry", the duration of the ferry crossing, in minutes (a positive integer) is given. This is followed by the frequency f ($f > 0$) of the ferry, that is, the number of times the ferry departs in a single hour. The next f integers give the departure times of the ferry, in ascending order. For example:

Manhiller Fodnes ferry 20 2 15 35

The ferry travels from Manhiller to Fodnes in 20 minutes, and it leaves twice an hour (on 0h15, 0h35, 1h15, 1h35,...). The beginning of the entire trip always starts at a full hour. The sections in a route are consecutive, that is, if a section goes from A to B then the next section starts at B. Every route in the input can be traveled in no more than 10 hours.

The input is terminated by the number zero on a line by itself.

Output

Output for each test case is a single line containing three items. The first item is the test case number. The second is the total travel time for an optimal scheme in the form hh:mm:ss. The third item is the maximal road speed in an optimal scheme rounded to two digits to the right of the decimal point.

Place a blank line after the output of each test case.

Example

Input:

```
1
Bygd Bomvei road 7
2
Ferje Overfarten ferry 20 2 5 25
Overfarten Havneby ferry 30 3 10 30 50
5
Begynnelse Brygge road 30
Brygge Bestemmelse ferry 15 4 10 25 40 55
Bestemmelse Veiskillet road 20
Veiskillet Grusvei road 25
Grusvei Slutt ferry 50 1 10
0
```

Output:

```
Test Case 1: 00:05:15 80.00

Test Case 2: 01:00:00 0.00

Test Case 3: 03:00:00 45.00
```

Added by: Blue Mary
Date: 2008-01-03
Time limit: 20s
Source limit: 50000B
Languages: All except: C99 strict
Resource: ACM/ICPC World Final 2002 (unofficial testdata)

SPOJ Problem Set (classical)

2274. Island Hopping

Problem code: ISLHOP

The company Pacific Island Net (PIN) has identified several small island groups in the Pacific that do not have a fast internet connection. PIN plans to tap this potential market by offering internet service to the island inhabitants. Each groups of islands already has a deep-sea cable that connects the main island to the closest internet hub on the mainland (be it America, Australia or Asia). All that remains to be done is to connect the islands in a group to each other. You must write a program to help them determine a connection procedure.

[IMAGE]

For each island, you are given the position of its router and the number of island inhabitants. In the figure, the dark dots are the routers and the numbers are the numbers of inhabitants. PIN will build connections between pairs of routers such that every router has a path to the main island. PIN has decided to build the network such that the total amount of cable used is minimal. Under this restriction, there may be several optimal networks. However, it does not matter to PIN which of the optimal networks is built.

PIN is interested in the average time required for new customers to access the internet, based on the assumption that construction on all cable links in the network begins at the same time. Cable links can be constructed at a rate of one kilometer of cable per day. As a result, shorter cable links are completed before the longer links. An island will have internet access as soon as there is a path from the island to the main island along completed cable links. If m_i is the number of inhabitants of the ith island and t_i is the time when the island is connected to the internet, then the average connection time is:

[IMAGE]

Input

The input consists of several descriptions of groups of islands. The first line of each description contains a single positive integer n, the number of islands in the group (n <= 50). Each of the next n lines has three integers x_i, y_i, m_i, giving the position of the router (x_i, y_i) and number of inhabitants m_i ($m_i > 0$) of the islands. Coordinates are measured in kilometers. The first island in this sequence is the main island.

The input is terminated by the number zero on a line by itself.

Output

For each group of islands in the input, output the sequence number of the group and the average number of days until the inhabitants are connected to the internet. The number of days should have two digits to the right of the decimal point. Use the output format in the sample given below.

Place a blank line after the output of each test case.

Example

Input:

```
7
11 12 2500
14 17 1500
9 9 750
7 15 600
19 16 500
8 18 400
15 21 250
0
```

Output:

```
Island Group: 1 Average 3.20
```

Added by: Blue Mary
Date: 2008-01-03
Time limit: 1s
Source limit: 50000B
Languages: All except: C99 strict
Resource: ACM/ICPC World Final 2002 (unofficial testdata)

SPOJ Problem Set (classical)

2275. Toil for Oil

Problem code: OIL

Prospecting for new sources of oil has become a high-technology industry. With improved drilling technology it has become economically viable to seek out ever smaller and harder to reach deposits of oil. However, using exploratory drilling to locate these deposits is not cost-efficient, so researchers have developed methods to detect oil indirectly.

One such method to detect oil is sonar, which uses reflected sound waves to locate caves in underground rock formations. Determining how much oil can be contained in such a cave is a difficult problem.

[IMAGE]

In this problem, you will be given some cross-sections of underground caves, represented by polygons such as the ones shown in the figure. Some of the points bounding the polygon may be holes through which oil can seep out into the surrounding rock (represented by black circles in the figure). Given the polygonal shape of the cave and the positions of the holes, you must compute the maximum amount of oil that could be in the cave (shown as gray shaded areas in the figure). This amount is limited by the fact that, in any connected body of oil, the oil level can never be above a hole, since it would drain into the surrounding rock instead.

Input

The input contains several cave descriptions, each in the form of a polygon that specifies a cross-section of a cave. The first line of each description contains a single integer n, representing the number of points on the polygon (3 <= n <= 100).

Each of the following n lines contains three integers xi, yi, hi. The values (xi, yi) give the positions of the points on the boundary of the polygon in counterclockwise order. The polygon is simple, that is, it does not cross or touch itself. The value of hi is equal to 1 if the point is a hole through which oil can seep out, and 0 otherwise. The "upward" direction in each case is the positive y-axis.

The input is terminated by a zero on a line by itself.

Output

For each cave description, print its oil capacity. Approximate the oil capacity by the area within the given cross-section that may contain oil, rounded to the nearest integer.

Example

Input:

```
4
10 0 0
5 10 1
0 20 0
```

```
-10 0 0
11
0 6 0
1 5 1
6 0 0
10 4 0
8 6 0
6 4 0
4 6 0
8 10 0
10 8 0
12 10 0
8 14 1
0
```

Output:

```
150
27
```

Test data has been corrected on Feb.5, 2009.

Added by: Blue Mary
Date: 2008-01-03
Time limit: 1s
Source limit: 50000B
Languages: All except: C99 strict
Resource: ACM/ICPC World Final 2002 (unofficial testdata)

SPOJ Problem Set (classical)

2276. Partitions

Problem code: RECTNG2

A partition of a rectangle is either a whole rectangle or a subdivision the rectangle into either a upper part and a lower part or a left part and a right part, and each part is a partition of the corresponding rectangle. Figure 1 shows several examples of partitions.

[IMAGE]

Figure 2 shows three equal sized rectangles, partitioned into sub-rectangles. Partition B is obtained from partition A by partitioning two of the sub-rectangles of A. Generally, if a partition B is obtained from A by partitioning one or more of its sub-rectangles, we say that B is finer than A, or that A is coarser than B. This relation is partial: partition C is neither coarser nor finer than A or B.

[IMAGE]

Given two partitions D and E of the same rectangle, infinitely many partitions exist that are finer than both D and E. In Figure 3 both F and G are finer than D and E. Among the partitions that are finer than both D and E, a unique one exists that is coarsest. This partition is called the infimum of D and E. In Figure 3, partition F is the infimum of D and E. [IMAGE]

In Figure 4, both H and J are coarser than D and E. Here J is the finest partition that is coarser than D and E. Then J is the supremum of D and E. [IMAGE]

Write a program that, given two partitions of the same rectangle, finds the infimum and the supremum of these partitions.

Input

The input file contains one or more test cases. The first line of each test case gives the width w and height h of the rectangle (0 < w, h <= 20). In the next h+1 lines the two partitions are given, as in the sample. Each of these lines contains 4*w+3 characters. The first 2*w+1 of these belong to the first partition; the last 2*w+1 of these belong to the second partition. A space separates the two partitions. Horizontal lines are created using underscores '_', vertical lines using 'l'.

The input is terminated by a pair of zeroes.

Output

For every case in the input file the output contains a single line containing the case number (in the format shown in the sample), followed by the infimum and the supremum of the two partitions, using the same format as the input.

Place a blank line after the output of each test case.

Example

Input:

```
4 3
 _ _ _ _   _ _ _ _
|_ _ _ _| |_|_ _ _|
|   |   | |       |
|_ _|_ _| |_ _ _ _|
3 4
 _ _ _   _ _ _
| |   | | |   |
| |   | |_|_ _|
|_|_ _| |   | |
|_ _|_| |_ _|_|
0 0
```

Output:

```
Case 1:
 _ _ _ _   _ _ _ _
|_|_ _ _| |_ _ _ _|
|   |   | |       |
|_ _|_ _| |_ _ _ _|
Case 2:
 _ _ _   _ _ _
| |   | |     |
|_|_ _| |     |
|_|_|_| |     |
|_ _|_| |_ _ _|
```

Added by: Blue Mary
Date: 2008-01-03
Time limit: 1s
Source limit: 50000B
Languages: All except: C99 strict
Resource: ACM/ICPC World Final 2002 (unofficial testdata)

SPOJ Problem Set (classical)

2277. Silly Sort

Problem code: SSORT

Your younger brother has an assignment and needs some help. His teacher gave him a sequence of numbers to be sorted in ascending order. During the sorting process, the places of two numbers can be interchanged. Each interchange has a cost, which is the sum of the two numbers involved.

You must write a program that determines the minimal cost to sort the sequence of numbers.

Input

The input file contains several test cases. Each test case consists of two lines. The first line contains a single integer n (n>1), representing the number of items to be sorted. The second line contains n different integers (each positive and less than 1000), which are the numbers to be sorted.

The input is terminated by a zero on a line by itself.

Output

For each test case, the output is a single line containing the test case number and the minimal cost of sorting the numbers in the test case.

Place a blank line after the output of each test case.

Example

Input:

```
3
3 2 1
4
8 1 2 4
5
1 8 9 7 6
6
8 4 5 3 2 7
0
```

Output:

```
Case 1: 4

Case 2: 17

Case 3: 41

Case 4: 34
```

Added by:	Blue Mary
Date:	2008-01-03
Time limit:	1s
Source limit:	50000B
Languages:	All except: C99 strict
Resource:	ACM/ICPC World Final 2002 (unofficial testdata)

SPOJ Problem Set (classical)

2317. Bracket Sequence

Problem code: LEXBRAC

Correct Bracket Sequence (CBS) is a sequence that can be obtained through following rules:
1) An empty string is the CBS.
2) If A is a CBS, then B = (A) is also a CBS.
3) If A is a CBS, then B = [A] is also a CBS.
4) If A and B are CBS, then C = AB is also a CBS.
Length of the CBS is the number of brackets in it, and this number is always even.
Assume that '(' < ')' < '[' < ']'.
CBS a1a2 ... an is lexicografically smaller than the CBS b1b2 ... bn if and only if there exists an integer i, i <= n, so that aj=bj, for each j, 1 <= j < i and ai < bi.

Illustration

Enumerate all CBS length 4 in lexicographical order: (()), ()(), ()[], ([]), [()], [[]], [](), [][].

Task

Your task is to find k-th CBS with length n in lexicographical order

Input

Contains 2 integers n (2..250) and k (1..10^120)

Output

Print the k-th CBS with length n in lexicographical order

Example

Input:
```
4
3
```
Output:
```
()[]
```

Added by: Chinh Nguyen
Date: 2008-01-10
Time limit: 1s
Source limit:50000B
Languages: All

SPOJ Problem Set (classical)

2318. Overlapping Words

Problem code: WORDS

Consider the following game: you are given a word W that contains only lower-case letters ("a" to "z"). You are also given a set of words and you are allowed to do the following operation: choose a suffix of W and replace it with another word (you can do this only if that word contains the suffix as its prefix). For example, if the current words is "acmicpc" you can choose to replace the suffix "pc" with any word in the set you are given that starts with "pc", e.g. "pcaaargh" thus forming the word "acmicpcaaargh". You can then repeat this procedure as many times as you wish. You are free to choose any suffix of the current word as long as it matches a prefix of the word you are replacing it with. You can also choose not to do the operation at all. Each suffix you replace must have length at least 1 (you cannot replace the "null suffix" with a word).

You are also given an integer L, 1 <= L <= 5. Your task is to find how many different subwords of length exactly L can be produced using the following operation. A "subword" means a substring of length L.

Input

The first line of the input file contains two integers, N and L separated by space. N, 1 <= N <= 100 is the number of words in the set (including the initial word). The next N lines describe one word each. No other symbols except small latin letters, and the end-of-line symbol are found on these lines. Each word has length at least L and at most 128. The word you start with is the first word in this set.

Output

The only line of the output should contain one integer: the number of different subwords of length L that can be produced using the operation described above.

Example

Input:
```
2 5
acmicpc
pcaaaaaaargh
```

Output:
```
11
```

Added by: Jelani Nelson (Minilek)
Date: 2008-01-10
Time limit: 30s
Source limit:50000B
Languages: All
Resource: MIT 1st Team Contest 2007

SPOJ Problem Set (classical)

2319. Sequence

Problem code: BIGSEQ

You are given the sequence of all K-digit binary numbers: 0, 1,..., 2^K-1. You need to fully partition the sequence into M chunks. Each chunk must be a consecutive subsequence of the original sequence. Let S_i (1 <= i <= M) be the total number of 1's in all numbers in the ith chunk when written in binary, and let S be the maximum of all S_i, i.e. the maximum number of 1's in any chunk. Your goal is to minimize S.

Input

In the first line of input, two numbers, K and M (1 <= K <= 100, 1 <= M <= 100, M <= 2^K), are given, separated by a single space character.

Output

In one line of the output, write the minimum S that can be obtained by some split. Write it without leading zeros. The result is not guaranteed to fit in a 64-bit integer.

Example

Input:
```
3 4
```

Output:
```
4
```

Added by: Jelani Nelson (Minilek)
Date: 2008-01-10
Time limit: 30s
Source limit:50000B
Languages: All
Resource: MIT 1st Team Contest 2007

SPOJ Problem Set (classical)

2320. Manhattan

Problem code: DISTANCE

The L_1 distance of two d-dimensional points is the sum of absolute values of their coordinate differences (i.e. $S_{i=1}^{d} |x_i - y_i|$ for two points x,y). Given N points in the plane you must find the farthest pair of points under the L_1 distance metric and output their distance.

Input

The first line of the input is "N d" (2 <= N <= 100000, 1 <= d <= 6) signifying that there are N points in d-dimensional space. N lines then follow, where the ith line is a space-separated list of d numbers, the coordinates of the ith point. All given coordinates are integers that are at most 1000000 in absolute value, and all given points are distinct.

Output

Your output should consist of a single integer, the farthest distance between a pair of input points, followed by a newline.

Example

Input:
```
3 2
0 0
-5 0
1 1
```

Output:
```
7
```

Added by: Jelani Nelson (Minilek)
Date: 2008-01-10
Time limit: 1s-30s
Source limit:50000B
Languages: All
Resource: MIT 1st Team Contest 2007

SPOJ Problem Set (classical)

2321. Segments

Problem code: SEGMENTS

There are N horizontal line segments in the plane. The ith segment has some height h_i (which may be negative) and runs from $x = a_i$ to $x = b_i$ ($a_i < b_i$). Segments do not contain their endpoints. You must draw a set of vertical lines (note *lines* and not *line segments*) so that every given horizontal segment is intersected at least once and at most R times by vertical lines in such a way that R is minimized.

Input

The first line of the input is N (1 <= N <= 400), the number of horizontal line segments. N lines then follow, where the ith line is "a_i b_i h_i". Each of a_i,b_i,h_i are 32-bit signed integers. Horizontal segments may overlap.

Output

Your output should consist of a single integer, the smallest value of R that is achievable, followed by a newline.

Example

Input:
```
3
0 1 5
0 2 -2
1 2 7
```

Output:
```
2
```

Added by: Jelani Nelson (Minilek)
Date: 2008-01-10
Time limit: 10s
Source limit:50000B
Languages: All
Resource: MIT 1st Team Contest 2007

SPOJ Problem Set (main)

2322. Tree Game

Problem code: TREEGAME

A complete binary tree of depth h is given. You can asign the value 0 or 1 to each leaf. Its internal nodes compute their values based on the values of their children, which is 0 if both children have value 1, and 1 otherwise. You play a game with a computer. The computer can ask you for a value of any leaf, and you can tell him either 0 or 1. The computer wants to know the root value, and he will keep asking until he is absolutely sure what the root value is. Your goal is to make him ask as many questions as possible. It is known that you can make a computer ask for all leaves. For a given sequence of leaves determine a sequence of 0's and 1's as answers to those leaves such that at no point before asking the last leaf value can the computer be sure of the root value. You must answer each question optimally (i.e. you should not make use of the knowledge of what the computer's (i+1)st query will be when you answer the ith query).

Input

In the first line of input the number h is given (1 <= h <= 15). In the second line a space-separated list of 2^h numbers are given. They are a permutation of the numbers 1, 2, ..., 2^h, and they represent the order of asked leaves (leaves of a tree are indexed from left to right).

Output

On a single line write a space-separated sequence of 0's and 1's corresponding to the values of leaves in the given order of being asked. If there are multiple solutions, write any of them. Do not output a response for the last query.

Example

Input:
```
3
5 2 7 3 1 6 8 4
```

Output:
```
1 1 1 1 0 0 1
```

Added by: Jelani Nelson (Minilek)
Date: 2008-01-10
Time limit: 30s
Source limit:50000B
Languages: All
Resource: MIT 1st Team Contest 2007

SPOJ Problem Set (classical)

2323. Broken Compass

Problem code: COMPASS

A group of adventurers are on an expedition to reach the North Pole. The only instrument they have is a compass that points to the pole. However, the compass is broken and although its measurements are consistent, the direction it points to is some fixed but unknown angle away from true North. For example, if the error angle is 90 degrees, the compass will always point East at times when it should point North. This error angle may be zero, in which case the compass is not broken.

Input

To overcome this difficulty the group decided to take measurements at several different points and try to recover the location of the North pole from that. For simplicity, we assume that the region around the North pole is a plane in which a coordinate system is introduced. Each point is then described by a pair of real numbers (x,y), x,y (- [-200, 200]. One of those points (not necessarily (0,0)) is the location of the pole. The compass would always point to that location if it were not broken. Instead, it always points a fixed angle away from the true direction to the pole. In this problem we assume that the magnetic pole is the same as the geographic pole and that it is a point source.

Output

The first line of input specifies one integer number N, N < 10, the number of measurements taken. Each of the next N lines contains four space separated real numbers: the first two are the x and y coordinates of the point at which the measurement was taken, the third and fourth are (u_x, u_y), the direction of a unit vector indicating where the compass needle points. There should be only one line of output that should contain two real numbers separated by space: the x and y coordinates of the pole. Your output is considered correct if it is within an additive 0.01 of the correct answer. All measurements are consistent and it is always possible to determine the location of the North pole from them. None of the measurements is made on the pole.

Example

Input:

```
4
1.000000 0.000000 -0.000000 -1.000000
-1.000000 0.000000 0.000000 1.000000
0.000000 1.000000 1.000000 0.000000
0.000000 -1.000000 -1.000000 0.000000
```

Output:

```
0.000000 0.000000
```

Added by:	Jelani Nelson (Minilek)
Date:	2008-01-10
Time limit:	30s
Source limit:	50000B
Languages:	All
Resource:	MIT 1st Team Contest 2007

SPOJ Problem Set (main)

2324. Mario

Problem code: MARIOGAM

Mario lives in an *N x M* maze grid. In this maze there are coin boxes, monsters, pipe systems, and walls. Whenever Mario enters a cell containing a coin box, he jumps to hit the box and gets as many coins as there are in the box (coin boxes do not disappear or lose coins after being hit). When Mario enters a cell with a monster, he loses a life. Pipe systems are like teleporters: for each system there is exactly one exit with at least one (but possibly several) entrances leading to that exit. When Mario walks into the entrance to a pipe system he is teleported to that pipe system's unique exit. Walking into a pipe system's exit does nothing special. Finally, Mario cannot walk into walls.

Mario decides to play a game. In the beginning of the game he starts with three lives at some given position, and at each time step he looks at all neighboring cells (excluding walls) and chooses one neighboring cell uniformly at random to walk to (x neighbors y if x is directly above, below, to the left, or to the right of y). If Mario has no non-wall neighboring cells then he stays at his current location. The game is over when either Mario is out of lives or it is impossible for him to collect more coins. Help Mario figure out the expected number of coins he will earn in one play of the game.

Input

The first line of the input is "*N M*" (1 <= *N,M* <= 15), giving the dimensions of the maze. What follows are *N* lines, each of which are *M* characters in length. The ith line displays the contents of the cells in the ith row of the maze. Mario starts in the unique cell with an '$' (which, beside holding Mario, is otherwise an empty cell). Cells with monsters are designated with '!'. Cells with coin boxes are represented by a number between *0* and *9* (inclusive), which is the number of coins in that coin box. Each pipe system is associated with a distinct letter between 'a' and 'z' (inclusive). A pipe system's entrances are designated with lower case letters, and the unique exit for a given pipe system has the corresponding capitalized letter (e.g. a pipe system with entrances labeled 'c' has exactly one exit, and it is labeled 'C'). Every pipe system appearing in the maze is guaranteed to have exactly one exit and at least one entrance. The character '#' designates a wall, and '.' designates an empty cell that Mario can just walk through.

Output

If the expected number of coins Mario collects is infinite, output -1. Otherwise, output a single real number, the expected number of coins Mario collects before the game is over. Your answer should be accurate to within either an absolute or relative error of 10^{-6} of the actual answer. Your output should be followed by a newline.

Example

Input:

```
2 3
$1!
```

```
a.A
```

Output:

```
3.000000000
```

Added by:	Jelani Nelson (Minilek)
Date:	2008-01-10
Time limit:	30s
Source limit:	50000B
Languages:	All
Resource:	MIT 1st Team Contest 2007

SPOJ Problem Set (main)

2325. String Distance

Problem code: STRDIST

Let A = $a_1 a_2 ... a_k$ and B = $b_1 b_2 ... b_l$ be strings of lengths k and l, respectively. The string distance between A and B is defined in the following way (d[i,j] is the distance of substrings $a_1 ... a_i$ and $b_1 ... b_j$, where 0 <= i <= k and 0 <= j <= l -- i or j being 0 represents the empty substring). The definition for d[i, j] is d[0, 0] = 0 and for (i, j) != (0, 0) d[i, j] is the minimum of all that apply:

- d[i, j - 1] + 1, if j > 0
- d[i - 1, j] + 1, if i > 0
- d[i - 1, j - 1], if i > 0, j > 0, and $a_i = b_j$
- d[i - 1, j - 1] + 1, if i > 0, j > 0, and a_i != b_j
- d[i - 2, j - 2] + 1, if i >= 2, j >= 2, $a_i = b_{j-1}$, and $a_{i-1} = b_j$

The distance between A and B is equal to d[k,l].

For two given strings A and B, compute their distance knowing that it is not higher than 100.

Input

In the first line, k and l are given, giving the lengths of the strings A and B (1 <= k, l <= 10^5). In the second and third lines strings A and B, respectively, are given. A and B contain only lowercase letters of the English alphabet.

Output

In the first line, write one number, the distance between A and B, followed by a newline.

Example

Input:
```
8 8
computer
kmpjutre
```

Output:
```
4
```

Added by: Jelani Nelson (Minilek)
Date: 2008-01-10
Time limit: 30s
Source limit:50000B
Languages: All
Resource: MIT 1st Team Contest 2007

SPOJ Problem Set (classical)

2371. Another Longest Increasing Subsequence Problem

Problem code: LIS2

Given a sequence of **N** pairs of integers, find the length of the **longest increasing subsequence** of it.

An **increasing sequence** $A_1..A_n$ is a sequence such that for every $i < j, A_i < A_j$.

A **subsequence** of a sequence is a sequence that appears in the same relative order, but not necessarily contiguous.

A pair of integers (x_1, y_1) is less than (x_2, y_2) **iff** $x_1 < x_2$ and $y_1 < y_2$.

Input

The first line of input contains an integer **N** (2 <= **N** <= 100000).

The following **N** lines consist of **N** pairs of integers (x_i, y_i) (-10^9 <= x_i, y_i <= 10^9).

Output

The output contains an integer: the length of the longest increasing subsequence of the given sequence.

Example

Input:

```
8
1 3
3 2
1 1
4 5
6 3
9 9
8 7
7 6
```

Output:

```
3
```

Added by: Jin Bin
Date: 2008-01-20
Time limit: 2s
Source limit: 50000B
Languages: All except: C99 strict

SPOJ Problem Set (classical)

2412. Arranging Amplifiers

Problem code: ARRANGE

Scientists at the TIFR, Mumbai, are doing some cutting edge research on the Propagation of Signals. A young researcher comes up with a method of progressively amplifying signals, as they progress along a path. The method involves the placing of Amplifiers at regular distances along the line. Each amplifier is loaded with a number a(i), which is called its amplification factor. The method of amplification is simple: an amplifier which recieves a signal of strength X, and has Y loaded in it, results in a signal of strength Y^X [Y to the power X]. In course of his research, the young scientist tries to find out, that given a set of n amplifiers loaded with a(0), a(1), a(2), ..., a(n-1), which particular permutation of these amplifiers, when placed at successive nodes, with the initial node given a signal of strength 1, produces the strongest output signal.
this is better illustrated by the following example : 5 6 4
4^(5^(6^1)) is the strength of the strongest signal, which is generated by putting amplifier loaded with 6 in first place, 5 in second place and 4 in third place.
Given a list of integers specifying the set of amplifiers at hand, you must find out the order in which they must be placed, to get the highest signal strength. In case their exist multiple permutations with same output, you should print the one which has bigger amplifiers first.

Input

First line of input contains T, the number of test cases. For each test case first line contains a number ni, which is equal to the number of amplifiers available. Next line contains n integers, separated by spaces which denote the values with which the amplifiers are loaded.

Output

Output contains T lines, one for each test case. Each line contains ni integers, denoting the order in which the amplifiers should be kept such that the result is strongest.

Example

Input:
```
2
3
5 6 4
2
2 3
```

Output:
```
6 5 4
2 3
```

Constraints and Limits

T <= 20, Ni <= 10^5.
Each amplifier will be loaded with a positive integer p, 0 < p <= 10^9.
No two amplifier > 1 shall be loaded with the same integer.

Added by:	Ajay Somani
Date:	2008-02-04
Time limit:	5s
Source limit:	50000B
Languages:	All
Resource:	CodeCraft 08 Anshuman Singh

SPOJ Problem Set (classical)

2413. Building Beacons

Problem code: BUILD

A Millionaire Eccentric Mathematician purchases a small Island, to build his residence. In order to keep away Trespassers, he decides to Build some Beacons (Towers) on the perimeter of the Island. To build these Beacons, he visits the Mainland to purchase Stone.
Being obsessed with Prime Numbers, the stone blocks he orders are all Regular Tridecagons (13 Sides). However, their heights are variable. On delivery of the Blocks, he orders them to be flattened perfectly from the top and the bottom, so that the Tridecagonal shape is viewable from the Top (or Bottom), in such a way that the Height of each block measures to a power of 2. The stones, procured from various sources, are of Variable strength. He intends to build the Beacons by placing Stone Blocks one over the other. The heights of the Towers are specified by the Mathematician, and they are also Prime Numbers. He instructs the Architect to make sure that once the Beacons are complete, the strengths of the used stones add up to the Maximum Possible Number.
> The Architect, not knowing how the job can be done, or even whether it CAN be done, with such complicated restrictions, approaches you to find this out for him.

Input<h3>

First line of input contains a positive integer T, the number of test cases. The first line of each test case contains a positive integer K, equal to the total number of stones available. This line is followed by K lines, each containing a pair of positive integers X,Y. Here X denotes the height of that stone (here the height would be 2^X) and Y denotes its value. Followed is a line containing an integer N, the number of towers. Each of the next N lines contain a single positive number, which is the height H of the corresponding tower the mathematician wants to build.

Output

Output contains T lines, one for each test case, containing a number denoting the maximum sum total of strengths of rocks used to construct the Beacons if solution is possible, else the message "Plan Failed!".

Example

Input:

```
1
2
2 3
0 4
1
5
```

Output:

```
7
```

Constraints and Limits

T <=11, H < 10^281, 0 <= X <= 977, 0 < Y <= 977, 0 <= k <= 977 , N <= 97

Added by:	**Ajay Somani**
Date:	**2008-02-05**
Time limit:	**4s**
Source limit:	**50000B**
Languages:	**All**
Resource:	**CodeCraft 08, Problem Setter: Gaurav Agarwal,Anshuman Singh**

SPOJ Problem Set (classical)

2414. Calculate The Cost

Problem code: CCOST

In a small Village near the Himalayas, there is a rich Land-Owner, in Possession of a vast, rectangular tract of land. Unknown to him, a Major Oil Corporation has verified the existence of a vast Oil Resource beneath the land owned by him.
>
> The Oil Company sends a Man to negotiate the purchase of a rectangular field from within the landowner's land, with sides parallel to those of his area. The Landowner, valuing his land according to the trees growing in it and the area to be purchased, gives the company man a Map of his Land, marking the location of trees of different types, and a list of the worth of each type of tree.
>
To ensure the most economic purchase of land with the required dimensions, the Company Man provides you with the data in his possession, and alongwith that, a list of the land areas that he considers good by his judgement.
>
You must provide, for each land area that he has listed, the Sum Total of the values of the Trees that lie Within or On the Boundary of that land area.

Input

The first line of the input contains an integer T, which is the number of test cases. For each test case, the first line contains an integer n, equal to the number of trees in the area. This line is followed by n lines each containing 3 integers separated by spaces which are coordinate of the tree (x, y) and value of that tree. Following this is an integer R, equal to the number of proposols of land areas given by the Company Man. Next R lines contain 4 integers each (x1, y1, x2, y2) which are the coordinates of lower left (x1,y1) and upper right (x2, y2) corner of the rectangular area.

Output

For each test case, your program should output R lines containing the sum of values of the Trees which lie inside or on the corresponding rectangular plot. There should NOT BE any blank lines between output of different test cases.

Example

Input:

```
1
3
1 1 2
2 2 3
3 3 4
2
1 1 1 2
0 0 5 5
```

Output:

```
2
9
```

Constraints and Limits

T <= 10, n <= 10^5, r <= 50000, 0 <= x,y <= 10^7, value of any tree <= 10^4.
For any rectangular area 0 <= x1 <= x2 <= 10^7, 0 <= y1 <= y2 <= 10^7
Note 1: There can be more than one trees at the same point.
> Note 2:<b> The input data is large (about 15 MB), Be careful about your input output routines.
>.

Added by:<td>	**Ajay Somani**
Date:	**2008-02-05**
Time limit:	**8s**
Source limit:	**50000B**
Languages:	**All**
Resource:	**CodeCraft 08, Problem Setter: Ajay Somani,Anshuman Singh**

SPOJ Problem Set (classical)

2415. Kirchhof Law

Problem code: RESIST

Input

Multiple test cases. For each test case:

The first line contains integers N and M; N is a number of nodes in the circuit (2 < N <= 100), M is the number of resistors (0 <= M <= 300). Each of the next M lines consists of three integers A_i, B_i and R_i -- description of a resistor that has resistance R_i connecting the nodes A_i and B_i (1 <= A_i , B_i <= N; 1 <= R_i <= 100).

Input terminates by EOF.

Output

For each test case, output the total resistance between the nodes 1 and N rounded within two digits after a decimal points.

Example

Input:
```
4 5
1 2 15
2 4 5
1 3 10
3 4 10
2 3 1
```

Output:
```
9.40
```

Added by: Blue Mary
Date: 2008-02-05
Time limit: 1s
Source limit:50000B
Languages: All except: C99 strict
Resource: Timus Online Judge, test data by g201513

SPOJ Problem Set (classical)

2416. Distinct Subsequences

Problem code: DSUBSEQ

Given a string, count the number of distinct subsequences of it (including empty subsequence). For the uninformed, A subsequence of a string is a new string which is formed from the original string by deleting some of the characters without disturbing the relative positions of the remaining characters. >For example, "AGH" is a subsequence of "ABCDEFGH" while "AHG" is not.

Input<h3>

First line of input contains an integer T which is equal to the number of test cases. You are required to process all test cases. Each of next T lines contains a string s.

Output

Output consists of T lines. Ith line in the output corresponds to the number of distinct subsequences of ith input string. Since, this number could be very large, you need to output ans%1000000007 where ans is the number of distinct subsequences.

Example

Input:

```
3
AAA
ABCDEFG
CODECRAFT
```

Output:

```
4
128
496
```

Constraints and Limits

T <= 100, length(S) <= 100000
> All input strings shall contain only uppercase letters.

Added by:<td>	**Ajay Somani**
Date:	**2008-02-05**
Time limit:	**2s**
Source limit:	**50000B**
Languages:	**All**
Resource:	**CodeCraft 08, Problem Setter: Jin Bin**

SPOJ Problem Set (classical)

2417. Eliminate The Enemies

Problem code: ENEMY

In a modification of the popular game 'Pacman', the player has to move in a two-dimensional grid. Several cells of the grid are blocked. The player can start from any cell that is not blocked and can move in any of the directions, i.e. north, west,south or east, provided that the cells are unblocked.
>
> As soon as the player passes a cell, an enemy is generated in that cell, making it impossible for the player to pass through that cell again. Thus, the player can pass through any given cell only once. The player has to traverse all the unblocked cells in the grid in order to win .
>
> The player can begin at any free cell. Note that the same path with different starting points and even with the same starting point but with different paths of traversal is treated as different routes. The problem requires you to print the total number of all such possible routes.

Input

Each test case starts with a line containing two integers, m and n. Each of the next m lines contain a string of n characters describing the configuration of the grid. '*' denotes a blocked cell and '.' denotes unblocked cells. The input ends with a case having m = 0 and n = 0 and this case need not be processed.

Output

For each test case, print one line containing the total number of possible routes for the corresponding case. As this number can be quite large, you should print ans%1000000007 where ans is the required result.

Example

Input:

```
3 3
...
.*.
...
3 7
...*...
.*.*.*.
.......
3 3
***
*.*
***
0 0
```

Output:

```
16
8
1
```

Constraints and Limits

m <= 100, n <= 7, number of test cases <= 50

Added by: Ajay Somani
Date: 2008-02-05
Time limit: 6s
Source limit:50000B
Languages: All
Resource: CodeCraft 08, Problem Setter: Jin Bin

SPOJ Problem Set (classical)

2418. Flying Frogs

Problem code: FFROG

WiseFrog, The King of FrogLand is on his Deathbed. He has 2 Sons, SensibleFrog and SmartFrog. Both of them are "Infinitely Intelligent". To decide who will succeed him as King, he devises a Strategy Game for the two Sons-
>
> An Arena is constructed, in the form of a Rectangle, having m X n Square Areas. They are labelled as (i,j), starting from the Upper Left extremity of the Arena [i=0,1,2...n-1; j=0,1,2...m-1]. The Squares in the Arena are filled with Flying Frogs, in a Random Manner, such that there can be any number of Frogs in each square.
>
> Once the Arena is ready, the 2 Frog Princes begin to play the Game, which is played in the following manner: SensibleFrog, being the King’s favourite, starts the Game.
>
> Each Prince takes his turn alternately. In his turn, he is permitted a maximum of K moves, and a minumum of 1 move. A "Move" is defined as the Issuing of an Order to any Frog of his choice. The "order" consists of the Direction to jump in, and the Number of Squares to Cover (which should be positive), the directions of movement permitted being Up and Left Only. However, the Order must not be such that it causes the Frog to land outside the Arena. Being Flying Frogs, the frogs in the Arena can jump any distance without trouble.
>
> If there arrives a situation where the Prince having his turn does not have ANY move Possible (that is, ALL the Flying Frogs are already at the top-left most square of the arena), the other Prince is declared the Winner. Given all the Starting Conditions, your task is to find out who becomes the King of FrogLand.
>

Input<h3>

First line of input contains an integer T, equal to the number of test cases. Followed is the description of each of the T test cases and you are required to process all test cases. First line of each test case contains three integers m,n,k (in this order). Each of the next m lines contain n integers separated by spaces. Jth integer in ith line corresponds to the number of Flying Frogs in Square (i,j) in the arena.

Output

Output contains one line for each test case. You have to output "SensibleFrog Wins!." if SensibleFrog wins and "SmartFrog Wins!." otherwise.

Example

Input:

```
3
1 1 1
837465
2 2 1
0 0
0 1
2 2 2
0 0
0 2
```

Output:

```
SmartFrog Wins!.
SmartFrog Wins!.
SensibleFrog Wins!.
```

Constraints and Limits

T <= 20, 1 <= m,n <= 1000, m*n <= 10^5, 1 <= k <= 10^9 , 0 <= Number of Frogs in a square <= 10^100.
> The total input data in each of the input file doesn't exceed 5 MB.

Added by:<td>	**Ajay Somani**
Date:	**2008-02-05**
Time limit:	**4s**
Source limit:	**50000B**
Languages:	**All**
Resource:	**CodeCraft 08, Problem Setter: Ajay Somani**

SPOJ Problem Set (classical)

2419. G-Line Grid

Problem code: GLGRID

The 21st century introduces the multicores. As a result a research is going on in parallel Computing. With time the number of processor would grow very large. As of now, Professor Biloo at IIIT asks a student to implement the following code on multiple G-line processors.
>

```
for(i=1;i<=x;i++){
        for(j=1;j<=x;j++){
                for(k=1;k<=a;k++){
                        z=z%y;
                }
        }
        for(j=1;j<=b;j++){
                z=zy;
        }
}
for(i=1;i<=c;i++){
        z=z%y;
}
```

>

The students experiments and finds that the only significant operations are the modulus(%) and division() operation which take almost equal time. The time taken by other operations may be ignored in the order analysis. He finds a algorithm to solve the problem in which these operations can be carried out in random order. For his testing he chooses M processors . Each processor will carry out exactly M operations (% or /) .The performance is optimal when such a scheme exactly covers all the operations.
Puzzled, the student finds that this can only be done for some specific values of x for given a,b and c. He wants to trick the professor, so he needs few values of x for which his algorithm works. However, to make the professor feel that he manually did it these values of x need to be as small as possible.
>
> Given the values of a,b,c and k, output the first k values of x, for which the student's algorithm works.
>
> **Note:** The value of x should be greater than or equal to 0.
>

Input<h3>

The first line of input contains an integer t , the number of testcases. For each testcase , there is exactly one line which contains 4 space separated a,b,c and k.

Output

For each test case, output the corresponding k values of x, each in successive different lines.

Example

Input:

```
1
1 2 1 4
```

Output:

```
0
1
2
3
```

Constraints and Limits

t <= 10. The values in the output vi <= 10^12. Each of the intermediate values will fit in a 64 bit variable. The values a,b,c would be such that 0 <= a,b,c <= 100 and b^2-4ac >= 0. k <= 1000.
> Note : <b> Test data to this problem was modified on Feb 7.
> Note 2: <b> There were some mistakes in the test data discovered on March 11, 2008. New tricky cases provided by Blue Mary are also put up now and some "Accepted" solutions have received wrong answer. My apologies to one and all for the mistakes.

Added by:	**Ajay Somani**
Date:	**2008-02-05**
Time limit:	**2s**
Source limit:	**50000B**
Languages:	**All**
Resource:	**CodeCraft 08 Anshuman Singh**

SPOJ Problem Set (classical)

2420. Hospital at Hands

Problem code: HHAND

In a remote part of the Country, there lies a group of towns, quite far from any other areas. These towns are connected by a set of roads, having the property that there is exactly one path connecting any two towns, and every town is connected.
>
> Apollo Hospitals Ltd. decides to invest in this area, and build some Hospitals. Their analyst has a monumental task ahead of him. His job is to find out a Set of continuous towns, from among them, to build one hospital in each. The path connecting the first town to the last town in the set (which obviously passes through all the remaining ones) should not be more than length L, to avoid inconvenience to the visiting doctors. Also, the analyst has to make sure that his selection of target towns is such that the people of the area have to cover the least distance to reach the hospital closest to them.
>
> Thus, the towns where Hospitals will be built have to be chosen keeping in mind that the sum of distances that people from each town will need to cover, in order to reach the Hospital closest to them, should be Minimum. You have to find this minimum sum.
>

Input<h3>

First line of input contains an integer T which is equal to the number of test cases. You are required to process all test cases.Each test case starts with 2 space separated integers N,L. N denotes the number of towns and L is the length of path connecting first and last town in the set. Next N-1 lines follow each contains two space separated integers a and b denoting a road between A and B. A and B are 0 based.

Output

Output consists of T lines. I^{th} line in the output corresponds to the minimum sum total of the distances of all the towns with the nearest hospital for the I^{th} test case.

Example

Input:

```
2
3 1
0 1
```

```
1 2
4 1
0 1
1 2
2 3
```

Output :

```
1
2
```

Constraints and Limits

T <= 30, n <= 10000, 0 < L <= 100

Added by:	**Ajay Somani**
Date:	**2008-02-05**
Time limit:	**5s**
Source limit:	**50000B**
Languages:	**All**
Resource:	**CodeCraft 08 Aryan-Gaurav,Ajay Somani**

SPOJ Problem Set (classical)

2421. Incrementing The Integer

Problem code: ININT

Starting from the number ’1’, every time you can choose a digit from the current number and add it to the number itself. 23, for example, could be changed into 25 or 26. To get 100, using the above scheme, paths A and B are both possible. A requires 21 steps, but B needs only 17 (which is also the minimum)
A. 1-2-4-8-16-17-18-19-20-22-24-28-36-39-48-56-62-68-76-83-91-100
> B. 1-2-4-8-16-17-24-28-36-39-48-56-62-68-76-83-91-100
>
C is another 17 step solution for 100.
>
> C. 1-2-4-8-16-22-24-28-36-39-48-56-62-68-76-83-91-100
>
> Now, you are given several numbers, for each number, print the minimum steps S and number of solutions T. As T could be quite large, you should print T%1000000007 instead.

Input

Each line of input contains a integer K as a test case. Input ends with End Of File.

Output

For each test case print the minimum steps and solutions in a single line. If it’s impossible to get the number, print "IMPOSSIBLE" instead. (without the quotes).

Example

Input:
```
16
100
87
```

Output:
```
4 1
17 2
IMPOSSIBLE
```

Constraints and Limits

Number of test cases <= 100, 1 <= K <= 10^9.

Added by:	Ajay Somani
Date:	2008-02-05
Time limit:	6s
Source limit:	50000B
Languages:	All
Resource:	CodeCraft 08, Problem Setter: Jin Bin

SPOJ Problem Set (classical)

2422. Jazzy Job

Problem code: JAZZYJOB

With the magnification of the Energy Crisis, Chemists have decided to re-examine the existing procedures of preparation of various Chemical Substances.
>
> As part of this Project, they list all the elements that they commonly find as raw material (Initial Reactants), and the ones that they intend to produce (Final Products). They also prepare an extensive list of the various known reactions/processes that are used to convert one substance to another.
>
> One major issue that they find is that the Initiation of many reactions needs absurdly large amounts of energy. They wish to keep low the activation energies used in the new procedures.
>
> Knowing all this, they now attempt to find out such methods of preparing the target substances :
> (1) in which the highest value of activation energy needed by any of the reactions that make up the path, in any of the methods, does not exceed a given upper value,
> (2) which minimizes the Total reactions/procedures performed in All the (Initial Reactant) --> (Final Product) conversions.
>
> Each substance, on being given a specific amount of energy, converts into some other substance. The formed substance is unique for a particular value of Energy. The process of creating each successive Target Substance (Final Product) starts from one of the Source Substances Only, and leaves no by-products.
>
> Your task is to find the minimum value of upper bound such that all final products are obtainable with that upper bound and for this minimum value, find out the minimum number of conversions to get all the final products.
>
> **Note:** None of the procedures are Reversible, unless explicity stated. Also, if a procedure Is reversible, the Energy requirement may or may not be same. You may assume that there is a limitless supply of the Initial Reactants.
>

Input<h3>

The first line contains an integer t which is the number of test cases. Each test case begins with a line containing three integers : number of Initial eactants(S), Final Products(D) and the total number of elements (N). Then S+D lines follow, first S lines contain IDs of Initial reactant (0 based) and next D lines contain ID's of Final products (0 based). Then follows a line containing an integer R which is number of reactions possible. Then follow R lines, each containing three integers,

the Substance(S), the converted substance(C) and the activation energy(A) units required for the reaction. 0 <= S,C < n.

Output

For each test case , output in a different line ,2 integers (a,b) separated by spaces where a is the minimum upper value and b is the minimum number of conversions required for corresponding a. In case that all final products are not obtainable for any value of upper bound, Print a single line with message "Excessive Energy.".

Example

Input:

```
1
2 2 6
1
3
0
3
6
1 2 2
2 4 3
4 5 1
4 2 2
3 0 4
5 0 1
```

Output:

```
3 4
```

Constraints and Limits

N <= 10000, S <= N,D <= N,R < 100000 0 < A < 1000000000

Added by:	**Ajay Somani**
Date:	**2008-02-05**
Time limit:	**4s**
Source limit:	**50000B**
Languages:	**All**
Resource:	**CodeCraft 08 Aryan-Gaurav**

SPOJ Problem Set (classical)

2423. Minimal Triangulations of Graphs

Problem code: MINTRIAN

Check whether the given graph is chordal.

Input

The first line contains an integer 1<=t<=200 denoting the number of test cases. Then t graphs are given (not necessarily connected). Each graph is described by two lines. The first line contains a string of the form: n=nodes,m=edges: The second line gives the edges of the graph separated by commas. Each edge is given as a pair of vertices: {u,v}. Vertices of the graph are denoted with integers 0...,n-1.

Output

For each test case print YES if the graph is chordal, or NO if it isn't.

Example

Input:

```
2
n=6,m=4
{0,1} {2,3} {3,4} {3,5}
n=6,m=7
{0,3} {1,2} {1,3} {2,4} {2,5} {3,4} {3,5}
```

Output:

```
YES
NO
```

Added by: Tomasz Goluch
Date: 2008-02-05
Time limit: 60s
Source limit:50000B
Languages: All

SPOJ Problem Set (classical)

2426. Palindromes

Problem code: PLD

A palindrome is a word, phrase, number or other sequence of units that has the property of reading the same in either direction, e.g. 'racecar', 'solos'.

Task

You are given a number k (2<=k<=30000) and a non-empty string S whose length does not exceed 30000 lowercase letters.
We say two palindromes are different when they start from different positions. How many different palindromes of the length k does S contains?

Input

The first line contains K. The second line contains S. K does not exceed the length of S.

Output

The first and only line should consist of a single number - the number of palindromes found.

Example

Input:
5
ababab
Output:
2

Added by: Chinh Nguyen
Date: 2008-02-07
Time limit: 0.5s
Source limit:50000B
Languages: All

SPOJ Problem Set (classical)

2450. Counting Rabbits

Problem code: RABBIT1

Rabbits are incredible animals. One of their more interesting characteristics is related with their reproduction. If we keep a couple of adult rabbits in optimal conditions of life, it is scientifically proved that, each month, that couple is capable of procreating a new couple of young rabbits. You should know that only the adult couples may procreate and that the time taken by a young couple of rabbits to grow (that is, to become adult) is of 1 month. For the convenience of this task, we will be dealing with immortal rabbits.

Farmer Luis (FL) is a great admirer of rabbits. FL bought in the market 1 couple of adult rabbits (alive, of course) and know wants to raise as many rabbits as he can. Unfortunately, there is a little problem, FL has boxes where he can only put exactly 2^M (1 <= M <= 20) couples of rabbits (neither more nor less). FL can use as many boxes as he wishes as long as he fulfils the condition above. FL would like to know how many couples of rabbits he will not be able to put inside boxes if he raises rabbits for N (1 <= N <= 2147483647) months and then tries to 'box' them (put them inside boxes). You should help FL with these calculations. You must consider that FL starts with 1 adult couple of rabbits the 1st month, and that couples of rabbits reproduce and grow as stated in the 1st paragraph.

Input

Line 1: C (1 <= C <= 100), the number of calculations your program will be requested to do

Lines 2-C+1: two integers N and M (in that order)

Output

Lines 1-C: on each lines print S, which is the number of rabbits FL will not be able to 'box' for calculation # i

Example

Input:
```
1
5 2
```

Output:
```
0
```

Output explanation:

After growing couples of rabbits during 5 months, FL has 5 adult couples and 3 young couples (8 couples in total). FL has boxes where can put 2^2 = 4 couples of rabbits, so he can use 2 boxes to 'box' all the 8 couples. If FL had instead grown couples of rabbits for 4 months, he would have 5 couples in total; thus 1 couple would have remained un-'boxed' (the answer would have been 1).

Added by: Abel Nieto Rodriguez
Date: 2008-02-14
Time limit: 1s
Source limit: 50000B
Languages: All
Resource: Mindbend 2002 Programming Contest

SPOJ Problem Set (classical)

2485. Phone Lines

Problem code: PHONELIN

There are several cities and towers on a straight line. Towers can be set to connection-accepting by paying a cost. We are given the location (on the X-axis), of the towers and the cities. Our job is to set up certain towers as connection-accepting. Now every city, pays you an amount equal to **D - distance_travelled_by_data**, for every unit of data (for every tower) it can send. (distance_travelled_by_data = cityX - towerX); Our job here is to setup connections on differrent towers to get maximal profit.
Each city when it wants to route some data to a tower works with the following algorithm:
(1) Find the nearest tower to the left of the city.
(2) If it is within the range 'D', it sends the data to that tower. If this tower exceeds the range D, or if the tower doesnt accept connections, the city cant send the data and stops immediately. (Doesnt check the next available tower);
(3) If the data is sent sucessfully: Then the city
(3.1) Skips three towers. (Doesnt care if these three towers are connection-accepting or not);
(3.2) Tries to send data to the next tower (the fouth one after the skipping), by using step (2);

Input format:
Input consists of multiple testcases.
First line of each test case, contains two integers: D C T; The range, the number of cities and the number of towers, respectively.
Second line contains exactly C integers saying the location of the cities (on the X-axis).
The next T lines contain exactly two integers: location[i] connection-cost[i]; which is the position of tower i, and the cost to setup tower i as connection-accepting;
The input ends with a line: "-1 -1 -1"

Output format:
For each test case, output a single line saying the maximum amount of profit you can make.

Constraints:
Now two points (towers or cities), will have the same X-coordinate. T,C <= 100.

Sample Input:

```
4 9 6
23
43
18
15
29
50
41
31
40
32 2
26 0
46 7
```

```
48 0
50 3
38 1
-1 -1 -1
```

Sample Output:

```
5
```

Added by: Prasanna
Date: 2008-02-25
Time limit: 2s
Source limit: 50000B
Languages: All
Resource: ByteCode 2008

SPOJ Problem Set (classical)

2511. Magic Program IV

Problem code: MAGIC4

Click here to get the solution(TLE)

Added by:	Jin Bin
Date:	2008-03-05
Time limit:	1s
Source limit:	2000B
Languages:	All except: C99 strict
Resource:	own problem

SPOJ Problem Set (classical)

2523. Mispelling

Problem code: GNY07A

Misspelling is an art form that students seem to excel at. Write a program that removes the nth character from an input string.

Input

The first line of input contains a single integer N, (1 <= N <= 1000) which is the number of datasets that follow.

Each dataset consists of a single line of input containing M, a space, and a single word made up of uppercase letters only. M will be less than or equal to the length of the word. The length of the word is guaranteed to be less than or equal to 80.

Output

For each dataset, you should generate one line of output with the following values: The dataset number as a decimal integer (start counting at one), a space, and the misspelled word. The misspelled word is the input word with the indicated character deleted.

Example

Input:

```
4
4 MISSPELL
1 PROGRAMMING
7 CONTEST
3 BALLOON
```

Output:

```
1 MISPELL
2 ROGRAMMING
3 CONTES
4 BALOON
```

Added by: Marco Gallotta
Date: 2008-03-11
Time limit: 60s
Source limit:50000B
Languages: All
Resource: ACM Greater New York Regionals 2007

SPOJ Problem Set (classical)

2524. Conversions

Problem code: GNY07B

Conversion between the metric and English measurement systems is relatively simple. Often, it involves either multiplying or dividing by a constant. You must write a program that converts between the following units:

Type	Metric	English equivalent
Weight	1.000 kilograms	2.2046 pounds
	0.4536 kilograms	1.0000 pound
Volume	1.0000 liter	0.2642 gallons
	3.7854 liters	1.0000 gallon

Input

The first line of input contains a single integer N, (1 <= N <= 1000) which is the number of datasets that follow.

Each dataset consists of a single line of input containing a floating point (double precision) number, a space and the unit specification for the measurement to be converted. The unit specification is one of kg, lb, l, or g referring to kilograms, pounds, liters and gallons respectively.

Output

For each dataset, you should generate one line of output with the following values: The dataset number as a decimal integer (start counting at one), a space, and the appropriately converted value rounded to 4 decimal places, a space and the unit specification for the converted value.

Example

Input:

```
5
1 kg
2 l
7 lb
3.5 g
0 l
```

Output:

```
1 2.2046 lb
2 0.5284 g
3 3.1752 kg
4 13.2489 l
5 0.0000 g
```

Added by: Marco Gallotta
Date: 2008-03-11
Time limit: 60s
Source limit: 50000B
Languages: All
Resource: ACM Greater New York Regionals 2007

SPOJ Problem Set (classical)

2525. Encoding

Problem code: GNY07C

Chip and Dale have devised an encryption method to hide their (written) text messages. They first agree secretly on two numbers that will be used as the number of rows (R) and columns (C) in a matrix. The sender encodes an intermediate format using the following rules:

1. The text is formed with uppercase letters [A-Z] and <space>.
2. Each text character will be represented by decimal values as follows:

```
<space> = 0, A = 1, B = 2, C = 3, ..., Y = 25, Z = 26
```

The sender enters the 5 digit binary representation of the characters' values in a spiral pattern along the matrix as shown below. The matrix is padded out with zeroes (0) to fill the matrix completely. For example, if the text to encode is: "ACM" and R=4 and C=4, the matrix would be filled in as follows:

[IMAGE]

```
A = 00001, C = 00011, M = 01101
          (one extra 0)
```

The bits in the matrix are then concatenated together in row major order and sent to the receiver. The example above would be encoded as: 0000110100101100

Input

The first line of input contains a single integer N, (1 <= N <= 1000) which is the number of datasets that follow.

Each dataset consists of a single line of input containing R (1<=R<=20), a space, C (1<=C<=20), a space, and a text string consisting of uppercase letters [A-Z] and <space>. The length of the text string is guaranteed to be <= (R*C)/5.

Output

For each dataset, you should generate one line of output with the following values: The dataset number as a decimal integer (start counting at one), a space, and a string of binary digits (R*C) long describing the encoded text. The binary string represents the values used to fill in the matrix in row- major order. You may have to fill out the matrix with zeroes (0) to complete the matrix.

Example

Input:

```
4
4 4 ACM
5 2 HI
2 6 HI
5 5 HI HO
```

Output:

```
1 0000110100101100
2 0110000010
3 010000001001
4 0100001000011010110000010
```

Added by: Marco Gallotta
Date: 2008-03-11
Time limit: 60s
Source limit: 50000B
Languages: All
Resource: ACM Greater New York Regionals 2007

SPOJ Problem Set (classical)

2526. Decoding

Problem code: GNY07D

Chip and Dale have devised an encryption method to hide their (written) text messages. They first agree secretly on two numbers that will be used as the number of rows (R) and columns (C) in a matrix. The sender encodes an intermediate format using the following rules:

1. The text is formed with uppercase letters [A-Z] and <space>.
2. Each text character will be represented by decimal values as follows:

```
<space> = 0, A = 1, B = 2, C = 3, ..., Y = 25, Z = 26
```

The sender enters the 5 digit binary representation of the characters' values in a spiral pattern along the matrix as shown below. The matrix is padded out with zeroes (0) to fill the matrix completely. For example, if the text to encode is: "ACM" and R=4 and C=4, the matrix would be filled in as follows:

[IMAGE]

```
A = 00001, C = 00011, M = 01101
          (one extra 0)
```

The bits in the matrix are then concatenated together in row major order and sent to the receiver. The example above would be encoded as: 0000110100101100

Input

The first line of input contains a single integer N, (1 <= N <= 1000) which is the number of datasets that follow.

Each dataset consists of a single line of input containing R (1<=R<=20), a space, C (1<=C<=20), a space, and a string of binary digits that represents the contents of the matrix (R * C binary digits). The binary digits are in row major order.

Output

For each dataset, you should generate one line of output with the following values: The dataset number as a decimal integer (start counting at one), a space, and the decoded text message. You should throw away any trailing spaces and/or partial characters found while decoding.

Example

Input:

```
4
4 4 0000110100101100
5 2 0110000010
2 6 010000001001
5 5 0100001000011010110000010
```

Output:

```
1 ACM
2 HI
3 HI
4 HI HO
```

Added by: Marco Gallotta
Date: 2008-03-11
Time limit: 60s
Source limit: 50000B
Languages: All
Resource: ACM Greater New York Regionals 2007

SPOJ Problem Set (classical)

2527. Flipping Burned Pancakes

Problem code: GNY07E

The cook at the *Frobbozz Magic Pancake House* sometimes falls asleep on the job while cooking pancakes. As a result, one side of a stack of pancakes is often burned. Clearly, it is bad business to serve visibly burned pancakes to the patrons. Before serving, the waitress will arrange the stacks of pancakes so that the burned sides are facing down. You must write a program to aid the waitress in stacking the pancakes correctly.

We start with a stack of ***N*** pancakes of distinct sizes, each of which is burned on one side. The problem is to convert the stack to one in which the pancakes are in size order with the smallest on the top and the largest on the bottom and burned side down for each pancake. To do this, we are allowed to flip the top ***k*** pancakes over as a unit (so the ***k-th*** pancake is now on top and the pancake previously on top is now in the ***k-th*** position and the burned side goes from top to bottom and vice versa).

For example (+ indicates burned bottom, - a burned top):

```
          +1 -3 -2 [flip 2] => +3 -1 -2 [flip 1] => -3 -1 -2 [flip 3] =>
+2 +1 +3 [flip 1] => -2 +1 +3 [flip 2] => -1 +2 +3 [flip 1] => +1 +2 +3
```

You must write a program which finds a sequence of at most ***(3n - 2)*** flips, which converts a given stack of pancakes to a sorted stack with burned sides down.

Input

The first line of the input contains a single decimal integer, N, the number of problem instances to follow. Each of the following N lines gives a separate dataset as a sequence of numbers separated by spaces. The first number on each line gives the number, M, of pancakes in the data set. The remainder of the data set is the numbers 1 through M in some order, each with a plus or minus sign, giving the initial pancake stack. The numbers indicate the relative sizes of the pancakes and the signs indicate whether the burned side is up (-) or down (+). M will be, at most, 30.

Output

For each dataset, you should generate one line of output with the following values: The dataset number as a decimal integer (start counting at one), a space, the number of flips (K, where K >= 0) required to sort the pancakes and a sequence of K numbers, each of which gives the number of pancakes to flip on the corresponding sorting step. There may be several correct solutions for some datasets. For instance 3 2 3 is also a solution to the first problem below.

Example

Input:

```
3
3 +1 -3 -2
4 -3 +1 -2 -4
5 +1 +2 +3 +4 -5
```

Output:

```
1 6 2 1 3 1 2 1
2 6 4 1 4 3 1 2
3 3 5 1 5
```

Added by: Marco Gallotta
Date: 2008-03-11
Time limit: 60s
Source limit: 50000B
Languages: All
Resource: ACM Greater New York Regionals 2007

SPOJ Problem Set (classical)

2528. Monkey Vines

Problem code: GNY07F

Deep in the Amazon jungle, exceptionally tall trees grow that support a rich biosphere of figs and juniper bugs, which happen to be the culinary delight of brown monkeys.

Reaching the canopy of these trees requires the monkeys to perform careful navigation through the tall tree's fragile vine system. These vines operate like a see-saw: an unbalancing of weight at any vine junction would snap the vine from the tree, and the monkeys would plummet to the ground below. The monkeys have figured out that if they work together to keep the vines properly balanced, they can *all* feast on the figs and juniper bugs in the canopy of the trees.

A *vine junction* supports exactly two *sub-vines*, each of which must contain the same number of monkeys, or else the vine will break, leaving a pile of dead monkeys on the jungle ground. For purposes of this problem, a *vine junction* is denoted by a pair of matching square brackets [ ], which may contain nested information about junctions further down its *sub-vines*. The nesting of vines will go no further than **25** levels deep.

[IMAGE]

You will write a program that calculates the *minimum* number of monkeys required to balance a particular vine configuration. There is **always** at least one monkey needed, and, multiple monkeys may hang from the same vine.

Input

The first line of input contains a single integer N, (1 <= N <= 1000) which is the number of datasets that follow.

Each dataset consists of a single line of input containing a vine configuration consisting of a string of [and] characters as described above. The length of the string of [and] will be greater than or equal to zero, and less than or equal to 150.

Output

For each dataset, you should generate one line of output with the following values: The dataset number as a decimal integer (start counting at one), a space, and the minimum number of monkeys required to reach the canopy successfully. Assume that all the hanging vines are reachable from the jungle floor, and that all monkeys jump on the vines at the same time.

Example

Input:

```
3
[]

[[][[]]]
```

Output:

```
1 2
2 1
3 8
```

Added by: Marco Gallotta
Date: 2008-03-11
Time limit: 60s
Source limit: 50000B
Languages: All
Resource: ACM Greater New York Regionals 2007

SPOJ Problem Set (classical)

2529. Model Rocket Height

Problem code: GNY07G

Just when you thought we had run out of model rocket height problems...

Yet another method used to determine the height achieved by a model rocket is the vertical line method. Two observers A and B are spaced D feet apart along a base line along one edge of the flat test field. The launch platform is equidistant from observers A and B and L feet from the base line. Each observer has a theodolite or some other device for measuring angle above the horizontal (elevation angle) of a distant object and the azimuth angle (the angle the vertical plane of the sight line makes with the line from A through B measured counter-clockwise). Each measuring device is on a stand. A's device is HA feet above the level of the launch platform and B's device is HB feet above the level of the launch platform. When a rocket is fired, near the top of its flight, it deploys a parachute and emits a puff of smoke. Each observer measures the elevation angle and azimuth angle of the puff of smoke from their location. If the peak location is on the wrong side of the baseline or outside the lines determined by A and B perpendicular to the base line, it is out of bounds and disqualified. From this information, the height of the rocket may be determined as follows:

Each sight line determines a vertical plane. These two planes intersect in a vertical line (thus the name of the method). Each sight line intersects this vertical line in a point. If these points are more than ERRDIST feet apart, an error is assumed and the flight is rejected. Otherwise, the point halfway between the two points where a sight line intersects the vertical line is computed. The rocket height is the distance of this midpoint above the launch platform.

You must write a program which, given the parameters D (the distance in feet between observers A and B), L (the distance in feet from the base line to the launch platform), HA (the distance of the measuring device A above the launch platform in feet), HB (the distance of the measuring device B above the launch platform in feet), ERRDIST (the maximum distance between the intersection points of a sight line with the vertical line), a (the elevation angle of the rocket in degrees measured by the left observer A), b (the elevation angle of the rocket in degrees observed by the right observer B), g (the azimuth angle in degrees measured by the left observer A) and d (the azimuth angle in degrees measured by the right observer B), computes the height of the rocket above the launch platform in feet to the nearest foot.

Input

The first line of input contains a single integer N, (1 <= N <= 1000) which is the number of datasets that follow.

The second line contains the parameters D, L, HA, HB and ERRDIST in that order as (floating point) decimal values. These values would be measured once at the beginning of the day and remain fixed through all rocket shots.

Each succeeding line of input represents a single dataset. Each dataset will contain the angles a, b, g and d in that order (measured in degrees) as (floating point) decimal values for a rocket shot.

Output

For each dataset of four angles, the output consists of a single line . If angles a, b and g are not strictly between 0 and 90 degrees or d is not strictly between 90 degrees and 180 degrees, the line should contain the dataset number, a space and the word "DISQUALIFIED" (without the quotes). Otherwise, if the distance between the intersection points of a sight line with the vertical line is more that ERRDIST feet, the line should contain the dataset number, a space and the word "ERROR" (without the quotes). Otherwise, the line should contain the dataset number, a space and the height above the launch platform in feet to the nearest foot.

Example

Input:

```
4
100.0 300.0 5.25 2.92 5.00
40.1 36.2 35.3 151.6
64.9 71.1 15.7 160.1
44.9 41.2 33.1 152.5
44.9 41.2 33.1 52.5
```

Output:

```
1 50
2 ERROR
3 58
4 DISQUALIFIED
```

Added by: Marco Gallotta
Date: 2008-03-11
Time limit: 1s
Source limit:50000B
Languages: All
Resource: ACM Greater New York Regionals 2007

SPOJ Problem Set (classical)

2530. Tiling a Grid With Dominoes

Problem code: GNY07H

We wish to tile a grid 4 units high and ***N*** units long with rectangles (dominoes) 2 units by one unit (in either orientation). For example, the figure shows the five different ways that a grid 4 units high and 2 units wide may be tiled.

[IMAGE]

Write a program that takes as input the width, ***W***, of the grid and outputs the number of different ways to tile a 4-by-***W*** grid.

Input

The first line of input contains a single integer ***N***, (1 <= ***N*** <= 1000) which is the number of datasets that follow.

Each dataset contains a single decimal integer, the width, ***W***, of the grid for this problem instance.

Output

For each problem instance, there is one line of output: The problem instance number as a decimal integer (start counting at one), a single space and the number of tilings of a 4-by-***W*** grid. The values of ***W*** will be chosen so the count will fit in a 32-bit integer.

Example

Input:
```
3
2
3
7
```

Output:
```
1 5
2 11
3 781
```

Added by: Marco Gallotta
Date: 2008-03-12
Time limit: 60s
Source limit:50000B
Languages: All
Resource: ACM Greater New York Regionals 2007

SPOJ Problem Set (classical)

2531. Spatial Concepts Test

Problem code: GNY07I

The Flathead Testing Corporation (FTC) supplies various tests for Human Resources departments at many companies. One type of test they supply includes spatial concepts questions such as:

When the following figure is folded back on the interior lines it forms a cube.

[IMAGE]

Which of the following could be an image of one corner of the resulting cube?

[IMAGE]

Unfortunately, FTC was recently embarrassed when one such question on a test had no solution among the choices and another (given in the example) had two solutions among the choices (1 and 3).

FTC needs a routine which will read in a specification of the unfolded cube and specifications of corner views and determine, for each corner view, whether it is a view of a corner of the cube specified in the unfolded part.

FTC uses the following images as faces of each cube. Each image is symmetrical about the vertical axis and has a distinguished end (up in each image).

[IMAGE]

The unfolded cube is specified by a string of six pairs of a letter indicating the image on the face and a number indicating the orientation of the distinguished end of the face: 1 is up, 2 is right, 3 is down and 4 is left. The faces are specified in the order given in the following figure with the orientations indicated in the square to the right:

[IMAGE] [IMAGE]

So the unfolded cube in the example is specified as "F3E4E2D3C2F3". FTC has a routine which reads this specification and generates the unfolded image for the question.

The answer images are specified by three pairs of a letter and a digit indicating a face image and an orientation as indicated in the following diagram. The faces are specified in the order top, right, left (indicated by numbers in brackets in the figures), that is clockwise around the center vertex starting at the top. The orientation of the distinguished end of each face is indicated by the numbers on the edges in the diagram. They circle each face clockwise, starting at the center vertex.

[IMAGE]

For the example, the answer figures are specified as "C2D2F2", "E3F3C4", "F2C2D2", "D1E1F3" and "E1C1E1". Again, FTC has a routine which reads this specification and generates each answer image for the question. They just need your routine to make sure there is exactly one correct answer to each question.

Input

The first line of input contains a single integer N, (1 <= N <= 1000) which is the number of datasets that follow.

Each dataset consists of six lines of input. The first line of input is the specification for the folded out cube as described above. This line is followed by five lines, each of which gives the specification of one answer image as described above.

Output

For each dataset, output on a single line the dataset number, (1 through N), a blank, the number of answers which are solutions of the problem (corners of the cube specified in the folded out line), a blank and five 'Y' or 'N' characters separated by a blank indicating which of the answer images was a solution ('Y' for a solution, 'N' for not a solution).

Example

Input:

```
2
F3E4E2D3C2F3
C2D2F2
E3F3C4
F2C2D2
D1E1F3
E1C1E1
A2F4F1A3A3C4
C3A4A2
F3F4A1
F3C4A1
A2C3A2
A4A4F1
```

Output:

```
1 2 Y N Y N N
2 0 N N N N N
```

Added by: Marco Gallotta
Date: 2008-03-12
Time limit: 60s
Source limit: 50000B
Languages: All
Resource: ACM Greater New York Regionals 2007

SPOJ Problem Set (classical)

2565. Another Permutation Problem

Problem code: PERMUT3

Given a permutation of **n** elements (1, 2, ..., n): A = (a_1, a_2, ..., a_n). We define a sequence P(A)=(p_1, p_2, ..., p_{n-1}) where $p_i = 0$ if $a_i > a_{i+1}$ and $p_i = 1$ if $a_i < a_{i+1}$. Given a permutation B, find the number of all permutations C where P(C)=P(B) including the permutation B itself.

The length of your solution should not be more than 0.5kB.

Input

Multiple test cases. For each test case:

The first line contains an integer **n**(1<= **n** <=100).The second line contains **n** integers representing the permutation, all of which are separated by single spaces.

Input terminates by a single zero.

Output

For each test case:

The output contains a single line with a single integer - the number of the permutations having the same value for P(A) when given the permutation A.

Example

Input:
```
2
1 2
4
1 3 2 4
0
```

Output:
```
1
5
```

Added by: Blue Mary
Date: 2008-03-21
Time limit: 6s
Source limit:512B
Languages: All except: C99 strict
Resource: ACM Southeastern European Regional Contest 2001

SPOJ Problem Set (classical)

2643. Starcraft I

Problem code: SC1

Background

You may play the game Starcraft I first before you do this problem ^_^.

Description

Suppose you are using Protoss. At the beginning of the game, you have *n* probes, a nexus and almost unlimited number of pylons. You can build a probe in the nexus per 3 Starcraft time units(STs), and this will cost you *z* units of minerals. A probe can gather *x* units of minerals or *y* units of gas per ST. What's the minimum time to get *A* units of minerals and *B* units of gas, if you build probes at nexus only and don't build any buildings?

Assume that in the current map there are almost unlimited mineral fields and unlimited vespene geysers, and on each vespene geyser, a Protoss Assimilator has been built successfully.

Input

Multiple test cases, the number of them is given in the very first line.

Each test case contains one line with 6 integers *n*, *x*, *y*, *z*, *A*, *B* separated by one space. All numbers in the input file will be less than 21.

Output

For each test case, output one line, which contains a single integer, the minimum time in ST.

Example

Input:
```
1
1 2 3 4 5 6
```

Output:
```
5
```

Added by: Blue Mary
Date: 2008-04-09
Time limit: 7s
Source limit:50000B
Languages: All except: C99 strict
Resource: www.blizzard.com

SPOJ Problem Set (classical)

2648. Archiver

Problem code: KPARCH

One of your friends wants to write his own archiver. He is going to replace neighboring equal substrings with only one copy. For example, he is going to change substring "AA" with something like "2(A)" and if "A" is long enough it will reduce the file size.
But, before performing any coding stuff he wants to know how many such double substrings are there in the input file.
He asks you to help him, because this task is very difficult for him.

Input

Input file contains the text to be archived. It will only contain Latin letters (big and small). Its size will not exceed 200000 symbols. Letters are case sensitive, i.e. "X" is not equal to "x".

Output

Write a number of substrings of input text which can be written as "AA", i.e. consist of two equal concatenated parts.

Example

Input:
```
abcdefg
```

Output:
```
0
```

Input:
```
blabla
```

Output:
```
1
```

Input:
```
aCacaacaa
```

Output:
```
4
```

Added by: Pavel Kuznetsov
Date: 2008-04-11
Time limit: 1s
Source limit:50000B
Languages: All
Resource: IT Festival Arkhangelsk 2007

SPOJ Problem Set (classical)

2649. Weird sorting

Problem code: KPSORT

You are given **N** integer numbers a_1, a_2, ..., a_N. All you need to do is to sort them in non-decreasing order.
The bad thing is you are only allowed to perform one action. You can pick any number in the sequence and then reverse all elements to the left and to the right of it.
For example, suppose you were given the sequence (7, 1, 3, 9, 8) then, depending on the picked number you'll get the following results:

Picked number	Resulting sequence
7	(7, 8, 9, 3, 1)
1	(7, 1, 8, 9, 3)
3	(1, 7, 3, 8, 9)
9	(3, 1, 7, 9, 8)
8	(9, 3, 1, 7, 8)

In this problem you are to figure out whether the given sequence can be sorted or not, applying allowed action zero or more times.

Input

Input will contain multiple test cases (not more than 100). Each case will start with the number of elements in the sequence N (1 <= N <= 100), followed by the N integers not exceeding 1000 by the absolute value. Input ends with the value N = 0.

Output

For each test case write "1" if corresponding sequence can be sorted and "0" otherwise. Output must not contain spaces or line breaks.

Example

Input:
```
5
7 1 3 9 8
2
2 1
0
```

Output:
```
10
```

Added by: Pavel Kuznetsov
Date: 2008-04-12
Time limit: 1s
Source limit:50000B
Languages: All
Resource: IT Festival Arkhangelsk 2007

SPOJ Problem Set (classical)

2658. Art of War

Problem code: WAR

The *Warring States Period* (473-22l BC) refers to the centuries of turmoil following the Spring and Autumn Period. China was divided into many little kingdoms that were constantly fighting with each other. Unlike in previous ages, when chivalry played an important role in battles and the states fought mostly for balance of power or to resolve disputes, in this period the aim of battle was to conquer and completely annihilate the other states. Eventually seven states, known as the ‘‘Seven Great Powers’’ rose to prominence: Qi, Chu, Yan, Han, Zhao, Wei, and Qin. After numerous alliances and counter-alliances, Qin defeated all the other states one by one, putting an end to the Warring States Period.

You are given a map that shows the position of the capital for each state, and the borders between the states as a series of line segments. Your job is to determine which states were fighting with each other. This is pretty easy to determine - if two states had a common border, then they were fighting.

Input

The input contains several blocks of test cases. Each case begins with a line containing two integers: the number 1<= n <=600 of states, and the number 1<= m <= 4000 of border segments. The next n lines describe the coordinates of capitals, there are two integers in each line. The next m lines after that describe the m border segments. Each line contains four integers x_1, y_1, x_2 and y_2 meaning that there is a border segment from (x_1, y_1) to (x_2, y_2).(It is not given in the input what the two states on the two sides of the border are, but it can be deduced from the way the borders go.)

Each state is enclosed by a continuous borderline. The states are surrounded by an infinite wasteland, thus a border segment either separates two states, or a state from the wasteland. It is not possible that the same state is on both sides of a border segment, or the wasteland is on both sides of a border segment. There is exactly one capital in each state, and there is no capital in the wasteland. The border segments do not cross each other, they can meet only at the end points.

The input is terminated by a block with $n = m = 0$.

Output

For each test case, you have to output n lines that describe the enemies of the n states (recall that if two states share a border, then they are enemies). Each line begins with an integer, the number x of enemies the given state has. This number is followed by x numbers identifying the enemies of the state. These numbers are between 1 and n and number 1 refers to the first capital appearing in the input, number n refers to the last.

Example

Input:

```
4 12
3 2
11 8
12 17
1 19
0 0 10 0
10 0 20 0
20 0 20 10
20 10 20 20
20 20 10 20
10 20 0 20
0 20 0 10
0 10 0 0
10 0 10 10
0 10 10 10
20 10 10 10
10 20 10 10
4 16
170 13
24 88
152 49
110 130
60 60 140 60
140 60 140 140
140 140 60 140
60 140 60 60
0 0 200 0
200 0 200 200
200 200 0 200
0 200 0 0
40 40 160 40
160 40 160 160
160 160 40 160
40 160 40 40
20 20 180 20
180 20 180 180
180 180 20 180
20 180 20 20
0 0
```

Output:

```
2 2 4
2 1 3
2 2 4
2 1 3
1 2
2 1 3
2 2 4
1 3
```

Added by: Blue Mary
Date: 2008-04-19
Time limit: 25s
Source limit:50000B
Languages: All except: C99 strict
Resource: ACM Central European Programming Contest 2004(unofficial test data)

SPOJ Problem Set (classical)

2660. Example

Problem code: EXAMPLE

As you may have probably noticed a problem statement in a programming contest consists of several sections. The most important section is, of course, the "Example" section. Some seasoned contestants even start reading the problem statement from the examples. And, unfortunately, the least read section is the problem description section. It is quite disappointing for the problem authors because they feel that their writing skills are largely wasted.

So the authors decided to describe examples in the same language, as the rest of the problem statement using the following rules.

- Natural integer numbers shall be written in plain English. The numbers shall be less than one hundred. The designator number shall be written before numbers, except when the corresponding number is used as a **repetition factor**. For example, *number zero*, *number sixteen*, or *number sixty one*.
- Sequences of characters (strings) shall be written either in quotes, or in apostrophes, for example *"John's pen"*, or *'5" tall'*. Note, that ' may be used in strings enclosed in " and vice versa. The designator string shall be written before strings, for example *string 'Hello'*.
- The designator *space* denotes one space character.
- A number, string or space may be prefixed with a **repetition factor**. The repetition factor is a number greater than one. The designator after the repetition factor is written in plural form. For example, *four numbers five*, or *six strings 'A'*. If the repetition factor is used for numbers, the numbers are separated with one space character. So, the former example means *5 5 5 5*, but the latter example *AAAAAA*.
- Let the numbers, strings and spaces with possible repetition factors be called **fragments**. Fragments may be organized into **sequences** using the *followed by* copulative. For example, *number five followed by number six*. One implicit space character is assumed between numbers, a number and a string, and a string and a number so the example above means *5 6*.
- An example is described line by line. The first line is always described as *The first line...*. The following lines are described either as *The next line...* or as *The next # lines...*, where # is a number greater than one. Empty lines are described as *is empty* or *are empty*. Other lines are described as *contains* or *contain* followed by sequences. The first letter of a sentence is capitalized. The sentence is terminated by a full stop (**.**). The full stop is not separated by space from the preceding word, but is separated by at least one space from the next word.

Input

The input contains a free-flow text describing an example. Words are separated by an arbitrary number of spaces and newlines. There are no whitespace characters after the last full stop. The total size of the input will be no greater than 1 MB.

Output

The output should contain the decoded example. The total size of the output shall be no greater than 1 MB.

Example

Input:

```
The first line contains four
numbers twenty eight. The next
line is empty. The next two
lines contain six strings '-'.
The next line contains number
four followed by number seventy
seven followed by string 'meat'
followed by three strings "!".
```

Output:

```
28 28 28 28
------

------
4 77 meat!!!
```

Added by: Blue Mary
Date: 2008-04-19
Time limit: 1s
Source limit: 50000B
Languages: All except: C99 strict
Resource: ACM/ICPC NEERC Moscow Subregional Contest 2007

SPOJ Problem Set (classical)

2661. Illumination

Problem code: ILLUM

Two cubes and a light bulb are placed in a three-dimensional euclidean space. You are expected to find out if one of them casts shadow on the other one and if so, calculate the area of this shadow.

Input

Multiple test cases. For each test case:

The first line of the input contains the coordinates of the bulb. It is followed by two groups of four lines each that describe the cubes. Each line of the cube description contains the coordinates of a vertex (see the figure where the vertices are marked and labeled in the same order as they are given in the input).

[IMAGE]

All the coordinates are given with 5 digits after decimal point. It is guaranteed that the cubes do not intersect, the light bulb is outside both of them, and doesn't lie on any of the planes that contain their faces. A light bulb should be regarded as a point light source.

Input terminates by EOF.

Output

For each test case:

The output should contain a single line with two numbers separated with a space character. The first one is the number of the cube that has a shadow on it (1 or 2). The second is the area of the shadow. If none of the given cubes casts shadow on the other the output should contain a single number -1.

Note: if your output has an error with absolute value less than 10^{-2}, it will be judged as Accepted. i.e. You may output any number of digits after decimal point.

Example

Input:

```
-1.00000 1.00000 1.00000
0.00000 0.00000 0.00000
2.00000 0.00000 0.00000
0.00000 2.00000 0.00000
0.00000 0.00000 2.00000
5.00000 0.00000 0.00000
7.00000 0.00000 0.00000
5.00000 2.00000 0.00000
5.00000 0.00000 2.00000
0.00000 0.00000 0.00000
1.00000 1.00000 1.00000
```

```
2.00000 1.00000 1.00000
1.00000 2.00000 1.00000
1.00000 1.00000 2.00000
-1.00000 -1.00000 -1.00000
-1.00000 -2.00000 -1.00000
-2.00000 -1.00000 -1.00000
-1.00000 -1.00000 -2.00000
```

Output:

```
2 4.000
-1
```

Added by: Blue Mary
Date: 2008-04-19
Time limit: 2s
Source limit: 50000B
Languages: All except: C99 strict
Resource: ACM/ICPC NEERC Moscow Subregional Contest 2007

SPOJ Problem Set (classical)

2662. Put a Point in a Hyperspace

Problem code: PUTIN

Input

Multiple test cases, the number of them is given in the very first line.

For each test case:

The first line contains 3 space-separated integers K(2<=K<=30), S(2<=S<=10000), M(0<=M<=20). M lines follow, each contains K non-negative integers a_{ij} (1<=i<=M, 1<=j<=K), which shows that there is one point (a_{i1}, a_{i2}, ... a_{ik}) in the K-D hyperspace. No two point will be the same, and none of them lies on any (coordinate) axis.

Output

For each test case:

Output a single integer which shows the number of the points B(b_1, b_2, ... b_k) in the hyperspace satiesfied the following constraints:

- B is not on any (coordinate) axis.
- For each 1<= *i* <= M, there exist *j*, 1<=j<=k, such that $b_j < a_{ij}$.
- For each 1<=j<=k, b_j is a non-negative integer.
- The sum of b_j doesn't exceed S.

Example

Input:
```
1
2 4 2
1 3
2 1
```

Output:
```
2
```

Hint
```
The two points are (1,1) and (1,2).
```

Added by: Blue Mary
Date: 2008-04-19
Time limit: 90s
Source limit: 50000B
Languages: All except: C99 strict
Resource: ACM/ICPC NEERC Northern Subregion Contest 2003; description by Blue Mary

SPOJ Problem Set (classical)

2666. Query on a tree IV

Problem code: QTREE4

You are given a tree (an acyclic undirected connected graph) with N nodes, and nodes numbered 1,2,3...,N. Each edge has an integer value assigned to it(note that the value can be negative). Each node has a color, white or black. We define dist(a, b) as the sum of the value of the edges on the path from node a to node b.

All the nodes are white initially.

We will ask you to perfrom some instructions of the following form:

- **C a** : change the color of node a.(from black to white or from white to black)
- **A** : ask for the maximum dist(a, b), both of node a and node b must be white(a can be equal to b). Obviously, as long as there is a white node, the result will alway be non negative.

Input

- In the first line there is an integer N (N <= 100000)
- In the next N-1 lines, the i-th line describes the i-th edge: a line with three integers a b c denotes an edge between a, b of value c (-1000 <= c <= 1000)
- In the next line, there is an integer Q denotes the number of instructions (Q <= 100000)
- In the next Q lines, each line contains an instruction "C a" or "A"

Output

For each "A" operation, write one integer representing its result. If there is no white node in the tree, you should write "They have disappeared.".

Example

Input:

```
3
1 2 1
1 3 1
7
A
C 1
A
C 2
A
C 3
A
```

Output:

```
2
2
0
They have disappeared.
```

Some new test data cases were added on Apr.29.2008, all the solutions have been rejudged.

Added by: Qu Jun
Date: 2008-04-25
Time limit: 1s-6s
Source limit:50000B
Languages: All except: C99 strict
Resource: william

SPOJ Problem Set (classical)

2668. Polygon

Problem code: POLYSSQ

You are given N different points in the plane. No any 3 of them are collinear. Write a program that finds out the smallest area of a convex polygon with K vertices which are taken from the given points.

Input

Two integers, N and K, are written on the first line in the standard input. It follows N lines, each containing a pair of coordinates for the corresponding given point. Every two numbers on a line in the input are separated by a space. Constraints: 0 < N < 50, 0 < K < 11. The coordinates of the given points are nonnegative integers, less than 9999.

Output

Your program has to output an integer that is equal to the integer part of minimal area. If there does not exist any convex polygon as is described above, your program has to output 0.

Example

Input:

```
4 3
0 0
1 1
0 10
10 0
```

Output:

```
5
```

Added by: Chinh Nguyen
Date: 2008-04-26
Time limit: 1s
Source limit:50000B
Languages: All
Resource: Bulgarian OI

SPOJ Problem Set (classical)

2670. Count Minimum Spanning Trees

Problem code: MSTS

Your task is simple in this problem: count the number of **minimum spanning tree** (Wikipedia) in a simple undirected graph. The number of minimum spanning trees mean in how many ways you can select a subset of the edges of the graphs which forms a minimum spanning tree.

Input

The first line of input contains two integers **N** (1 <= **N** <= 100), **M** (1 <= **M** <= 1000). Nodes are labeled from 1 to **N**. In the following **M** lines, every line contains three integers $\mathbf{a_i}$, $\mathbf{b_i}$, $\mathbf{c_i}$, representing an undirected edge from node $\mathbf{a_i}$ to node $\mathbf{b_i}$, with weight $\mathbf{c_i}$. (1 <= $\mathbf{a_i}$!= $\mathbf{b_i}$ <= **N**, 1 <= $\mathbf{c_i}$ <= 1,000,000,000). You can assume there is at most one edge between two nodes, and the graph described by input is connected.

Output

Print the **answer** % 31011.

Example

Input:
```
4 6
1 2 1
1 3 1
1 4 1
2 3 2
2 4 1
3 4 1
```

Output:
```
8
```

Added by: Jin Bin
Date: 2008-04-29
Time limit: 1s
Source limit: 50000B
Languages: All except: C99 strict
Resource: Jiangsu TSC for Chinese NOI 08, day 2

SPOJ Problem Set (classical)

2699. Recursive Sequence (Version II)

Problem code: SPP

Sequence (a_i) of natural numbers is defined as follows:
$a_i = b_i$ (for $i <= k$)
$a_i = c_1 a_{i-1} + c_2 a_{i-2} + ... + c_k a_{i-k}$ (for $i > k$)
where b_j and c_j are given natural numbers for $1<=j<=k$. Your task is to compute $a_m + a_{m+1} + a_{m+2} + ... + a_n$ for given $m <= n$ and output it modulo a given positive integer p.

Input

On the first row there is the number C of test cases (equal to about 50).
Each test contains four lines:
k - number of elements of (c) and (b) ($1 <= k <= 15$)
$b_1,...,b_k$ - k natural numbers where $0 <= b_j <= 10^9$ separated by spaces
$c_1,...,c_k$ - k natural numbers where $0 <= c_j <= 10^9$ separated by spaces
m, n, p - natural numbers separated by spaces ($1 <= m <= n <= 10^{18}$, $1<= p <= 10^8$)

Output

Exactly C lines, one for each test case: $(a_m + a_{m+1} + a_{m+2} + ... + a_n)$ modulo p.

Example

Input:
```
1
2
1 1
1 1
2 10 1000003
```

Output:
```
142
```

Added by: Blue Mary
Date: 2008-05-15
Time limit: 7s
Source limit:50000B
Languages: All except: C99 strict
Resource: Problem SEQ

SPOJ Problem Set (classical)

2709. Untitled Problem

Problem code: UNTITLED

We consider a sequence $\mathbf{S_1}$ is **equal** to a sequence $\mathbf{S_2}$, if and only if they satisfy the following conditions:

- The length of them are equal.
- Let **Len** be the length of them. For each i,j(1 <= i, j <= **Len**, i != j):If $\mathbf{S_1}$[i] is smaller than $\mathbf{S_1}$[j], $\mathbf{S_2}$[i] must be smaller than $\mathbf{S_2}$[j]; If $\mathbf{S_1}$[i] is greater than $\mathbf{S_1}$[j], $\mathbf{S_2}$[i] must greater than $\mathbf{S_2}$[j].

Now you are given a sequence **S** and another **N** sequences $\mathbf{T_1}$, $\mathbf{T_2}$ .... $\mathbf{T_N}$.

We say position **i** is **OK**, if and only if **S**[1..**i**] contains a suffix which is **equal** to a sequence from { $\mathbf{T_1}$, $\mathbf{T_2}$ ... $\mathbf{T_N}$ }. You need to print the positions which is **OK** in increasing order.

Input

Multiple test cases, the number of them(no more than 3) is given in the very first line.

For each test case:

- The first line contains an integer **M** (**M** > 1) which denote the number of sequences. **i.e. M = N + 1.**
- **M** * 2 lines follow, each two lines describe one sequence.For each two lines, the first line contains an integer **L** which denote the length of this sequence. The second line contains **L** integers(all the integers don't exceed 2^{31}-1) that represent this sequence. The first sequence described is **S**, the next **N** sequences represent $\mathbf{T_1}$ ... $\mathbf{T_N}$.
- You can assume that there are no same integer in any one sequence.
- The length of **S** is no more than 400000, and the total length of **T** is no more than 100000.

Output

For each test case: Print the positions which is **OK** in increasing order.

Example

Input:

```
2
2
1
1
1
2
3
3
3 1 2
2
```

```
4 5
2
10 1
```

Output:

```
1
2
3
```

Added by: Qu Jun
Date: 2008-05-19
Time limit: 8s
Source limit: 50000B
Languages: All except: C99 strict
Resource: Based on a problem from USACO 2005 Dec Gold Division

SPOJ Problem Set (classical)

2714. Cow Cars

Problem code: COWCAR

N (1 <= N <= 50,000) cows conveniently numbered 1, ..., N are driving in separate cars along a highway in Cowtopia. Cow i can drive in any of M different high lanes (1 <= M <= N) and can travel at a maximum speed of S_i (1 <= S_i <= 1,000,000) km/hour.

After their other bad driving experience, the cows hate collisions and take extraordinary measures to avoid them. On this highway, cow i reduces its speed by D (0 <= D <= 5,000) km/hour for each cow in front of it on the highway (though never below 0 km/hour). Thus, if there are K cows in front of cow i, the cow will travel at a speed of max(S_i - D*K, 0). While a cow might actually travel faster than a cow directly in front of it, the cows are spaced far enough apart so crashes will not occur once cows slow down as described.

Cowtopia has a minimum speed law which requires everyone on the highway to travel at a a minimum speed of L (1 <= L <= 1,000,000) km/hour, so sometimes some of the cows will be unable to take the highway if they follow the rules above. Write a program that will find the maximum number of cows that can drive on the highway while obeying the minimum speed limit law.

Input

The first line contains the four integers N, M, D, and L. For the next N lines, line i+1 contains the integer S_i.

Output

Print a single integer denoting the maximum number of cows that can take the highway.

Example

Input:
```
3 1 1 5
5
7
5
```

Output:
```
2
```

We can obtain two cows by putting either cow with speed 5 first and the cow with speed 7 second.

Added by: Neal Wu
Date: 2008-05-22
Time limit: 1s
Source limit:50000B
Languages: All
Resource: USACO Open 2008

SPOJ Problem Set (classical)

2715. Glasnici

Problem code: GLASNICI

A long straight road connects two villages. Along the road, N messengers are stationed and, when needed, they exchange messages using mostly their legs, but also their vocal cords and ears. The first messenger (the closest to the first village) has a radio-receiver which he uses to keep track of current ongoings in the country. When he finds out who has been evicted from whichever reality show is currently popular, he starts running as fast as he can to share the unfortunate (or fortunate) news with everyone else. While running, he shouts the name of the evicted person so that any fellow messengers that are close enough can hear him. Meanwhile, the remaining messengers do not merely sit and wait, but also run themselves, all with the selfless goal of sharing the news with everyone as fast as possible. The running and shouting proceeds as follows:

- Each of the messengers may run whenever, in either direction, at a speed of at most 1 unit per second, or may decide not to run at all and stand still.
- All messengers that know the news shout it at all times. One messenger can hear another messenger shouting (and learn the news) if the distance between them is at most K units.

Write a program that, given the initial locations of the messengers, determines the least amount of time (in seconds) needed for all messengers to learn the news. The location of every messenger is given with a positive real number - the distance from the first village. As mentioned above, initially only the first messenger knows the news.

Input

The first line contains the integer T (1 <= T <= 25), the number of test cases. Then for each test case, the first line contains the real number K (0 <= K <= 1,000,000), the largest distance at which two messengers can hear each other, and the integer N (1 <= N <= 100,000), the number of messengers. Each of the following N lines contains one real number D (0 <= D <= 1,000,000,000), the distance of one messenger from the first village. The distances will be sorted in ascending order. It is possible for multiple messengers to be at the same location.

Output

For each test case, output a real number, the least time for all messengers to learn the news. Your output needs to be within 0.01 of the official output.

Example

Input:

```
2
3.000 2
0.000
6.000
2.000 4
0.000
4.000
```

```
4.000
8.000
```

Output:

```
1.500
1.000
```

Warning: large input/output data.

Added by: Neal Wu
Date: 2008-05-23
Time limit: 20s
Source limit: 50000B
Languages: All
Resource: Croatian Olympiad in Informatics 2008

SPOJ Problem Set (classical)

2716. Maximal Quadrilateral Area

Problem code: QUADAREA

You are trying to build a house, but unfortunately you currently have only four available walls with side lengths a, b, c, and d. You want your house to be as big as possible, so you would like to know the largest possible area of any quadrilateral you can construct with these four side lengths.

Input

The first line contains the integer T (1 <= T <= 2,000), the number of tests. Each test contains a single line with four real numbers: a, b, c, and d (0 < a, b, c, d < 1,000). Note that it will always be possible to form a valid quadrilateral with these lengths; that is, the sum of any three side lengths will be strictly larger than the other one.

Output

For each test case, print a single line containing the largest possible area. Your output will be accepted if it is within 0.01 of the official answer.

Example

Input:
```
2
1 2 1 2
0.5 0.5 0.5 0.5
```

Output:
```
2.00
0.25
```

For the first test case, it is optimal to construct a rectangle, and for the second, a square is optimal.

Added by: Neal Wu
Date: 2008-05-24
Time limit: 1s
Source limit:50000B
Languages: All

SPOJ Problem Set (classical)

2727. Army Strength

Problem code: ARMY

The next MechaGodzilla invasion is on its way to Earth. And once again, Earth will be the battleground for an epic war.

MechaGodzilla's army consists of many nasty alien monsters, such as Space Godzilla, King Gidorah, and MechaGodzilla herself.

To stop them and defend Earth, Godzilla and her friends are preparing for the battle.

Problem specification

Each army consists of many different monsters. Each monster has a strength that can be described by a positive integer. (The larger the value, the stronger the monster.)

The war will consist of a series of battles. In each battle, the weakest of all the monsters that are still alive is killed.

If there are several weakest monsters, but all of them in the same army, one of them is killed at random. If both armies have at least one of the weakest monsters, a random weakest monster of MechaGodzilla's army is killed.

The war is over if in one of the armies all monsters are dead. The dead army lost, the other one won.

You are given the strengths of all the monsters. Find out who wins the war.

Input specification

The first line of the input file contains an integer T specifying the number of test cases. Each test case is preceded by a blank line.

Each test case starts with line containing two positive integers NG and NM - the number of monsters in Godzilla's and in MechaGodzilla's army. Two lines follow. The first one contains NG positive integers - the strengths of the monsters in Godzilla's army. Similarly, the second one contains NM positive integers - the strengths of the monsters in MechaGodzilla's army.

Output specification

For each test case, output a single line with a string that describes the outcome of the battle.

If it is sure that Godzilla's army wins, output the string "Godzilla".

If it is sure that MechaGodzilla's army wins, output the string "MechaGodzilla".

Otherwise, output the string "uncertain".

Example

input:

```
2

1 1
1
1

3 2
1 3 2
5 5
```

output:

```
Godzilla
MechaGodzilla
```

Hint

In the first test case, there are only two monsters, and they are equally strong. In this situation, MechaGodzilla's monster is killed and the war ends.

In the second test case, the war will consist of three battles, and in each of them one of Godzilla's monsters dies.

For all the test cases, **int** in C/C++/Java or **longint** in Pascal is enough.

Added by: Blue Mary
Date: 2008-05-24
Time limit: 1s-2s
Source limit: 50000B
Languages: All except: C99 strict
Resource: IPSC 2008

SPOJ Problem Set (classical)

2728. Breaking in

Problem code: BREAK

Mayco has recently been hired as a security consultant for a well-known software company. At the moment, he's working on his first assignment - trying to determine which of the company's servers would be the best targets for potential attackers. It is a bit difficult, though, because some of the servers "trust" some of the others. If an attacker compromises a server, he or she can also freely access all servers that trust it (and servers that trust them, and so on).

By definition, the importance of a server S is the number of servers the attacker would be able to access if he compromised S. The most important servers are those with the highest importance. (Note that there can be more than one most important server. This is also illustrated in the example below.)

Problem specification

The network consists of N computers, numbered 1 to N, inclusive. The trust between computers is described by M ordered pairs (A,B) of numbers, denoting that computer A trusts computer B. The trust is not assumed to be mutual - i.e., if a computer A trusts computer B, it does not necessarily imply that computer B trusts computer A.

Input specification

The first line of the input file contains an integer T specifying the number of test cases. Each test case is preceded by a blank line.

Each test case starts with a line containing the numbers N and M(1<= N <=9000, 1<= M <= 52000). Each of the following M lines contains two integers, A and B, denoting that computer A trusts computer B.

Output specification

For each test case, the output shall contain one line with the numbers of all of the most important servers. The numbers must be listed in increasing order and separated by single spaces.

Example

input:

```
2

5 4
3 1
3 2
4 3
5 3

6 5
1 2
```

```
2 3
3 1
1 4
5 6
```

output:

```
1 2
4
```

Note

Blue Mary has found a pruning which will make the program very efficient. So the time limit of the hard test case is changed from 60 seconds to 15 seconds. If you have some even harder test case, please send it to me, and I'll add it to the standard input file.

Added by: Blue Mary
Date: 2008-05-24
Time limit: 1s-15s
Source limit:50000B
Languages: All except: C99 strict
Resource: IPSC 2008

SPOJ Problem Set (classical)

2731. Inventing Test Data

Problem code: INVENT

Preparing a problem set is a very hard task. There are always issues with clarity of problem statements, bugs in our solutions, input or output data, and so on. Sometimes, despite our best efforts, these issues are only found during the contest, and this can really spoil it.

To prevent this from happening in the future, we already started to prepare data for IPSC 2009, and we decided to use your help in doing so. Currently we are working on a simple textbook problem: "Given a weighted undirected complete graph, find its minimum spanning tree." (See the Definitions below if you are not sure what a spanning tree is.)

Almost everythig is already prepared for this problem: the problem statement, our solution, and also the desired output data. The only (and quite important) thing left is the input data. But creating it is not as simple as it looks.

The bad thing that can happen is that a graph can have more than one minimum spanning tree. If we used such a graph in the input data, we would have to write a complicated checker. And we are too lazy to do this. Therefore we want to find an input data that avoids such cases.

Moreover, we want the test data to be good. If all the other edges were much more expensive, the minimum spanning tree would be obvious, and many incorrect algorithms would be able to find it. Therefore we want all the edge weights to be as small as possible.

Definitions

A graph is a set of nodes, and a set of links. Each link connects two nodes. Each pair of nodes is connected by at most one link. Each link is assigned a positive integer (its weight). The sum of the weights of all links in a graph is the weight of that graph.

If every two nodes are connected by a link we say that the graph is complete.

A sequence of nodes v_0, ..., v_n such that for each i the nodes v_i and v_{i+1} are connected by a link, is called a path.

If every two nodes in a graph are connected by a path, we say that the graph is connected.

If there is exactly one path between any two nodes we say that graph is a tree.

A spanning subgraph of a connected graph G is a connected graph that contains all nodes of G and some (not necessarily all) of its links.

A spanning subgraph T of a graph G is called the minimum spanning tree of G if and only if no other spanning subgraph has a smaller weight.

Note that a given graph can have more than one spanning tree. Also note that a spanning tree is always a tree.

Problem specification

Given a weighted tree T, you are to find the minimum possible weight of a complete graph G such that T is the only minimum spanning tree of G.

Input specification

The first line of the input file contains an integer T specifying the number of test cases. Each test case is preceded by a blank line.

First line of each test case contains an integer N(N <= 15000) - number of nodes in the tree. The nodes are numbered from 1 to N, inclusive. The following N - 1 lines contain a description of the tree. Each of these lines contains three integers a_i, b_i, w_i(1<= a_i, b_i <=N, 1<= w_i <=10000) meaning that node a_i is connected with node b_i by a link with weight w_i.

Output specification

For each test case, the output shall contain one line containing one integer - the minimum possible weight of a complete graph such that the given tree is its unique minimal spanning tree.

Example

Input:

```
2

3
1 2 4
2 3 7

4
1 2 1
1 3 1
1 4 2
```

output:

```
19
12
```

Hint

In the first test case, we have to add a link between nodes 1 and 3 with weight at least 8.

In the second test case, the optimal graph contains the link 2 - 3 with weight 2, and links 2 - 4 and 3 - 4 with weigths 3 each.

Added by:	Blue Mary
Date:	2008-05-24
Time limit:	1s
Source limit:	50000B
Languages:	All except: C99 strict
Resource:	IPSC 2008

SPOJ Problem Set (classical)

2733. K Equal Digits

Problem code: KEQ

Every once in a while, Mishka Jabereen sends an interesting mathematical puzzle to his friends. This week's puzzle will be about so-called "repetitive" numbers - the ones whose decimal expansion has just one kind of digit in it. (Examples of such numbers include 7, 11, and 5555.) The puzzle is about finding the largest repetitive number subject to two additional restrictions:

```
(A) It must be divisible by at least one number from a given set.
(B) It can not have more than a given number of digits.
```

A concrete example of such a problem statement would be: Find the largest repetitive number with at most 47 digits, which is divisible by 42 or 47! Mishka is currently playing around with a few such problem statements and he'd like to know all the answers, so that he can choose the nicest one.

Problem specification

A puzzle is described by a number K, the maximal number of digits allowed in the repetitive number, and a set of numbers d_1, d_2, ..., d_R. Your task is to find the greatest repetitive number X that has at most K digits when written in decimal notation, and it is divisible by at least one of the d_i.

Input specification

The first line of the input file contains an integer T specifying the number of test cases. Each test case is preceded by a blank line.

Each test case starts with a line containing the numbers K($1 <= K <= 10^9$) and R($1 <= R <= 7$). The next R lines contain the numbers d_i ($1 <= d_i <= 10^{14}$).

Output specification

We can describe a repetitive number by specifying its number of digits N and the digit D it contains.

For each test case, the output shall contain one line containing two integers N and D that describe the largest repetitive number that satisfies the conditions from the problem statement.

Example

input:

```
3

47 2
42
47

99 4
123
```

```
234
345
456

3 1
4700
```

output:

```
46 9
96 6
1 0
```

Note that in the third test case "3 0" would not be a correct answer, as "000" is not a valid integer.

Blue Mary's Note: This problem can be solved in a very short time but a naive solution may not terminate within 5 hours. Thanks to Robert Gerbicz's help.

Added by:	Blue Mary
Date:	2008-05-24
Time limit:	0.5s
Source limit:	50000B
Languages:	All except: C99 strict
Resource:	IPSC 2008

SPOJ Problem Set (classical)

2734. Large party

Problem code: LARGE

Irena and Sirup are organizing their engagement party next weekend. They want to invite almost everybody. They have just bought a very big round table for this occasion. But they are now wondering how should they distribute people around the table. Irena claimed that when there are more than K women next to each other, this group will chat together for the whole night and won't talk to anybody else.

Sirup had no other choice but to agree with her. However, being a mathematician, he quickly became fascinated by all the possible patterns of men and women around the table.

Problem specification

There will be N people sitting at the round table. Some of them will be men and the rest will be women.

Your task is to count in how many ways it is possible to assign the places to men and women in such a way that there will not be more than K women sitting next to each other.

If one assignment can be made from another one by rotating all the people around the table, we consider them equal (and thus count this assignment only once).

Input specification

The first line of the input file contains an integer T specifying the number of test cases. Each test case is preceded by a blank line.

The input for each test case consists of a single line that contains the two integers N(1<= N <1000) and K.

Output specification

For each test case output a single line with one integer - the number of ways how to distribute people around the table, modulo 100000007.

Example

input:

```
3

3 1

3 3

4 1
```

output:

```
2
4
3
```

Hint:

In the first test case there are two possibilities: MMM or MMW (M is a man, W is a woman).

In the second test case there are two more possibilities: MWW and WWW.

In the third test case the three possibilities are: MMMM, MMMW, and MWMW.

A Note: The official solution is too slow for this problem...

Added by: Blue Mary
Date: 2008-05-24
Time limit: 1.5s
Source limit: 50000B
Languages: All except: C99 strict
Resource: IPSC 2008

SPOJ Problem Set (classical)

2735. Simplify the Railroad System

Problem code: RAIL

The Slovak national railroad company has recently built new tracks. They want to update their railroad map according to these changes. But they want the map to be as simple as possible. So they decided to remove from the map all the stations that have exactly two other direct connections to other stations (i.e., a single railroad passing through the station).

Problem specification

You will be given the complete map of Slovak railroads. It consists of railway stations numbered from 1 to N, and railroad segments between some pairs of these stations. For each railroad segment we are given its length.

Your task is to remove all such stations that are directly connected with exactly two other stations, and output the new map. The new map must contain correct distances between the remaining stations.

Input specification

The first line of the input file contains an integer T specifying the number of test cases. Each test case is preceded by a blank line.

Each test case begins with a line with two integers N(1<= N <=2000) and M(1<= M <=3000). The number N denotes the number of stations and M is the number of railroad segments. M lines follow, each with 3 integers a, b, and c (1 <= a,b <= N) specifying that there is a railroad segment of length c connecting stations a and b.

You can assume that in each test case there is a path between every two stations, and there is at most one railroad between any pair of cities directly. Also, there will always be at least 2 stations that are not directly connected to exactly two other stations.

Output specification

For each test case, the output shall consist of multiple lines. The first line shall contain a positive integer K - the number of railroads on the simplified map. Each of the next K lines shall contain three integers a, b (a must be no more than b), and c stating that there is a railroad of length c between stations a and b on the simplified map.

The railroads should be listed in lexicographic order, i.e. the railroad with less a should be listed first,if two railroads have the same a, then the one with less b should be listed first. If two railroad have the same a and b, the one with less c should be listed first.

Print a blank line between outputs for different test cases.

Example

input:

```
2

3 2
1 2 1
2 3 1

4 4
1 2 1
2 3 2
3 4 3
4 2 1
```

output:

```
1
1 3 2

2
1 2 1
2 2 6
```

Hint

In the first case we removed station 2 beacuse it had exactly 2 direct connections.

In the second case we removed stations 3 and 4. We see that there is now a railroad from station 2 back to itself.

For all the test cases, **int** in C/C++/Java or **longint** in Pascal is enough.

Added by: Blue Mary
Date: 2008-05-24
Time limit: 1s-2s
Source limit:50000B
Languages: All except: C99 strict
Resource: IPSC 2008

SPOJ Problem Set (classical)

2737. Perfect Rhyme

Problem code: PRHYME

```
A perfect rhyme is not a crime,
it is something that exceeds time,
a bit of science, a piece of art,
soft as a pillow, sharp as a dart.
```

Everyone tried it, but only few chosen ones succeeded. It is a hard task with an unclear path, but a famous end - should you reach it. Many compare it to finding the Holy Grail, or even to finding Waldo. The task is to find a perfect rhyme.

Problem specification

Given is a wordlist **L**, and a word **w**. Your task is to find a word in **L** that forms a perfect rhyme with **w**. This word **u** is uniquely determined by these properties:

- It is in **L**.
- It is different from **w**.
- Their common suffix is as long as possible.
- Out of all words that satisfy the previous points, **u** is the lexicographically smallest one.

Notes

A prefix of a word is any string that can be obtained by repeatedly deleting the last letter of the word. Similarly, a suffix of a word is any string that can be obtained by repeatedly deleting the first letter of the word.

For example, consider the word *different*.

This word is both its own prefix and suffix. Its longest other prefix is *differen*, and its longest other suffix is *ifferent*. The string *rent* is its yet another, even shorter suffix. The strings *eent* and *iffe* are neither prefixes nor suffixes of the word *different*.

Let **u** and **v** be two different words. We say that **u** is lexicographically smaller than **v** if either **u** is a prefix of **v**, or if *i* is the first position where they differ, and the *i*-th letter of **u** is earlier in the alphabet than the *i*-th letter of **v**.

For example, *dog* is smaller than *dogs*, which is smaller than *dragon* (because *o* is less than *r*).

Input specification

The input file consists of two parts. The first part contains the wordlist **L**, one word per line. Each word consists of lowercase English letters only, and no two words are equal.

The first part is terminated by an empty line.

The second part follows, with one query word **w** per line.

You may assume that in either part of the input, the length of a word will be no more than 30. And the number of words in each part of the input will be no more than 250000. The input file will be less than 5MB.

Output specification

For each query in the input file output a single line with its perfect rhyme. The output must be in lowercase.

Example

input:
```
perfect
rhyme
crime
time

crime
rhyme
```

output:
```
time
crime
```

In the second test case, there were two candidates that had an equally long common suffix (crime and time), the lexicographically smaller one was chosen.

Warning: large input/output data, be careful with certain languages

---

Added by: Blue Mary
Date: 2008-05-26
Time limit: 20s
Source limit:50000B
Languages: All except: C99 strict
Resource: IPSC 2008

SPOJ Problem Set (classical)

2742. Summing Sums

Problem code: SUMSUMS

The N (1 <= N <= 50,000) cows, conveniently numbered 1, 2, ..., N, are trying to learn some encryption algorithms. After studying a few examples, they have decided to make one of their own! However, they are not very experienced at this, so their algorithm is very simple:

Each cow i is given a starting number C_i (0 <= C_i < 90,000,000), and then all the cows perform the following process in parallel:

- First, each cow finds the sum of the numbers of the other N-1 cows.
- After all cows are finished, each cow replaces her number with the sum she computed. To avoid very large numbers, the cows will keep track of their numbers modulo 98,765,431.

They told Canmuu the moose about it in November; he was quite impressed. Then one foggy Christmas Eve, Canmuu came to say:

> "Your algorithm is too easy to break! You should repeat it T (1 <= T <= 1,414,213,562) times instead."

Obviously, the cows were very frustrated with having to perform so many repetitions of the same boring algorithm, so after many hours of arguing, Canmuu and the cows reached a compromise: You are to calculate the numbers after the encryption is performed!

Input

- Line 1: Two space-separated integers: N and T.
- Lines 2..N+1: Line i+1 contains a single integer: C_i.

Output

- Lines 1..N: Line i contains a single integer representing the number of cow i (modulo 98,765,431) at the end of the encryption.

Example

Input:
```
3 4
1
0
4
```

Output:
```
26
25
29
```

The following is a table of the cows' numbers for each turn:

	Cows' numbers		
Turn	Cow1	Cow2	Cow3
0	1	0	4
1	4	5	1
2	6	5	9
3	14	15	11
4	26	25	29

Warning: large input/output data.

Added by: Neal Wu
Date: 2008-05-28
Time limit: 1s
Source limit: 50000B
Languages: All
Resource: USACO Chn 2007

SPOJ Problem Set (classical)

2743. Prefix Tiling

Problem code: PRETILE

You are given a string S with N (1 <= N <= 100,000) characters from 'A' to 'Z', inclusive. For an integer L between 1 and N, inclusive, we define match (L) as the length of the longest prefix of S that can be tiled by the length-L prefix of S; more specifically, match (L) is the smallest 0-based index k such that S [k] != S [k mod L], or N if no such k exists. For example, when S = "ABCAB", match (1) = 1, match (3) = 5, and match (4) = 4. Compute the sum match (1) + match (2) + ... + match (N).

Input

The first line contains the integer T (1 <= T <= 10), the number of tests. For each test, there is a single line containing the string S.

Output

For each test case, print a single line containing one integer: the value of match (1) + match (2) + ... + match (N).

Example

Input:
```
2
ABCAB
ZZZZZZ
```

Output:
```
17
36
```

For the first test case, match (1) + match (2) + match (3) + match (4) + match (5) = 1 + 2 + 5 + 4 + 5 = 17. For the second, the sum is equal to 6 * 6 = 36.

Warning: large input/output data.

Added by: Neal Wu
Date: 2008-05-28
Time limit: 1s-2s
Source limit: 50000B
Languages: All

SPOJ Problem Set (classical)

2815. Increasing Subsequences

Problem code: INCSEQ

Given a sequence of N (1 <= N <= 10,000) integers S_1, ..., S_N (0 <= S_i < 100,000), compute the number of increasing subsequences of S with length K (1 <= K <= 50 and K <= N); that is, the number of K-tuples i_1, ..., i_K such that 1 <= i_1 < ... < i_K <= N and S_{i_1} < ... < S_{i_K}.

Input

The first line contains the two integers N and K. The following N lines contain the integers of the sequence in order.

Output

Print a single integer representing the number of increasing subsequences of S of length K, modulo 5,000,000.

Example

Input:
```
4 3
1
2
2
10
```

Output:
```
2
```

The two 3-tuples are (1, 2, 4) and (1, 3, 4), both corresponding to the subsequence 1, 2, 10.

Added by: Neal Wu
Date: 2008-06-20
Time limit: 1s-2s
Source limit:50000B
Languages: All
Resource:

SPOJ Problem Set (classical)

2816. Common Subsequences

Problem code: CSUBSEQS

You are given four strings, each consisting of at most 50 lower case letters ('a'-'z'). Count the number of non-empty common subsequences of them (the number of distinct non-empty strings which are subsequences of all four strings). Note that a subsequence does not have to be contiguous.

Input

Four lines: each line consists of a single string.

Output

An integer representing the answer.

Example

Input:
```
aabb
abab
baba
acba
```

Output:
```
4
```

The four sequences are "a", "b", "aa", and "ab".

Added by: Jin Bin
Date: 2008-06-22
Time limit: 3s
Source limit:50000B
Languages: All except: C99 strict
Resource: co-author: Neal Wu

SPOJ Problem Set (classical)

2817. Distinct Increasing Subsequences

Problem code: INCDSEQ

Given a sequence of N (1 <= N <= 10,000) integers S_1, ..., S_N (0 <= S_i < 1,000,000,000), compute the number of distinct increasing subsequences of S with length K (1 <= K <= 50 and K <= N).

Input

The first line contains the two integers N and K. The following N lines contain the integers of the sequence in order.

Output

Print a single integer representing the number of distinct increasing subsequences of S of length K, modulo 5,000,000.

Example

Input:
```
4 3
1
2
2
10
```

Output:
```
1
```

The only increasing subsequence of length 3 is 1, 2, 10.

Added by: Jin Bin
Date: 2008-06-22
Time limit: 2s
Source limit: 50000B
Languages: All except: C99 strict
Resource: co-author: Neal Wu

SPOJ Problem Set (classical)

2826. Round-Robin Scheduling

Problem code: RRSCHED

A computer processor is given N tasks to perform (1 <= N <= 50,000). The i-th task requires T_i seconds of processing time (1 <= T_i <= 1,000,000,000). The processor runs the tasks as follows: each task is run in order, from 1 to N, for 1 second, and then the processor repeats this again starting from task 1. Once a task has been completed, it will not be run in later iterations. Determine, for each task, the total running time elapsed once the task has been completed.

Input

The first line of the input contains the integer N, and the next N lines contain the integers T_1 through T_N.

Output

Output N lines, the i-th of which contains an integer representing the time elapsed when task i has been processed.

Example

Input:

```
5
8
1
3
3
8
```

Output:

```
22
2
11
12
23
```

The second task is completed during the first iteration, finishing 2 seconds in. On the third iteration, the third and fourth tasks complete at 11 seconds and 12 seconds, respectively. Finally, on the eighth iteration, the first and last tasks complete at 22 seconds and 23 seconds, respectively.

Warning: large input/output data.

(See this post for more information.)

Added by: Neal Wu
Date: 2008-07-02
Time limit: 1s-2s
Source limit: 50000B
Languages: All
Resource: TopCoder High School SRM 52 (with raised constraints)

SPOJ Problem Set (classical)

2829. Time Limit Exceeded

Problem code: TLE

Given integers N (1 <= N <= 50) and M (1 <= M <= 15), compute the number of sequences a_1, ..., a_N such that:

- $0 <= a_i < 2^M$
- a_i is not divisible by c_i ($0 < c_i <= 2^M$)
- a_i & $a_{i+1} = 0$ (that is, a_i and a_{i+1} have no common bits in their binary representation)

Input

The first line contains the number of test cases, T (1 <= T <= 10). For each test case, the first line contains the integers N and M, and the second line contains the integers c_1, ..., c_N.

Output

For each test case, output a single integer: the number of sequences described above, modulo 1,000,000,000.

Example

Input:
```
1
2 2
3 2
```

Output:
```
1
```

The only possible sequence is 2, 1.

time limit has been doubled, enjoy :D

Added by: Jin Bin
Date: 2008-07-04
Time limit: 1s-10s
Source limit:5000B
Languages: All except: C99 strict
Resource: co-author: Neal Wu

SPOJ Problem Set (classical)

2832. Find The Determinant III

Problem code: DETER3

Given a NxN matrix A, find the Determinant of A % P.

Input

Multiple test cases (the size of input file is about 3MB, all numbers in each matrix are generated randomly).

The first line of every test case contains two integers , representing N (0 < N < 201) and P (0 < P < 1,000,000,001). The following N lines each contain N integers, the j-th number in i-th line represents A[i][j] (- 1,000,000,001 < A[i][j] < 1,000,000,001).

Output

For each test case, print a single line contains the answer.

Example

Input:

```
1 10
-528261590
2 2
595698392 -398355861
603279964 -232703411
3 4
-840419217 -895520213 -303215897
537496093 181887787 -957451145
-305184545 584351123 -257712188
```

Output:

```
0
0
2
```

Added by: Jin Bin
Date: 2008-07-05
Time limit: 20s
Source limit:50000B
Languages: All except: C99 strict
Resource: own problem

SPOJ Problem Set (classical)

2833. Super Dice Game

Problem code: SDGAME

Alice and Bob are playing a game. The game consists of a circular track of M (2 <= M <= 1,000,000,000) cells labeled 0 through M - 1. Initially both players start at cell 0. The game progresses by having each player take turns rolling one of N (1 <= N <= 10,000) 'super-dice' labeled 0 through N - 1. The actual mechanics of the 'super-dice' is not very well understood; however, it is known that they will only ever turn up a number between 0 and 1,000,000,000 inclusive after a roll. After rolling the super-dice the number of spaces a player moves is determined by the product of a contiguous subsequence of the values shown on the dice (There are special rules for determining the range that vary each move that will not be discussed).

To make matters more complicated, after any turn if Alice and Bob land on the same cell the value shown on all dice is multiplied by the label of the cell they are on. Note in this way it is possible for some dice to show numbers greater than 1,000,000,000. This multiplier does not apply to future rolls.

After playing this game for a while, Alice and Bob have grown frustrated because the calculations became too difficult. Given the series of R (1 <= R <= 100,000) dice rolls and ranges, help Alice and Bob determine their position after each move. Assume that all dice start out showing 1.

Input

The first line contains R, N, and M each separated by a space. R lines follow. Each line will contain d v a b separated by a space. d indicates the label of the dice rolled. v indicates the value shown on the dice. a and b indicate the range of dice used to determine the move distance.

Output

R lines containing the position of the player that just rolled after their roll.

Example

Input:

```
6 4 20
1 5 1 1
3 10 2 3
2 3 0 3
1 2 0 3
1 5 1 2
0 7 0 1
```

Output:

```
5
10
15
10
10
0
```

Output Explanation:

```
For your assistance, here is the state of the dice after each turn:
[1, 5, 1, 1]
[1, 5, 1, 10]
[1, 5, 3, 10]
[1, 2, 3, 10]
[10, 50, 30, 100]
[7, 50, 30, 100]
```

Warning: large Input/Output data, be careful with certain languages

---

Added by: Mark Gordon
Date: 2008-07-05
Time limit: 2s-10s
Source limit: 50000B
Languages: All
Resource: own problem

SPOJ Problem Set (classical)

2835. Memory Limit Exceeded

Problem code: MLE

Given **n** points on X-Y plane. To each point, you are to find the other point who is closest to it with respect to the Euclidean distance.

Input

T (<= 15) test cases. Each starts with an integer **n** (2<= **n** <=100000). Then **n** lines follow. Each contains two space-seperated integers, the X and Y coordinate of the corresponding point, respectively. No two points in one test case will coincide.

Output

For each test case, output **n** lines. The i-th of them should contain the squared distance between the i-th point from the input and its nearest neighbour.

Example

Input:

```
2
10
17 41
0 34
24 19
8 28
14 12
45 5
27 31
41 11
42 45
36 27
15
0 0
1 2
2 3
3 2
4 0
8 4
7 4
6 3
6 1
8 0
11 0
12 2
13 1
14 2
15 0
```

Output:

```
200
```

```
100
149
100
149
52
97
52
360
97
5
2
2
2
5
1
1
2
4
5
5
2
2
2
5
```

Warning: enormous input/output data, be careful with certain languages

Note: In Sphere Online Judge system, "Memory Limit Exceeded" will be shown as "Runtime Error(other)", with the 0.00 second run-time & 92-200k memory used, or "Runtime Error(SIGSEGV)" with 250M memory used.

Added by: Blue Mary
Date: 2008-07-08
Time limit: 23s
Source limit:50000B
Languages: All except: C99 strict
Resource: ACM Central European Programming Contest, Wrocław 2008

SPOJ Problem Set (classical)

2852. Broken Keyboard

Problem code: BROKEN

Bruce Force's keyboard is broken, only a few keys are still working. Bruce has figured out he can still type texts by switching the keyboard layout whenever he needs to type a letter which is currently not mapped to any of the *m* working keys of the keyboard.

You are given a text that Bruce wishes to type, and he asks you if you can tell him the maximum number of consecutive characters in the text which can be typed without having to switch the keyboard layout. For simplicity, we assume that each key of the keyboard will be mapped to exactly one character, and it is not possible to type other characters by combination of different keys. This means that Bruce wants to know the length of the largest substring of the text which consists of at most *m* different characters.

Input

The input contains several test cases, each test case consisting of two lines. The first line of each test case contains the number *m* (*1 <= m <= 128*), which specifies how many keys on the keyboard are still working. The second line of each test case contains the text which Bruce wants to type. You may assume that the length of this text does not exceed 1 million characters. Note that the input may contain space characters, which should be handled like any other character.

The last test case is followed by a line containing one zero.

Output

For each test case, print one line with the length of the largest substring of the text which consists of at most *m* different characters.

Example

Input:

```
5
This can't be solved by brute force.
1
Mississippi
0
```

Output:

```
7
2
```

Added by: Adrian Kuegel
Date: 2008-07-12
Time limit: 10s
Source limit: 50000B
Languages: All
Resource: University of Ulm Local Contest 2008

SPOJ Problem Set (classical)

2853. Decode the Strings

Problem code: PDECODE

Bruce Force has had an interesting idea how to encode strings. The following is the description of how the encoding is done:

Let $x_1,x_2,...,x_n$ be the sequence of characters of the string to be encoded.

1. Choose an integer *m* and *n* pairwise distinct numbers $p_1,p_2,...,p_n$ from the set {*1*, *2*, ..., *n*} (a permutation of the numbers *1* to *n*).
2. Repeat the following step *m* times.
3. For *1* <= i <= *n* set y_i to x_{p_i}, and then for *1* <= i <= *n* replace x_i by y_i.

For example, when we want to encode the string "hello", and we choose the value *m = 3* and the permutation *2, 3, 1, 5, 4*, the data would be encoded in 3 steps: "hello" -> "elhol" -> "lhelo" -> "helol".

Bruce gives you the encoded strings, and the numbers *m* and p_1, ..., p_n used to encode these strings. He claims that because he used huge numbers *m* for encoding, you will need a lot of time to decode the strings. Can you disprove this claim by quickly decoding the strings?

Input

The input contains several test cases. Each test case starts with a line containing two numbers *n* and *m* (*1 <= n <= 80*, *1 <= m <= 10^9*). The following line consists of *n* pairwise different numbers $p_1,...,p_n$ (*1* <= p_i <= *n*). The third line of each test case consists of exactly *n* characters, and represent the encoded string. The last test case is followed by a line containing two zeros.

Output

For each test case, print one line with the decoded string.

Example

Input:

```
5 3
2 3 1 5 4
helol
16 804289384
13 10 2 7 8 1 16 12 15 6 5 14 3 4 11 9
scssoet tcaede n
8 12
5 3 4 2 1 8 6 7
encoded?
0 0
```

Output:

```
hello
second test case
encoded?
```

Added by:	Adrian Kuegel
Date:	2008-07-12
Time limit:	5s
Source limit:	50000B
Languages:	All
Resource:	University of Ulm Local Contest 2008

SPOJ Problem Set (classical)

2855. Forest

Problem code: FOREST2

Bruce Force is standing in the forest. He wonders what is the tree trunk the farthest away which is not blocked from his view by other tree trunks.

Bruce has made a map of the trees in the forest. The map shows his current position as the origin of a cartesian coordinate system. Tree *i* is shown on the map as a circle with the center (x_i, y_i) and radius r_i. You may assume that a tree trunk is visible if and only if there exists a line segment on the map from the origin (0,0) to a point on the border of the circle representing the tree trunk, where the line segment does not intersect or touch another circle.

Input

The input contains several test cases. The first line of each test case contains one number *n* (*1 <= n <= 1000*), where *n* specifies how many trees are on the map. The following *n* lines contain 3 integers x_i, y_i, r_i each, (*-10000 <=* x_i, y_i *<= 10000* , *1 <=* r_i *<= 1000*) where (x_i, y_i) is the center of the circle representing tree trunk *i*, and r_i is the radius of the circle. You may assume that no two circles in the input intersect, i.e., for any two circles, the distance between their centers is more than the sum of their radii. Moreover, you may assume that no circle contains the origin.

The last test case is followed by a line containing one zero.

Output

For each test case, print one line with the maximum euclidean distance from the origin to a visible tree. The distance to a tree should be measured using the point of the tree closest to the origin, no matter if this point is in fact visible or not.

Round the answer to 3 digits after the decimal point.

Example

Input:

```
3
10 10 11
1 1 1
-20 -10 20
5
1 2 2
-2 1 1
2 -1 1
-1 -2 2
10000 -10000 1000
0
```

Output:

```
3.142
1.236
```

Added by:	Adrian Kuegel
Date:	2008-07-12
Time limit:	5s
Source limit:	50000B
Languages:	All
Resource:	University of Ulm Local Contest 2008

SPOJ Problem Set (classical)

2856. Help Bob

Problem code: HELPBOB

Bob loves Pizza but is always out of money. One day he reads in the newspapers that his favorite pizza restaurant, Alfredo's Pizza Restaurant, is running a competition: they will donate a big pizza to the first person who will tell them the lowest price per area that can be achieved by buying any of the pizzas at most once. "That task is easy!", thinks Bob, "For each pizza I just calculate the average price and the lowest quotient will be the answer.".

Unfortunately the problem is a bit more complicated: with some pizzas Alberto gives out discount coupons for getting another pizza cheaper and even worse, those coupons can be combined. The pizzas have to be bought one after the other, and it is not possible to use a coupon to get a discount retrospectively for a pizza which has already been bought. Can you help Bob to become the first to solve this task and to get a pizza for free?

Input

The input file contains several test cases. Each test case starts with a number *m*, the number of pizzas Alfredo offers. Input is terminated by *m=0*. Otherwise, *1 <= m <= 15*. Then follow *m* lines describing the pizzas. Each of those following lines describes pizza *i* (*1 <= i <= m*) and starts with 3 integer numbers p_i, a_i and n_i specifying the price of the pizza, its area and the number of discount coupons you get when buying it, *1 <=* p_i *<= 10000*, *1 <=* a_i *<= 10000* and *0 <=* n_i *< m*. Then follow n_i pairs of integer numbers $x_{i,j}$ and $y_{i,j}$ specifying the index $x_{i,j}$ (*1 <=* $x_{i,j}$ *<= m,* $x_{i,j}$ *!= i*) of the pizza you get a discount coupon for and the discount in percentage terms $y_{i,j}$ (*1 <=* $y_{i,j}$ *<= 50*) you get when buying pizza $x_{i,j}$. You may assume that for each *i* the values $x_{i,j}$ are pairwise distinct.

Output

For each test case print one line containing the lowest price per area that can be achieved by buying any of the pizzas at most once. Round this number to 4 places after the decimal point. Note that you can combine an arbitrary number of discount coupons: for a pizza with price 10 and two rabatt coupons for that pizza with a 50 and a 20 on it, you would only have to pay 10 * 0.8 * 0.5 = 4 monetary units.

Example

Input:

```
1
80 30 0
2
200 100 1 2 50
200 100 0
5
100 100 2 3 50 2 50
100 100 1 4 50
100 100 1 2 40
```

```
600 600 1 5 10
1000 10 1 1 50
0
```

Output:

```
2.6667
1.5000
0.5333
```

Added by: Adrian Kuegel
Date: 2008-07-12
Time limit: 5s
Source limit: 50000B
Languages: All
Resource: University of Ulm Local Contest 2008

SPOJ Problem Set (classical)

2877. Another understanding of Super Dice Game

Problem code: SDGAME2

When we were trying to solve the problem SDGAME, we got a misunderstanding of it.We didn't get AC until we were told the original meaning.But we think our kind of understanding is also interesting and is worthy of doing.So enjoy the problem.

Alice and Bob are playing a game. The game consists of a circular track of M (2 <= M <= 1,000,000,000) cells labeled 0 through M - 1. Initially both players start at cell 0. The game progresses by having each player take turns rolling one of N (1 <= N <= 10,000) 'super-dice' labeled 0 through N - 1. The actual mechanics of the 'super-dice' is not very well understood; however, it is known that they will only ever turn up a number between 0 and 1,000,000,000 inclusive after a roll. After rolling the super-dice the number of spaces a player moves is determined by the product of a contiguous subsequence of the values shown on the dice **(which are available)**(There are special rules for determining the range that vary each move that will not be discussed).**If all the values are unavailable,the player moves one space.Iff the number on the dice is more than 1,000,000,000 or less than 0, the dice is unavailable.**

To make matters more complicated, after any turn if Alice and Bob land on the same cell the value shown on all dice**(neither available nor unavailable)** is multiplied by the label of the cell they are on. Note in this way it is possible for some dice to show numbers greater than 1,000,000,000.

After playing this game for a while, Alice and Bob have grown frustrated because the calculations became too difficult. Given the series of R (1 <= R <= 100,000) dice rolls and ranges, help Alice and Bob determine their position after each move. Assume that all dices start out showing 1 **and all dices are available**.

Input

The first line contains R, N, and M each separated by a space. R lines follow. Each line will contain d v a b separated by a space. d indicates the label of the dice rolled. v indicates the value shown on the dice. a and b indicate the range of dice used to determine the move distance.

Output

R lines containing the position of the player that just rolled after their roll.

Example

Input:

```
6 4 4
0 1000000000 1 1
1 999999998 1 1
2 500000000 3 3
0 1 2 2
3 1 0 3
0 6 0 3
```

Output:

```
1
2
2
2
0
0
```

Output Explanation:

```
For your assistance, here is the state of the dice after each turn:(* means unavailable)
Before all rolls:
[1,1,1,1](0,0)
After first roll:
[1000000000,1,1,1](1,0)
After second roll:
[1000000000,999999998,1,1](1,2)
After third roll:
[1000000000,999999998,500000000,1](2,2)
All dices multiply 2:
[*,*,1000000000,2](2,2)
After forth roll:
[1,*,1000000000,2](2,2)
All dices multiply 2:
[2,*,*,4](2,2)
After fifth roll:
[2,*,*,1](0,2)
After sixth roll:
[6,*,*,1](0,0)
All dices multiply 0:
[0,0,0,0](0,0)
```

Test data has been updated, all submissions have been rejudged

---

Added by: Zhang Taizhi
Date: 2008-07-24
Time limit: 2s-4s
Source limit:50000B
Languages: All except: C99 strict
Resource: Based on 2833. Super Dice Game

SPOJ Problem Set (classical)

2878. Knights of the Round Table

Problem code: KNIGHTS

Being a knight is a very attractive career: searching for the Holy Grail, saving damsels in distress, and drinking with the other knights are fun things to do. Therefore, it is not very surprising that in recent years the kingdom of King Arthur has experienced an unprecedented increase in the number of knights. There are so many knights now, that it is very rare that every Knight of the Round Table can come at the same time to Camelot and sit around the round table; usually only a small group of the knights is there, while the rest are busy doing heroic deeds around the country.

Knights can easily get over-excited during discussions-especially after a couple of drinks. After some unfortunate accidents, King Arthur asked the famous wizard Merlin to make sure that in the future no fights break out between the knights. After studying the problem carefully, Merlin realized that the fights can only be prevented if the knights are seated according to the following two rules:

* The knights should be seated such that two knights who hate each other should not be neighbors at the table. (Merlin has a list that says who hates whom.) The knights are sitting around a round table, thus every knight has exactly two neighbors.

* An odd number of knights should sit around the table. This ensures that if the knights cannot agree on something, then they can settle the issue by voting. (If the number of knights is even, then it can happen that "yes" and "no" have the same number of votes, and the argument goes on.)

Merlin will let the knights sit down only if these two rules are satisfied, otherwise he cancels the meeting. (If only one knight shows up, then the meeting is canceled as well, as one person cannot sit around a table.) Merlin realized that this means that there can be knights who cannot be part of any seating arrangements that respect these rules, and these knights will never be able to sit at the Round Table (one such case is if a knight hates every other knight, but there are many other possible reasons). If a knight cannot sit at the Round Table, then he cannot be a member of the Knights of the Round Table and must be expelled from the order. These knights have to be transferred to a less-prestigious order, such as the Knights of the Square Table, the Knights of the Octagonal Table, or the Knights of the Banana-Shaped Table. To help Merlin, you have to write a program that will determine the number of knights that must be expelled.

Input

The input contains several blocks of test cases. Each case begins with a line containing two integers 1 <= n <= 1000 and 1 <= m <= 1000000. The number n is the number of knights. The next m lines describe which knight hates which knight. Each of these m lines contains two integers k_1 and k_2, which means that knight number k_1 and knight number k_2 hate each other (the numbers k_1 and k_2 are between 1 and n).

The input is terminated by a block with n = m = 0.

Output

For each test case you have to output a single integer on a separate line: the number of knights that have to be expelled.

Example

Input:

```
5 5
1 4
1 5
2 5
3 4
4 5
0 0
```

Output:

```
2
```

Warning: large input/output data, be careful with certain languages

---

Added by: Blue Mary
Date: 2008-07-24
Time limit: 10s
Source limit: 50000B
Languages: All except: C99 strict
Resource: ACM Central European Programming Contest, Budapest 2005

SPOJ Problem Set (classical)

2879. The Cow Doctor

Problem code: DOCTOR

Texas is the state having the largest number of cows in the US: according to the 2005 report of the NationalAgricultural Statistics Service, the bovine population of Texas is 13.8 million. This is higher than thepopulation of the two runner-up states combined: there are only 6.65 million cows in Kansas and 6.35millions cows in Nebraska.

There are several diseases that can threaten a herd of cows, the most feared being "Mad Cow Disease" or Bovine Spongiform Encephalopathy (BSE); therefore, it is very important to be able to diagnose certain illnesses. Fortunately, there are many tests available that can be used to detect these diseases.

A test is performed as follows. First a blood sample is taken from the cow, then the sample is mixed with a test material. Each test material detects a certain number of diseases. If the test material is mixed with a blood sample having any of these diseases, then a reaction takes place that is easy to observe. However, if a test material can detect several diseases, then we have no way to decide which of these diseases is present in the blood sample as all of them produce the same reaction. There are materials that detect many diseases (such tests can be used to rule out several diseases at once) and there are tests thatdetect only a few diseases (they can be used to make an accurate diagnosis of the problem).

The test materials can be mixed to create new tests. If we have a test material that detects diseases A and B; and there is another test material that detects diseases B and C, then they can be mixed toobtain a test that detects diseases A, B, and C. This means that if we have these two test materials, then there is no need for a test material that tests diseases A, B, and C-such a material can be obtained bymixing these two.

Producing, distributing, and storing many different types of test materials is very expensive, and inmost cases, unnecessary. Your task is to eliminate as many unnecessary test materials as possible. Ithas to be done in such a way that if a test material is eliminated, then it should be possible to mix an equivalent test from the remaining materials. ("Equivalent" means that the mix tests exactly the samediseases as the eliminated material, not more, not less).

Input

The input contains several blocks of test cases. Each case begins with a line containing two integers: the number 1 <= n <= 300 of diseases, and the number 1 <= m <= 200 of test materials. The next m lines correspond to the m test materials. Each line begins with an integer, the number 1 <= k <= 300 of diseases that the material can detect. This is followed by k integers describing the k diseases. These integers are between 1 and n .

The input is terminated by a block with n = m = 0 .

Output

For each test case, you have to output a line containing a single integer: the maximum number of test materials that can be eliminated.

Example

Input:

```
10 5
2 1 2
2 2 3
3 1 2 3
4 1 2 3 4
1 4
3 7
1 1
1 2
1 3
2 1 2
2 1 3
2 3 2
3 1 2 3
0 0
```

Output:

```
2
4
```

Warning: large input/output data, be careful with certain languages

---

Added by: Blue Mary
Date: 2008-07-24
Time limit: 10s
Source limit: 50000B
Languages: All except: C99 strict
Resource: ACM Central European Programming Contest, Budapest 2005

SPOJ Problem Set (classical)

2880. Wild West

Problem code: WILD

Once upon a time in the west... The quiet life of the villages on the western frontier are often stirred up by the appearance of mysterious strangers. A stranger might be a bounty hunter looking for a notorious villain, or he might be a dangerous criminal escaping the hand of justice. The number of strangers has become so large that they formed the Mysterious Strangers' Union. If you want to be a mysterious stranger, then you have to apply to the Union, and you have to pass three exams that test the three most important skills: shooting, fist-fighting, and harmonica playing. For each skill, the Admission Committeegives a score between 1 (worst) and m (best). Interestingly enough, there are no two members in the Union having exactly the same skills: for every two member, there is always at least one skill for which they have diofferent scores. Furthermore, it turns out that for every possible combination of scores there is exactly one member having these scores. This means that there are exactly m^3 strangers in the union.

Recently, some members left the Union and they formed the Society of Evil Mysterious Strangers. The aim of this group is to commit as many evil crimes as possible, and they are quite successful at it. Therefore, the Steering Committee of the Union decided that a Hero is needed who will destroy this evil society. A Hero is a mysterious stranger who can defeat every member of the Society of Evil Mysterious Strangers. A Hero can defeat a member if the Hero has a higher score in at least one skill. For example, if the evil society has two members, Colonel Bill, with a score of 7 for shooting, 5 for knife throwing and 3 for harmonica playing, and Rabid Jack, with a score 10 for shooting, 6 for knife throwing and 8 for harmonica playing, then a Hero with score 8 for shooting, 7 for knife throwing and 3 for harmonica playing can defeat both ofthem. However, someone with a score of 8 for shooting 6 for knife throwing and 8 for harmonica playing cannot be the Hero. Moreover, the Hero cannot be a member of the evil society.

Your task is to determine whether there is a member in the Union who can be the Hero. If so, thenyou have to count how many members are potential heroes.

Input

The input contains several blocks of test cases. Each block begins with a line containing two integers: the number 1 <= n <= 100000 of members in the Society of Evil Mysterious Strangers and the maximum value 2 <= m <= 100000 of the scores. The next n lines describe these members. Each line contains three integers between 1 and m: the scores for the three skills.

The input is terminated by a block with n = m = 0.

Output

For each test case, you have to output a single line containing the number of members in the Union who satisfy the requirements for becoming a Hero. If there is no such member, then output *0*. It can be assumed that the output is always at most 10^{18}.

Example

Input:

```
3 10
2 8 5
6 3 5
1 3 9
1 3
2 2 2
1 10000
2 2 2
0 0
```

Output:

```
848
19
999999999992
```

Warning: enormous input/output data, be careful with certain languages

Note: The input is too large, so we have 4 input files and the total time limit is 17s.

Added by: Blue Mary
Date: 2008-07-24
Time limit: 17s
Source limit: 50000B
Languages: All except: C99 strict
Resource: ACM Central European Programming Contest, Budapest 2005

SPOJ Problem Set (classical)

2881. Find the Clones

Problem code: CLONE

Doubleville, a small town in Texas, was attacked by the aliens. They have abducted some of the residents and taken them to the a spaceship orbiting around earth. After some (quite unpleasant) human experiments, the aliens cloned the victims, and released multiple copies of them back in Doubleville. So now it might happen that there are 6 identical person named Hugh F. Bumblebee: the original person and its 5 copies. The Federal Bureau of Unauthorized Cloning (FBUC) charged you with the task of determining how many copies were made from each person. To help you in your task, FBUC have collected a DNA sample from each person. All copies of the same person have the same DNA sequence, and different people have different sequences (we know that there are no identical twins in the town, this is not an issue).

Input

The input contains several blocks of test cases. Each case begins with a line containing two integers: the number 1 <= n <= 20000 people, and the length 1 <= m <= 20 of the DNA sequences. The next n lines contain the DNA sequences: each line contains a sequence of m characters, where each character is either 'A', 'C', 'G' or 'T'.

The input is terminated by a block with n = m = 0 .

Output

For each test case, you have to output n lines, each line containing a single integer. The first line contains the number of different people that were not copied. The second line contains the number of people that were copied only once (i.e., there are two identical copies for each such person.) The third line contains the number of people that are present in three identical copies, and so on: the i -th line contains the number of persons that are present in i identical copies. For example, if there are 11 samples, one of them is from John Smith, and all the others are from copies of Joe Foobar, then you have to print '1' in the first andthe tenth lines, and '0' in all the other lines.

Example

Input:

```
9 6
AAAAAA
ACACAC
GTTTTG
ACACAC
GTTTTG
ACACAC
ACACAC
TCCCCC
TCCCCC
0 0
```

Output:

```
1
2
0
1
0
0
0
0
0
```

Warning: large input/output data, be careful with certain languages

---

Added by: Blue Mary
Date: 2008-07-24
Time limit: 10s
Source limit:50000B
Languages: All except: C99 strict
Resource: ACM Central European Programming Contest, Budapest 2005

SPOJ Problem Set (classical)

2882. The Warehouse

Problem code: WARE

Secret Agent OmeGa-7 has found the secret weapon warehouse of the mad scientist Dr. Matroid. The warehouse is full of large boxes (possibly with deadly weapons inside the boxes). While inspecting the warehouse, OmeGa-7 accidentally triggered the alarm system. The warehouse has a very effective protection against intruders: if the alarm is triggered, then the floor is filled with deadly acid. Therefore, the only way OmeGa-7 can escape is to climb onto the boxes and somehow reach the exit on top of them. The exit is a hole in the ceiling,if OmeGa-7 can climb through this hole then he can escape using the helicopter parked on the roof. There is a ladder and a box below the hole, thus the goal is to reach this box.

The floor of the warehouse can be divided into a grid containing n * n cells, the size of each cell is 1 meter * 1 meter. Each cell is either fully occupied by one box or unoccupied. Each box is rectangular: the size of the base is 1 meter * 1 meter, and the height is either 2, 3, or 4 meters. In figure (a), you can see an example warehouse, where the numbers show the height of the boxes, E shows the exit, and the circle shows that Secret Agent OmeGa-7 is currently on the top of that box.

[IMAGE]

OmeGa-7 can do two things:

If he is standing on top of a box, and in an adjacent cell there is another box, then he can move to the top of this other box. For example, in the situation depicted in figure (a), he can move either to north or east, but not to west or south. Note that only these four directions are allowed, diagonal moves are not possible. The height difference between the two boxes does not matter.

The second thing OmeGa-7 can do is that he can topple the box he is standing on in one of the four directions. The effect of toppling is best shown by an example: in the situation shown in figure (b), he can topple the box west (figure (c)) or north (figure (d)). If a box of height h is toppled north (west, south, etc.) then it will occupy h consecutive cells to the north (west, south, etc.) of its original position. The original position will be unoccupied (but can be later occupied again by toppling another box). A box can only be toppled if the cells where it will fall are unoccupied. For example, in figure (a), the box where OmeGa-7 is standing cannot be toppled in any of the four directions.

By toppling a box, OmeGa-7 jumps one step in the direction that the box is toppled (see figures (c) and(d)). If a box is toppled, then it cannot be toppled again later. Recall that there is a box below the exit (at the cell marked with E in the figure), thus it is not possible to topple a box over this cell. The alarm system will soon release mutant poisonous biting bats, so OmeGa-7 has to leave the warehouse as quickly as possible. You have to help him by writing a program that will determine the minimum number of steps required to reach the exit. Moving to an adjacent box, or toppling a box is counted as one step.

Input

The input contains several blocks of test cases. The first line of each block contains three integers: the size 1 <= n <= 8 of the warehouse, and two integers i , j that describe the starting position of the secret agent. These numbers are between 1 and n ; the row number is given by i , the column number is given by j . The next n lines describe the warehouse. Each line contains a string of n characters. Each character corresponds to a cell of the warehouse. If the character is '.', then the cell is unoccupied. The characters '2', '3' and '4' correspond to boxes of height 2, 3 and 4, respectively. Finally, the character 'E' shows the location of the exit.

The input is terminated by a block with n = i = j = 0.

Output

For each test case, you have to output a single line containing an integer: the minimum number of steps required to reach the exit. If it is not possible to reach the exit, then output the text 'Impossible.'(without quotes).

Example

Input:

```
5 5 3
.2..E
...2.
4....
....4
..2..
0 0 0
```

Output:

```
18
```

Added by: Blue Mary
Date: 2008-07-24
Time limit: 1s
Source limit:50000B
Languages: All except: C99 strict
Resource: ACM Central European Programming Contest, Budapest 2005

SPOJ Problem Set (classical)

2883. Widget Factory

Problem code: WIDGET

The widget factory produces several different kinds of widgets. Each widget is carefully built by a skilled widgeteer. The time required to build a widget depends on its type: the simple widgets need only 3 days, but the most complex ones may need as many as 9 days.

The factory is currently in a state of complete chaos: recently, the factory has been bought by a new owner, and the new director has fired almost everyone. The new staff know almost nothing about building widgets, and it seems that no one remembers how many days are required to build each diofferent type of widget. This is very embarrassing when a client orders widgets and the factory cannot tell the client how many days are needed to produce the required goods. Fortunately, there are records that say for each widgeteer the date when he started working at the factory, the date when he was fired and what types of widgets he built. The problem is that the record does not say the exact date of starting and leaving the job, only the day of the week. Nevertheless, even this information might be helpful in certain cases: for example, if a widgeteer started working on a Tuesday, built a Type 41 widget, and was fired on a Friday,then we know that it takes 4 days to build a Type 41 widget. Your task is to figure out from these records (if possible) the number of days that are required to build the different types of widgets.

Input

The input contains several blocks of test cases. Each case begins with a line containing two integers: the number 1 <= n <= 300 of the different types, and the number 1 <= m <= 300 of the records. This line is followed by a description of the m records. Each record is described by two lines. The first line contains the total number 1 <= k <= 10000 of widgets built by this widgeteer, followed by the day of week when he/she started working and the day of the week he/she was fired. The days of the week are given bythe strings 'MON', 'TUE', 'WED', 'THU', 'FRI', 'SAT' and 'SUN'. The second line contains k integers separated by spaces. These numbers are between 1 and n , and they describe the diofferent types of widgets that the widgeteer built. For example, the following two lines mean that the widgeteer started working on a Wednesday, built a Type 13 widget, a Type 18 widget, a Type 1 widget, again a Type 13 widget,and was fired on a Sunday.

```
4 WED SUN
13 18 1 13
```

Note that the widgeteers work 7 days a week, and they were working on every day between their first and last day at the factory (if you like weekends and holidays, then do not become a widgeteer!).

The input is terminated by a test case with n = m = 0 .

Output

For each test case, you have to output a single line containing n integers separated by spaces: the number of days required to build the different types of widgets. There should be no space before the first number or after the last number, and there should be exactly one space between two numbers. If there is more than one possible solution for the problem, then write 'Multiple solutions.' (without the quotes). If you are sure that there is no solution consistent with the input, then write 'Inconsistent data.'(without the quotes).

Example

Input:

```
2 3
2 MON THU
1 2
3 MON FRI
1 1 2
3 MON SUN
1 2 2
10 2
1 MON TUE
3
1 MON WED
3
0 0
```

Output:

```
8 3
Inconsistent data.
```

Warning: large input/output data, be careful with certain languages

---

Added by: Blue Mary
Date: 2008-07-24
Time limit: 10s
Source limit:50000B
Languages: All except: C99 strict
Resource: ACM Central European Programming Contest, Budapest 2005

SPOJ Problem Set (classical)

2884. Martian Mining

Problem code: MARTIAN

The NASA Space Center, Houston, is less than 200 miles from San Antonio, Texas (the site of the ACM Finals this year). This is the place where the astronauts are trained for Mission Seven Dwarfs, the next giant leap in space exploration. The Mars Odyssey program revealed that the surface of Mars is very rich in yeyenum and bloggium. These minerals are important ingredients for certain revolutionary new medicines, but they are extremely rare on Earth. The aim of Mission Seven Dwarfs is to mine these minerals on Mars and bring them back to Earth.

The Mars Odyssey orbiter identified a rectangular area on the surface of Mars that is rich in minerals. The area is divided into cells that form a matrix of n rows and m columns, where the rows go from east to west and the columns go from north to south. The orbiter determined the amount of yeyenum and bloggium in each cell. The astronauts will build a yeyenum refinement factory west of the rectangular area and a bloggium factory to the north. Your task is to design the conveyor belt system that will allow them to mine the largest amount of minerals.

There are two types of conveyor belts: the first moves minerals from east to west, the second moves minerals from south to north. In each cell you can build either type of conveyor belt, but you cannot build both of them in the same cell. If two conveyor belts of the same type are next to each other, then they can be connected. For example, the bloggium mined at a cell can be transported to the bloggium refinement factory via a series of south-north conveyor belts.

The minerals are very unstable, thus they have to be brought to the factories on a straight path without any turns. This means that if there is a south-north conveyor belt in a cell, but the cell north of it contains an east-west conveyor belt, then any mineral transported on the south-north conveyor beltwill be lost. The minerals mined in a particular cell have to be put on a conveyor belt immediately, in the same cell (thus they cannot start the transportation in an adjacent cell). Furthermore, any bloggium transported to the yeyenum refinement factory will be lost, and vice versa.

[IMAGE]

Your program has to design a conveyor belt system that maximizes the total amount of minerals mined,i.e., the sum of the amount of yeyenum transported to the yeyenum refinery and the amount of bloggium transported to the bloggium refinery.

Input

The input contains several blocks of test cases. Each case begins with a line containing two integers: the number 1 <= n <= 500 of rows, and the number 1 <= m <= 500 of columns. The next n lines describe the amount of yeyenum that can be found in the cells. Each of these n lines contains m integers. The first line corresponds to the northernmost row; the first integer of each line corresponds to the westernmost cell of the row. The integers are between 0 and 1000. The next n lines describe in a similar fashion theamount of bloggium found in the cells.

The input is terminated by a block with n = m = 0.

Output

For each test case, you have to output a single integer on a separate line: the maximum amount of mineralsthat can be mined.

Example

Input:

```
4 4
0 0 10 9
1 3 10 0
4 2 1 3
1 1 20 0
10 0 0 0
1 1 1 30
0 0 5 5
5 10 10 10
0 0
```

Output

```
98
```

Warning: large input/output data, be careful with certain languages

---

Added by: Blue Mary
Date: 2008-07-24
Time limit: 10s
Source limit:50000B
Languages: All except: C99 strict
Resource: ACM Central European Programming Contest, Budapest 2005

SPOJ Problem Set (classical)

2885. Word Rings

Problem code: WORDRING

A word ring is a sequence of words where the last two letters of each word are the same as the first two letters of the next word (and the last two letters of the last word are the same as the first two letters of the first word). For example, the following sequence is a word ring:

```
intercommunicational
alkylbenzenesulfonate
tetraiodophenolphthalein
```

Your task is to write a program that, given a list of words, finds a word ring. You have to make the word ring as impressive as possible: the average length of the words in the ring has to be as large as possible. In the above example, the average length is (20 + 21 + 24)/3 = 21.6666 , which makes it somewhat impressive. Note that each word can be used at most once in the ring, and the ring can consist of a single word.

Input

The input contains several blocks of test cases. Each case begins with a line containing a single integer 1 <= n <= 100000 , the number of possible words that can be used. The next n lines contain these words. The words contain only the characters 'a'-'z' and the length of each word is at most 1000.

The input is terminated by a block with n = 0.

Output

For each test case in the input, you have to output a single number on a separate line: the maximum average length of a ring composed from (a subset of) the words given in the input. The average length should be presented as a real number with two digits of precision. If it is not possible to compose a ring from these words, then output 'No solution.' (without quotes). To avoid rounding problems, we accept solutions with a maximum of 0.01(positive or negative) error.

Example

Input:

```
3
intercommunicational
alkylbenzenesulfonate
tetraiodophenolphthalein
0
```

Output:

```
21.66
```

Warning: large input/output data, be careful with certain languages

---

Added by:	Blue Mary
Date:	2008-07-24
Time limit:	13s
Source limit:	50000B
Languages:	All except: C99 strict
Resource:	ACM Central European Programming Contest, Budapest 2005

SPOJ Problem Set (classical)

2898. Party of Cloaked Killers

Problem code: PARTY2

N (1<= *N* <=100000) perfect killers (we number them 1, 2, 3, ..., *N*) meet at Blue Mary's house. Every killer has a kind of skill - cloak. No one can see them when they are cloaked - except only a small group of people, which will be discussed later.

We can group these killers into *M* (*M* >=3) groups, called group No.1, group No.2, group No.3, etc. If killer A is in group No. *x* and killer B is in group No. (*X*%*M*+1), A can see B even if B is cloaked. This prevent killers from doing some bad things without the risk of being punished.

To keep their identity secret, every killer keep cloaked during the party. After the party, Blue Mary asked everyone a question, "Which killers can you see in the party?" Although some killers forget some person they have ever seen during the party, Blue Mary collects extremely much information. Now she needs you help to determine the value of *M*, because no killer is willing to share this value with her.

Input

Ten test cases(given one after another, you have to process all!). For each test case:

The first line contains two integers *N* and *E*(1<= *E*<= 180000). E lines follow, each line contains two space-seperated integers A and B - killer No. A can see killer No.B even if he is cloaked.

Output

For each test case, output one line:

If the information given is contradictory, output one line "-1 -1". Otherwise output the largest and the smallest possible value of *M*, seperated by a single space.

Example

Input:

```
6 5
1 2
2 3
3 4
4 1
3 5
3 3
1 2
2 1
2 3
[and 8 test cases more]
```

Output:

```
4 4
-1 -1
[and 8 test cases more]
```

Warning: large input/output data, be careful with certain languages

---

Added by: Blue Mary
Date: 2008-07-30
Time limit: 3s
Source limit: 50000B
Languages: All except: C99 strict
Resource: Chinese National Olympiad in Informatics 2008, Day 1; Translate by Blue Mary

SPOJ Problem Set (classical)

2899. Volunteers

Problem code: VOL

ACM ICPC World Finals 2009, sponsored by IBM and hosted by KTH, Royal Institute of Technology will be held in Stockholm, Sweden. This contest will last for N(1<= N <= 1000) days. We need at least A_i volunteers in the i-th day. Now there are M(1<= M <=10000) kind of volunteers. The i-th type of volunteers will work from S_i-th day to T_i-th day, we will pay them \$$C_i$. Now your task is to minimize the money KTH pay for all the volunteers.

Input

Ten test cases(given one after another, you have to process all!). For each test case:

The first line contains two space-seperated integers N and M. The second line contains N nonnegative integers A_i. M lines follow, each contains three integers S_i, T_i and C_i. You may assume you can hire almost unlimited number of every type of volunteers.

Tip: During your calculation, **int** in C/C++/Java or **longint** in Pascal is enough.

Output

For each test case:

Output one line with an integer - the minimum cost.

Example

Input:

```
3 3
2 3 4
1 2 2
2 3 5
3 3 2
[and 9 test cases more]
```

Output:

```
14
[and 9 test cases more]
```

Added by: Blue Mary
Date: 2008-07-30
Time limit: 13s
Source limit:50000B
Languages: All except: C99 strict
Resource: Chinese National Olympiad in Informatics 2008, Day 1

SPOJ Problem Set (classical)

2901. One Geometry Problem

Problem code: GEOPROB

This is a problem of Euclidean Geometry. See the figure below.

[IMAGE]

Your task is as follows: given the lengths of segments b, c, and d, calculate the length of segment a.

Input

The input begins with the number t of test cases in a single line (t<=300). In each of the next t lines there are three integers b, c and d (0 <= b, c and d <= 10^{200} ; b,d < c) separated by a space.

Output

For every test case print the length of the segment a, one number per line.

Example

Input:
```
2
3 8 5
10 18 12
```

Output:
```
8
14
```

Added by: Frank Rafael Arteaga
Date: 2008-07-30
Time limit: 0.5s-1s
Source limit:4000B
Languages: All
Resource: My own Resource

SPOJ Problem Set (classical)

2903. Transportation

Problem code: TRANSP1

Blue Mary, the queen of Protoss, is planning a war against Zerg. Before the war she plans to make her base as safe as possible. Now there are N ($1 <= N <= 60$) nexuses available in the region controlled by Protoss, numbered 1, 2, ..., N. (Those who don't know what nexus is, please visit Blizzard Entertainment.) All the mineral and vespene gas stored in nexus i can be transported directly to nexus S_i.(i and S_i won't be the same.) Blue Mary's base is nexus 1, So all the mineral and vespene gas can be transported to base 1 directly or indirectly.

Blue Mary defines the safety of nexus i, **R(i)**, as the following:

C_i and k are numeral constants which will be given in the input file.

Suppose for a fixed i, set $T=\{P_1, P_2, P_3, ..., P_w\}$, then **x** is a member of T if and only if S_x is i. Any two P_js must be different.

Now Blue Mary wants to modify at most M ($1 <= M <= N$) S_i s, so that the safety of her base **R(1)** is maximized. To be a terran captive, also a great programmer, you must help her to solve this problem. Price is your life. Be careful! Blue Mary tells you that S_1 can't be modified. Don't ask your queen about the reason please.

Input

Ten test cases(given one after another, you have to process all!). For each test case:

The first line contains N, M and a real number k ($0.3 <= k <1$). The second line contains N space seperated integers S_i. The third line contains N positive real numbers C_i.

There is a single blank line between consecutive test cases.

Output

For each test case:

A single line - the maximized safety of nexus 1, rounded to two decimal places.

Example

Input:

```
4 1 0.5
2 3 1 3
10.0 10.0 10.0 10.0
```

```
[and 9 test cases more]
```

Output:

```
30.00
[and 9 test cases more]
```

Hint

Before modifying, the safety of the 4 bases are 22.8571, 21.4286,25.7143,10, respectively.

After modifying S_2 to 1, the safety of the 4 bases are 30, 25, 15, 10, respectively.

Added by: Blue Mary
Date: 2008-08-01
Time limit: 13s
Source limit:50000B
Languages: All except: C99 strict
Resource: Chinese National Olympiad in Informatics 2008, Day 2

SPOJ Problem Set (classical)

2905. Not a Triangle

Problem code: NOTATRI

You have N (3 <= N <= 2,000) wooden sticks, which are labeled from 1 to N. The i-th stick has a length of L_i (1 <= L_i <= 1,000,000). Your friend has challenged you to a simple game: you will pick three sticks at random, and if your friend can form a triangle with them (degenerate triangles included), he wins; otherwise, you win. You are not sure if your friend is trying to trick you, so you would like to determine your chances of winning by computing the number of ways you could choose three sticks (regardless of order) such that it is impossible to form a triangle with them.

Input

The input file consists of multiple test cases. Each test case starts with the single integer N, followed by a line with the integers L_1, ..., L_N. The input is terminated with N = 0, which should not be processed.

Output

For each test case, output a single line containing the number of triples.

Example

Input:

```
3
4 2 10
3
1 2 3
4
5 2 9 6
0
```

Output:

```
1
0
2
```

For the first test case, 4 + 2 < 10, so you will win with the one available triple. For the second case, 1 + 2 is equal to 3; since degenerate triangles are allowed, the answer is 0.

Added by: Neal Wu
Date: 2008-08-03
Time limit: 2s
Source limit:50000B
Languages: All

SPOJ Problem Set (classical)

2906. GCD2

Problem code: GCD2

Frank explained its friend Felman the algorithm of Euclides to calculate the GCD of two numbers. Then Felman implements it algorithm

```
int gcd(int a, int b)
{
        if (b==0)
                return a;
        else
                return gcd(b,a%b);
}
```

and it proposes to Frank that makes it but with a little integer and another integer that has up to 250 digits.

Your task is to help Frank programming an efficient code for the challenge of Felman.

Input

The first line of the input file contains a number representing the number of lines to follow. Each line consists of two number A and B (0 <= A <= 40000 and A <= B < 10^250).

Output

Print for each pair (A,B) in the input one integer representing the GCD of A and B.

Example

Input:

```
2
2 6
10 11
```

Output:

```
2
1
```

Added by: Frank Rafael Arteaga
Date: 2008-08-04
Time limit: 0.5s
Source limit: 1000B
Languages: All except: C99 strict JAVA PERL PYTH RUBY PHP LISP sbcl LISP clisp
Resource: My own Resource

SPOJ Problem Set (classical)

2916. Can you answer these queries V

Problem code: GSS5

You are given a sequence A[1], A[2], ..., A[N] . (|A[i]| <= 10000 , 1 <= N <= 10000). A query is defined as follows: Query(x1,y1,x2,y2) = Max { A[i]+A[i+1]+...+A[j] ; x1 <= i <= y1 , x2 j <= y2 and x1 <= x2 , y1 <= y2 }. Given M queries (1 <= M <= 10000), your program must output the results of these queries.

Input

The first line of the input consist of the number of tests cases <= 5. Each case consist of the integer N and the sequence A. Then the integer M. M lines follow, contains 4 numbers x1, y1, x2 y2.

Output

Your program should output the results of the M queries for each test case, one query per line.

Example

Input:

```
2
6 3 -2 1 -4 5 2
2
1 1 2 3
1 3 2 5
1 1
1
1 1 1 1
```

Output:

```
2
3
1
```

Added by:	Frank Rafael Arteaga
Date:	2008-08-06
Time limit:	1s
Source limit:	50000B
Languages:	All except: C99 strict
Resource:	K.-Y. Chen and K.-M. Chao, On the Range Maximum-Sum Segment Query Problem, 2007.

SPOJ Problem Set (tutorial)

2939. Query on a tree V

Problem code: QTREE5

You are given a tree (an acyclic undirected connected graph) with N nodes. The tree nodes are numbered from 1 to N. We define dist(a, b) as the number of edges on the path from node a to node b.

Each node has a color, white or black. All the nodes are black initially.

We will ask you to perfrom some instructions of the following form:

- 0 i : change the color of i-th node(from black to white, or from white to black).
- 1 **v** : ask for the minimum dist(u, **v**), node u must be white(u can be equal to **v**). Obviously, as long as node **v** is white, the result will alway be 0.

Input

- In the first line there is an integer N (N <= 100000)
- In the next N-1 lines, the i-th line describes the i-th edge: a line with two integers a b denotes an edge between a and b.
- In the next line, there is an integer Q denotes the number of instructions (Q <= 100000)
- In the next Q lines, each line contains an instruction "0 i" or "1 **v**"

Output

For each "1 **v**" operation, print one integer representing its result. If there is no white node in the tree, you should write "**-1**".

Example

Input:

```
10
1 2
1 3
2 4
1 5
1 6
4 7
7 8
5 9
1 10
10
0 6
0 6
0 6
1 3
0 1
0 1
1 3
1 10
```

```
1 4
1 6
```

Output:

```
2
2
2
3
0
```

Added by: Qu Jun
Date: 2008-08-13
Time limit: 6s
Source limit: 50000B
Languages: All
Resource: XunYunbo, modified from ZJOI07

SPOJ Problem Set (classical)

2940. Untitled Problem II

Problem code: UNTITLE1

You are given a sequence of N integers A_1, A_2 .. A_N. (-10000 <= A_i <= 10000, N <= 50000)

Let S_i denote the sum of A_1..A_i. You need to apply M (M <= 50000) operations:

- 0 x y k: increase all integers from A_x to A_y by k(1 <= x <= y <= N, -10000 <= k <= 10000).
- 1 x y: ask for max{ S_i | x <= i <= y }.(1 <= x <= y <= N)

Input

- In the first line there is an integer N.
- The following line contains N integers that represent the sequence.
- The third line contains an integer M denotes the number of operations.
- In the next M lines, each line contains an operation "0 x y k" or "1 x y".

Output

For each "1 x y" operation, print one integer representing its result.

Example

Input:

```
5
238 -9622 5181 202 -6943
5
1 3 4
0 5 5 4846
1 3 5
0 3 5 -7471
1 3 3
```

Output:

```
-4001
-4001
-11674
```

Use signed 64-bit integer :)

Added by:	**Qu Jun**
Date:	**2008-08-14**
Time limit:	**4s**
Source limit:	**50000B**
Languages:	**All**
Resource:	**Not an own problem**

SPOJ Problem Set (classical)

2944. Emmons

Problem code: SHOOTING

After the end of all the shooting competitions in XXIX Olympic Games in Beijing, Matthew Emmons will be known to more and more people because of his last - which is also his worst - shooting in the 50m Rifle 3*40 Men competitions. Four years before in Athens, he shot a wrong target and lost the gold metal which is almost at hands in the 50m Rifle 3*40 Men competition.

The following is Blue Mary's imagination :P

Emmons decides to practise shooting more assiduously. Because he is an excellent shooter, only 1 year later, he can even shoot precisely without collimation! To him, getting the gold metal of 50m Rifle 3*40 Men in XXX Olympic Games doesn't have any difficulty now.

His wife - Katerina Emmons, also a well-known excellent shooter - make a game to keep his interests with shot. The player has n (1<= n <=2000) bullets, each one has a value (a integer whose absolute value is less than 10000). There are m (1<= m <= n) targets, each with a point counter next to it. In the beginning of the game, all the counter are set to integer 1.

During the game, the player must choose a bullet and shoot any target. He must use all the bullet, each with at least(of course, at most) 1 time. And each target must be shot at least one time.

If the player shoot a target with a bullet valued **X**, the counter of the target will multiplied by **X**.

The final score of the game is sum of all the m counters.

Now Matthew needs your help to make his final score as high as possible. After that, he will show you his excellent shooting skills to get this score.

P.S. Even the things above is my imagination, I hope Matthew Emmons has good luck and wins the gold metal of 50m Rifle 3*40 Men in XXX Olympics in London.

Input

Multiple test cases, the number of them (<=50) is given in the very first line.

For each test case:

The first line contains two integers n and m. The second line contains n integers, the value for each bullet.

Output

For each test case:

The first and the only line contains a single integer - the highest possible final score.

Example

Input:

```
3
10 2
0 -1 -2 0 1 2 3 2 10 1
10 3
0 -1 -2 0 1 2 3 2 10 1
5 3
10 0 0 -1 -1
```

Output:

```
240
241
11
```

Hint:

```
For the first example, a possible solution is (0,0)(-1,-2,1,2,3,2,10,1).
For the second example, a possible solution is (0,0)(1,1)(-1,-2,2,3,2,10).
```

Added by: Blue Mary
Date: 2008-08-21
Time limit: 60s
Source limit: 50000B
Languages: All except: C99 strict
Resource: Chinese Team Selection Contest for IOI 2006, with description modified

SPOJ Problem Set (classical)

2946. Eclipse

Problem code: ECLIPSE

Every so often we hear on the news that there is going to be either a solar or lunar eclipse. Eclipses have a long history dating back well into the BC's. Astronomers study total solar eclipses very closely as they provide the rare opportunity to observe the corona.

An eclipse occurs when two celestial bodies and a star are (nearly) linearly aligned and the shadow cast by the one body intersects the other body, creating darkness on the latter body.

We are interested in determining when a solar eclipse will next occur. In Figure 1 you can see two labelled regions. The umbra is the area of total darkness -- a body in this region will experience a total solar eclipse. The penumbra is the area of partial darkness -- a body in this region will experience a partial solar eclipse.

You will be given the size and location of a star and two celestial bodies. Your task is to determine if the first celestial body creates a solar eclipse on the second celestial body. If it does then you are to determine whether it is a total or partial eclipse and whether the entire body is in eclipse. If part of the body is experiencing total eclipse while the entire body is experiencing at least a partial eclipse, we are only interested in the part that is in total eclipse.

Consider a scaled model of our solar system with the sun at the origin (0, 0, 0) with radius 700, the moon at position (49900, 1000, 149700) with radius 2 and Earth at position (50000, 1000, 150000) with radius 7. In Figure 1, the sun would be the star on the left and the moon would be the smaller body on the right. Part of Earth would then fall in the black umbra region and hence partly experience a total solar eclipse.

For any body:

- $1 <= r <= 10^6$
- $0 <= x, y, z <= 10^9$

It is guaranteed that any two bodies will be at least 1 unit apart, and that moving any one of the bodies by 1 unit (in any direction) will not change the answer.

[IMAGE]

Input

A test case is described by three lines, each describing the size and location of a single body. The first line contains four space-separated integers xs , ys , zs and rs , describing the center (xs , ys , zs) and radius rs of the star. The following two lines define the two celestial bodies in the same manner.

Test cases follow directly after one another with a -1 representing the end of the test cases.

Output

Each test case has a single line of output describing the type of eclipse for that case. If the second celestial body listed in the test case is experiencing an eclipse, then one of the following lines must be output:

- Entire total solar eclipse
- Part total solar eclipse
- Entire partial solar eclipse
- Part partial solar eclipse

If there is no solar eclipse, the line "No solar eclipse" must be output.

Example

Input:

```
0 0 0 700
49900 1000 149700 2
50000 1000 150000 7
0 0 0 10
50 0 100 40
60 0 200 1
-1
```

Output:

```
Part total solar eclipse
Entire total solar eclipse
```

Added by: Marco Gallotta
Date: 2008-08-27
Time limit: 30s
Source limit: 50000B
Languages: All
Resource: ACM Southern African Regionals 2007

SPOJ Problem Set (classical)

2962. Painting Blocks (Act I)

Problem code: PAINTBLK

n blocks are put in a line. You have *k*(1<=*k*<=15) kinds of dope, the *i*-th dope is enough to paint c_i (1<=c_i<=5) blocks. You may assume the sum of all the c_i equals to *n*. Your task is to calculate the number of ways to paint the blocks with these kinds of dope, such that no two adjacent blocks are painted with the same kind of dope.

Input

Ten test cases(given one after another, you have to process all!). For each test case, the first line contains an integer *k*, the second line contains *k* integers, c_1, c_2, ...c_k.

Output

Ten lines, each contains an integer, the number of ways modulo 1000000007.

Example

Input:
```
3
1 2 3
5
2 2 2 2 2
10
1 1 2 2 3 3 4 4 5 5
[and 7 test cases more]
```

Output:
```
10
39480
85937576
[and 7 test cases more]
```

Added by: Blue Mary
Date: 2008-09-01
Time limit: 3s
Source limit:50000B
Languages: All except: C99 strict
Resource: :P

SPOJ Problem Set (classical)

2963. Painting Blocks (Act II)

Problem code: PAINTBLC

n blocks are put in a line. You have *k*(1<=*k*<=15) kinds of dope, the *i*-th dope is enough to paint c_i (1<=c_i<=6) blocks. You may assume the sum of all the c_i equals to *n*. Your task is to calculate the number of ways to paint the blocks with these kinds of dope, such that no two adjacent blocks are painted with the same kind of dope.

Input

Input consists of multiple test cases, the number of them (<=2000) is given in the very first line. For each test case, the first line contains an integer *k*, the second line contains *k* integers, c_1, c_2, ...c_k.

Output

For each test case, output one line with an integer, the number of ways modulo 1000000007.

Example

Input:

```
3
3
1 2 3
5
2 2 2 2 2
10
1 1 2 2 3 3 4 4 5 5
```

Output:

```
10
39480
85937576
```

Added by: Blue Mary
Date: 2008-09-01
Time limit: 1s
Source limit:50000B
Languages: All except: C99 strict
Resource: g201513

SPOJ Problem Set (classical)

3002. Electrophoretic

Problem code: ELECTRO

Scientist Frank, majoring in electrochemistry, has developed line-shaped strange electrodes called F-electrodes. During being activated, each F-electrode causes a special potential on and between the two lines touching the F-electrode's endpoints at a right angle. Then electrically-charged particles located inside the potential area get to move in the direction parallel to the potential boundary (i.e. perpendicular to the F-electrode), either toward or against F-electrode. The moving direction can be easily controlled between the two possibles; it is also possible to get particles to pass through F-electrodes. In addition, unlike ordinary electrodes, F-electrodes can affect particles even infinitely far away, as long as those particles are located inside the potential area. On the other hand, two different F-electrodes cannot be activated at a time, since their potentials conflict strongly.

We can move particles on our will by controlling F-electrodes. However, in some cases, we cannot lead them to the desired positions due to the potential areas being limited. To evaluate usefulness of F-electrodes from some aspect, Frank has asked you the following task: to write a program that finds the shortest distances from the particles' initial positions to their destinations with the given sets of F-electrodes.

[IMAGE]

Input

The input consists of multiple test cases. The first line of each case contains N(1 <= N <= 100) which represents the number of F-electrodes. The second line contains four integers xs, ys, xt and yt, where (xs, ys) and (xt , yt) indicate the particle's initial position and destination. Then the description of N F-electrodes follow. Each line contains four integers F_{xs}, F_{ys}, F_{xt} and F_{yt}, where (F_{xs}, F_{ys}) and (F_{xt} , F_{yt}) indicate the two endpoints of an F-electrode. All coordinate values range from 0 to 100 inclusive.

The input is terminated by a case with N = 0.

Output

Your program must output the case number followed by the shortest distance between the initial position to the destination. Output "`Impossible`" (without quotes) as the distance if it is impossible to lead the elementary particle to the destination. Your answers must be printed with five digits after the decimal point.

Example

Input:

```
2
2 1 2 2
0 0 1 0
0 1 0 2
```

```
0
```

Output:

```
Case 1: 3.00000
```

Added by:	Jin Bin
Date:	2008-09-08
Time limit:	10s
Source limit:	50000B
Languages:	All except: C99 strict
Resource:	JAG wintercamp 08, day2

SPOJ Problem Set (classical)

3003. Median Filter

Problem code: FILTER

The median filter is a nonlinear digital filter used to reduce noise in images, sounds, and other kinds of signals. It examines each sample of the input through a *window* and then emits the *median* of the samples in the window. Roughly speaking, a window is an interval that contains a target sample and its preceding and succeeding samples; the median of a series of values is given by the middle value of the series arranged in ascending (or descending) order.

Let us focus on a typical median filter for black-and-white raster images. The typical filter uses a 3 × 3 window, which contains a target pixel and the eight adjacent pixels. The filter examines each pixel in turn through this 3 × 3 window, and outputs the median of the nine pixel values, i.e. the fifth lowest (or highest) pixel value, to the corresponding pixel. We should note that the output is just given by the pixel value in majority for black-andwhite images, since there are only two possible pixel values (i.e. black and white). The figure below illustrates how the filter works.

[IMAGE]

The edges of images need to be specially processed due to lack of the adjacent pixels. In this problem, we extends the original images by repeating pixels on the edges as shown in the figure below. In other words, the lacked pixels take the same values as the nearest available pixels in the original images.

[IMAGE]

You are requested to write a program that reads images to which the filter is applied, then finds the original images containing the greatest and smallest number of black pixels among all possible ones, and reports the difference in the numbers of black pixels.

Input

The input contains a series of test cases.

The first line of each test case contains two integers W and H (1 <= W, H <= 8), which indicates the width and height of the image respectively. Then H lines follow to describe the filtered image. The *i*-th line represents the *i*-th scan line and contains exactly W characters, each of which is either '#' (representing black) or '.' (representing white).

The input is terminated by a line with two zeros.

Output

For each test case, print a line that contains the case number followed by the difference of black pixels. If there are no original images possible for the given filtered image, print "`Impossible`" instead.

Obey the format as shown in the sample output.

Example

Input:

```
5 5
#####
#####
#####
#####
#####
4 4
####
####
####
####
4 4
#...
....
....
...#
4 4
.#.#
#.#.
.#.#
#.#.
0 0
```

Output:

```
Case 1: 10
Case 2: 6
Case 3: 2
Case 4: Impossible
```

Added by: Jin Bin
Date: 2008-09-08
Time limit: 20s
Source limit: 50000B
Languages: All except: C99 strict
Resource: JAG wintercamp 08, day2

SPOJ Problem Set (classical)

3004. Life Game

Problem code: LIFEGAME

You are working at a production plant of biological weapons. You are a maintainer of a terrible virus weapon with very high reproductive power. The virus has a tendency to build up regular hexagonal colonies. So as a whole, the virus weapon forms a hexagonal grid, each hexagon being a colony of the virus. The grid itself is in the regular hexagonal form with *N* colonies on each edge.

The virus self-propagates at a constant speed. Self-propagation is performed simultaneously at all colonies. When it is done, for each colony, the same number of viruses are born at every neighboring colony. Note that, after the self-propagation, if the number of viruses in one colony is more than or equal to the limit density *M*, then the viruses in the colony start self-attacking, and the number reduces modulo *M*.

Your task is to calculate the total number of viruses after *L* periods, given the size *N* of the hexagonal grid and the initial number of viruses in each of the colonies.

[IMAGE]

Input

The input consists of multiple test cases.

Each case begins with a line containing three integers *N* (1 <= *N* <= 6), *M* (2 <= *M* <= 10^9), and *L* (1 <= *L* <= 10^9). The following 2*N* - 1 lines are the description of the initial state. Each non-negative integer (smaller than *M*) indicates the initial number of viruses in the colony. The first line contains the number of viruses in the *N* colonies on the topmost row from left to right, and the second line contains those of *N* + 1 colonies in the next row, and so on.

The end of the input is indicated by a line "0 0 0".

Output

For each test case, output the test case number followed by the total number of viruses in all colonies after *L* periods.

Example

Input:

```
3 3 1
1 0 0
0 0 0 0
0 0 0 0 0
0 0 0 0
0 0 1
3 3 2
1 0 0
0 0 0 0
```

```
0 0 0 0 0
0 0 0 0
0 0 1
0 0 0
```

Output:

```
Case 1: 8
Case 2: 18
```

Added by: Jin Bin
Date: 2008-09-08
Time limit: 20s
Source limit: 50000B
Languages: All except: C99 strict
Resource: JAG wintercamp 08, day2

SPOJ Problem Set (classical)

3005. Subdividing a Land

Problem code: LAND

Indigo Real-estate Company is now planning to develop a new housing complex. The entire complex is a square, all of whose edges are equally *a* meters. The complex contains *n* subdivided blocks, each of which is a *b*-meter square. Here both *a* and *b* are positive integers.

However the project is facing a big problem. In this country, a percentage limit applies to the subdivision of a land, under the pretext of environmental protection. When developing a complex, the total area of the subdivided blocks must not exceed 50% of the area of the complex; in other words, more than or equal to 50% of the newly developed housing complex must be kept for green space. As a business, a green space exceeding 50% of the total area is a *dead space*. The primary concern of the project is to minimize it.

Of course purchasing and developing a land costs in proportion to its area, so the company also wants to minimize the land area to develop as the secondary concern. You, a member of the project, were assigned this task, but can no longer stand struggling against the problem with your pencil and paper. So you decided to write a program to find the pair of minimum *a* and *b* among those which produce the minimum dead space for given *n*.

Input

The input consists of multiple test cases. Each test case comes in a line, which contains an integer *n*. You may assume 1 <= *n* <= 10000.

The end of input is indicated by a line containing a single zero. This line is not a part of the input and should not be processed.

Output

For each test case, output the case number starting from 1 and the pair of minimum *a* and *b* as in the sample output.

You may assume both *a* and *b* fit into 64-bit signed integers.

Example

Input:

```
1
2
0
```

Output:

```
Case 1: 3 2
Case 2: 2 1
```

Added by: Jin Bin
Date: 2008-09-08
Time limit: 10s
Source limit: 50000B
Languages: All except: C99 strict
Resource: JAG wintercamp 08, day2

SPOJ Problem Set (classical)

3006. Connect Line Segments

Problem code: LINE

Your dear son Arnie is addicted to a puzzle named *Connect Line Segments*.

In this puzzle, you are given several line segments placed on a two-dimensional area. You are allowed to add some new line segments each connecting the end points of two existing line segments. The objective is to form a single polyline, by connecting all given line segments, as short as possible. The resulting polyline is allowed to intersect itself.

[IMAGE]

Arnie has solved many instances by his own way, but he is wondering if his solutions are the best one. He knows you are a good programmer, so he asked you to write a computer program with which he can verify his solutions.

Please respond to your dear Arnie's request.

Input

The input consists of multiple test cases.

Each test case begins with a line containing a single integer n ($2 <= n <= 14$), which is the number of the initial line segments. The following n lines are the description of the line segments. The i-th line consists of four real numbers: $x_{i,1}$, $y_{i,1}$, $x_{i,2}$, and $y_{i,2}$ ($-100 <= x_{i,1}, y_{i,1}, x_{i,2}, y_{i,2} <= 100$). $(x_{i,1}, y_{i,1})$ and $(x_{i,2}, y_{i,2})$ are the coordinates of the end points of the i-th line segment.

The end of the input is indicated by a line with single "0".

Output

For each test case, output the case number followed by the minimum length in a line.

The output value should be printed with five digits after the decimal point.

Example

Input:

```
4
0 1 0 9
10 1 10 9
1 0 9 0
1 10 9 10
2
1.2 3.4 5.6 7.8
5.6 3.4 1.2 7.8
0
```

Output:

```
Case 1: 36.24264
Case 2: 16.84508
```

Added by:	Jin Bin
Date:	2008-09-08
Time limit:	10s
Source limit:	50000B
Languages:	All except: C99 strict
Resource:	JAG wintercamp 08, day2

SPOJ Problem Set (classical)

3007. Oil Company

Problem code: OILCOMP

Irving & Cohen Petroleum Corporation has decided to develop a new oil field in an area. A preliminary survey has been done and they created a detailed grid map of the area which indicates the reserve of oil.

They are now planning to construct mining plants on several grid blocks according this map, but they decided not to place any two plants on adjacent positions to avoid spreading of fire in case of blaze. Two blocks are considered to be adjacent when they have a common edge. You are one of the programmers working for the company and your task is to write a program which calculates the maximum amount of oil they can mine, given the map of the reserve.

Input

The first line of the input specifies N, the number of test cases. Then N test cases follow, each of which looks like the following:

```
W H
r1,1 r2,1 ... rW,1
 ...
r1,H r2,H ... rW,H
```

The first line of a test case contains two integers W and H (1 <= W, H <= 20). They specifies the dimension of the area. The next H lines, each of which contains W integers, represent the map of the area. Each integer rx,y (0 <= rx,y < 10000) indicates the oil reserve at the grid block (x, y).

Output

For each test case, output the case number (starting from 1) and the maximum possible amount of mining in a line. Refer to the sample output section about the format.

Example

Input:

```
2
2 2
2 3
3 5
3 2
4 1 1
2 1 4
```

Output:

```
Case 1: 7
Case 2: 8
```

Added by: Jin Bin
Date: 2008-09-08
Time limit: 10s
Source limit: 50000B
Languages: All except: C99 strict
Resource: JAG wintercamp 08, day2

SPOJ Problem Set (classical)

3008. Finding the Top RPS Player

Problem code: RPS

A company "ACM Foods" is preparing for opening its chain shop in a certain area, but another company "ICPC Pizza" is also planning to set up its branch shop in the same area. In general, two competitive shops gain less incomes if they are located so close to each other. Thus, if both "ACM Foods" and "ICPC Pizza" went on opening, they would be damaged financially. So, they had a discussion on this matter and made the following agreement: only one of them can branch its shop in the area. It is determined by Rock-Paper-Scissors (RPS) which to branch the shop.

ACM Foods is facing financial difficulties and strongly desires to open their new shop in that area. The executives have decided to make every effort for finding out a very strong RPS player. They believes that players who win consecutive victories must be strong players. In order to find such a player for sure, they have decided their simple strategy.

In this strategy, many players play games of RPS repeatedly, but the games are only played between players with the same number of consecutive wins. At the beginning, all the players have no wins, so any pair of players can play a game. The games can be played by an arbitrary number of pairs simultaneously. Let us call a set of simultaneous games as a turn. After the first turn, some players will have one win, and the other players will remain with no wins. In the second turn, some games will be played among players with one win, and some other games among players with no wins. For the former games, the winners will have two consecutive wins, and the losers will lose their first wins and have no consecutive wins. For the latter games, the winners will have one win, and the losers will remain with no wins. Therefore, after the second turn, the players will be divided into three groups: players with two consecutive wins, players with one win, and players with no wins. Again, in the third turn, games will be played among players with two wins, among with one win, and among with no wins. The following turns will be conducted so forth. After a sufficient number of turns, there should be a player with the desired number of consecutive wins.

The strategy looks crazy? Oh well, maybe they are confused because of their financial difficulties. Of course, this strategy requires an enormous amount of plays. The executives asked you, as an employee of ACM Foods, to estimate how long the strategy takes. Your task is to write a program to count the minimum number of turns required to find a player with M consecutive wins among N players.

Input

The input consists of multiple test cases. Each test case consists of two integers N (2 <= N <= 20) and M (1 <= M < N) in one line.

The input is terminated by the line containing two zeroes.

Output

For each test case, your program must output the case number followed by one integer which indicates the minimum number of turns required to find a person with M consecutive wins.

Example

Input:

```
2 1
10 5
15 10
0 0
```

Output:

```
Case 1: 1
Case 2: 11
Case 3: 210
```

Added by: Jin Bin
Date: 2008-09-08
Time limit: 10s
Source limit: 50000B
Languages: All except: C99 strict
Resource: JAG wintercamp 08, day2

SPOJ Problem Set (classical)

3009. Revenge of Voronoi

Problem code: VORONOI

A discrete Voronoi diagram is a derivation of a Voronoi diagram. It is represented as a set of pixels. Each of the generatrices lies on the center of some pixel. Each pixel belongs to the generatrix nearest from the center of the pixel in the sense of Manhattan distance. The Manhattan distance d between two points (x_1, y_1) and (x_2, y_2) is given by the following formula:

$$d = |x_1 - x_2| + |y_1 - y_2|$$

Your task is to find a set of generatrices which generates a given discrete Voronoi diagram. In the given diagram, each generatrix is given a unique lowercase letter as its identifier, and each pixel is represented by the identifier of the generatrix the pixel belongs to. If a pixel has multiple generatrices at the same distance from its center, it belongs to the generatrix with the most preceding identifier among them (i.e. the smallest character code).

Input

The input consists of multiple test cases.

Each test case begins with a line containing two integers W (1 <= W <= 32) and H (1 <= H <= 32), which denote the width and height of the discrete Voronoi diagram.

The following H lines, each of which consists of W letters, give one discrete Voronoi diagram. Each letter represents one pixel.

The end of input is indicated by a line with two zeros. This is not a part of any test cases.

Output

For each test case, print the case number and the coordinates of generatrices as shown in the sample output. Each generatrix line should consist of its identifier, x-coordinate, and y-coordinate. Generatrices should be printed in alphabetical order of the identifiers. Each coordinate is zero-based where (0, 0) indicates the center of the top-left corner pixel of the diagram.

You may assume that every test case has at least one solution. If there are multiple solutions, any one is acceptable.

Print a blank line after every test case including the last one.

Example

Input:

```
4 3
ooxx
ooxx
ooxx
```

```
4 1
null
4 4
aabb
aabb
ccdd
ccdd
0 0
```

Output:

```
Case 1:
o 0 0
x 2 0

Case 2:
l 2 0
n 0 0
u 1 0

Case 3:
a 0 0
b 2 0
c 0 2
d 2 2
```

Added by: Jin Bin
Date: 2008-09-08
Time limit: 10s
Source limit: 50000B
Languages: All except: C99 strict
Resource: JAG wintercamp 08, day2

SPOJ Problem Set (classical)

3010. Castle Wall

Problem code: WALL

A new lord assumed the position by the death of the previous lord in a Far Eastern province.

The new greedy lord hates concave polygons, because he believes they need much wasted area to be drawn on paper. He always wants to modify them to convex ones.

His castle is currently surrounded by a wall forming a concave polygon, when seen from the above. Of course he hates it. He believes more area could be obtained with a wall of a convex polygon. Thus he has ordered his vassals to have new walls built so they form a convex polygon.

Unfortunately, there is a limit in the budget. So it might be infeasible to have the new walls built completely. The vassals has found out that only up to *r* meters of walls in total can be built within the budget. In addition, the new walls must be built in such a way they connect the polygonal vertices of the present castle wall. It is impossible to build both of intersecting walls.

After long persuasion of the vassals, the new lord has reluctantly accepted that the new walls might not be built completely. However, the vassals still want to maximize the area enclosed with the present and new castle walls, so they can satisfy the lord as much as possible.

Your job is to write a program to calculate, for a given integer *r*, the maximum possible area of the castle with the new walls.

Input

The input file contains several test cases.

Each case begins with a line containing two positive integers *n* and *r*. *n* is the number of vertices of the concave polygon that describes the present castle wall, satisfying 5 <= *n* <= 64. *r* is the maximum total length of new castle walls feasible within the budget, satisfying 0 <= *r* <= 400.

The subsequent *n* lines are the *x*- and *y*-coordinates of the *n* vertices. The line segments (x_i, y_i)-(x_{i+1}, y_{i+1}) (1 <= *i* <= *n* - 1) and (x_n, y_n)-(x_1, y_1) form the present castle wall of the concave polygon. Those coordinates are given in meters and in the counterclockwise order of the vertices.

All coordinate values are integers between 0 and 100, inclusive. You can assume that the concave polygon is simple, that is, the present castle wall never crosses or touches itself.

The last test case is followed by a line containing two zeros.

Output

For each test case in the input, print the case number (beginning with 1) and the maximum possible area enclosed with the present and new castle walls. The area should be printed with exactly one fractional digit.

Example

Input:

```
5 4
0 0
4 0
4 4
2 2
0 4
8 80
45 41
70 31
86 61
72 64
80 79
40 80
8 94
28 22
0 0
```

Output:

```
Case 1: 16.0
Case 2: 3375.0
```

Added by: Jin Bin
Date: 2008-09-08
Time limit: 10s
Source limit: 50000B
Languages: All except: C99 strict
Resource: JAG wintercamp 08, day2

SPOJ Problem Set (classical)

3033. Help the soldier

Problem code: SOLDIER

Igor, a famous russian soldier, must go to war in Afghanistan (we are in late 80's). His superiors allowed him to buy himself his equipment. So, he must buy 6 items: helmet, bulletproof vest, trousers, boots, tunic and a firearm. This items are represented with numbers from 1 to 6. There are N(6 < N < 101) items of this 6 types. Each item is characterized by its price p[i] (in rublas) and is quality q[i]. Igor has T (0 < T < 1001) rublas and he wants to maximize the total quality of his equipment. The total quality is the quality of the item with the lowest quality. Help him.

Input

On the first line there are two integers N and T. On the lines 2 ... N+1 there are 3 integers, type[i] (from 1 to 6) p[i] and q[i]. (0 < p[i], q[i] < T)

Output

Output the total quality.

Example

Input:
```
7 53
5 8 2
2 4 8
6 8 13
1 13 12
4 5 1
3 2 7
3 13 5
```

Output:
```
1
```

Note:
```
If there is no answer, output 0.
There can be less than 6 types of items.
```

Added by: Pripoae Toni
Date: 2008-09-14
Time limit: 0.5s
Source limit: 2048B
Languages: All
Resource: Original

SPOJ Problem Set (classical)

3070. How many subsequences

Problem code: SEQ5

Tom has again maths , and the teacher writes countless tables with exercises .... so boring. Then he remembers an old problem of informatics that he thought in a dream . He remembered , he has a number of positive integers and the job was to know how many subsequences that have between L and U distinct elements exist in that range. So the boring hour will pass quicker . But he needs your help , he is to exhausted after two hours of math with the agitated teacher .

Input

The first line of input file contains positive N, L, U. following N lines will contain a positive integer, each representing an element of the series.

Output

The first line of the output will display the number of sequences containing between L and U distinct elements.

Example

Input:

```
4 1 2
231
19
7
19
```

Output:

```
8
```

Notes:

```
1<= L <= U <= N <= 2^20
The value of an item number is a positive integers [1,2^32-1];
A subsequence is a lot of items that appear on consecutive positions in the initial row.
Be carefull with certain languages.Large imput data.
Tom thanks you for solving this problem and he awards you with points.
```

Added by: Pripoae Toni
Date: 2008-09-29
Time limit: 0.200s-2.5s
Source limit:50000B
Languages: All
Resource: Mircea Pasoi

SPOJ Problem Set (classical)

3105. Power Modulo Inverted

Problem code: MOD

Given 3 positive integers x, y and z, you can find $k = x^y \% z$ easily, by fast power-modulo algorithm. Now your task is the inverse of this algorithm. Given 3 positive integers x, z and k, find the smallest non-negative integer y, such that $k\%z = x^y \% z$.

Input

About 600 test cases.

Each test case contains one line with 3 integers x, z and k.($1 <= x, z, k <= 10^9$)

Input terminates by three zeroes.

Output

For each test case, output one line with the answer, or "No Solution"(without quotes) if such an integer doesn't exist.

Example

Input:

```
5 58 33
2 4 3
0 0 0
```

Output:

```
9
No Solution
```

Added by: Blue Mary
Date: 2008-10-04
Time limit: 17s
Source limit:50000B
Languages: All except: C99 strict
Resource: Folklore, description, standard program and test data by Blue Mary

SPOJ Problem Set (classical)

3106. Dictionary Subsequences

Problem code: DICTSUB

You have a dictionary of strings, and you want to perform some queries on the strings. In particular, you're given a single string T, and for each word W in the dictionary, you want to determine if W is a subsequence of T. A string B is a subsequence of a string C if you can remove zero or more of C's letters to form a string equal to B (but the order of remaining letters may not be rearranged).
Each word W in the dictionary will be described in the input as a run length encoded (RLE) string. That is, W will be described by several pairs of data values, where each pair of data values consists of a positive integer K with no leading zeros and a letter L. A data pair with values K and L represents a string with K occurrences of the character L. To get the uncompressed string, we concatenate all strings represented by the data pairs. For example, the RLE string 2A1B5C12A represents the string AABCCCCCAAAAAAAAAAAA.

Input

The first line of the input contains a positive integer C (0<C<10), the number of test cases to follow. Each case begins with a line containing a positive integer D (0<D<10000) representing the number of dictionary words and a string T with length between 1 and 10000. D lines follow, with each line containing a string with length between 1 and 200 in RLE format, which represents a dictionary word with uncompressed length between 1 and 10000. All uncompressed strings (T and dictionary words) will consist only of uppercase letters ('A'-'Z').

Output

Output for each case consists of several lines. There should be one line per dictionary word W (in the order of appearance in input) that will say either "YES" if W is a subsequence of T, or "NO" otherwise. Print a blank line after each test case.

Example

Input:
```
1
5 EFFERVESCENCE
2E
1E1F1V1C1E
1E2F1C1R
1S2E
1P1E2F
```

Output:
```
YES
YES
NO
YES
NO
```

Added by:	John Rizzo
Date:	2008-10-04
Time limit:	2s
Source limit:	50000B
Languages:	All
Resource:	Al-Khawarizm 2008 - Set by eleusive

SPOJ Problem Set (classical)

3107. Odd Numbers of Divisors

Problem code: ODDDIV

Given a positive odd integer K and two positive integers low and high, determine how many integers between low and high contain exactly K divisors.

Input

The first line of the input contains a positive integer C (0<C<100,000), the number of test cases to follow. Each case consists of a line containing three integers: K, low, and high (1<K<10000, 0<low<= high<10^10). K will always be an odd integer.

Output

Output for each case consists of one line: the number of integers between low and high, inclusive, that contain exactly K divisors.

Example

Input:

```
3
3 2 49
9 1 100
5 55 235
```

Output:

```
4
2
1
```

Added by: John Rizzo
Date: 2008-10-04
Time limit: 2s
Source limit:50000B
Languages: All
Resource: Al-Khawarizm 2008 - Set by eleusive

SPOJ Problem Set (classical)

3108. Charlesbert and Merangelou

Problem code: GRAPHGAM

Charlesbert and Merangelou are playing a two-player game on a game board. The game board consists of several circles, where some circles are connected to other circles. Charlesbert uses a game piece called the GoalPiece, and Merangelou uses a game piece called the ChasePiece.
The game is played as follows: at the start of the game, Charlesbert chooses a circle on the board and places the Goalpiece on it, and at the same time Merangelou places the ChasePiece on circle 1. During the next K seconds, Merangelou makes a sequence of moves (at one move per second), where a move consists of either leaving the ChasePiece on its current circle, or moving the ChasePiece from its current circle to a circle that is directly connected to it. If the ChasePiece and the GoalPiece lie on the same circle before or exactly K seconds after Charlesbert last placed the ChasePiece, Merangelou wins. Otherwise, Charlesbert chooses a new circle on the board and places the GoalPiece on it, and Merangelou will make K more moves (Charlesbert moves the GoalPiece before Merangelou makes another move). This process is repeated until the ChasePiece and the GoalPiece lie on the same circle. Note that it is possible for Merangelou to win in fewer than K moves.
Merangelou wants to win the game as quickly as possible, while Charlesbert wants to keep the game going for as long as possible. Assuming both players play optimally, find the shortest amount of time (in seconds) after which Merangelou is guaranteed to have won the game. If Merangelou cannot win the game (i.e. if Charlesbert can keep the game going forever), then print "INFINITE GAME" instead.

Input

The first line of the input contains a positive integer C (0<C<100), the number of test cases to follow. Each case begins with a line containing a two positive integers M and K (1<M<100, 0<K<100), representing the number of circles on the game board and the number of seconds between times that Charlesbert moves the ChasePiece. Each of the following M lines is a string of ’Y’ and ’N’ characters, with M characters per line. If the jth character of the ith line is ’Y’, then circles i and j are connected, otherwise circles i and j are not connected. The jth character of line i will always be the same as the ith character of the jth line, and the kth character of the kth line will always be ’N’. In other words, these M lines form an adjacency matrix for the circles. The ChasePiece always starts on the first circle.

Output

For each case, if Merangelou can win then print the shortest amount of time required for her to win. Otherwise, print "INFINITE GAME". The output for each case should appear on its own line.

Example

Input:

```
3
5 1
NNNNY
NNNNY
NNNNY
NNNNY
YYYYN
```

```
4 2
NYNN
YNYN
NYNY
NNYN
4 1
NYNY
YNYN
NYNY
YNYN
```

Output:

```
2
4
INFINITE GAME
```

Added by: John Rizzo
Date: 2008-10-04
Time limit: 2s
Source limit: 50000B
Languages: All
Resource: Al-Khawarizm 2008 - Set by eleusive

SPOJ Problem Set (classical)

3109. Longest Common Prefix

Problem code: STRLCP

The LCP (Longest Common Prefix) of two strings A[1..la] and B[1..lb] is defined as follows:
LCP(A[1..la],B[1..lb]) = max{L | L<=la && L<=lb && A[1..L] == B[1..L]}
Given an original string and several operations, you should write a program to process all the operations.

Input

The first line will be number of test cases T.
The first line of each test case is a string S with length L (1 <= L <= 100000).
The second line contains an integer Q(1 <= Q <= 150000), representing the number of operations.
Each of the following Q lines represents an operation:
Q i j: print LCP(S[i..L], S[j..L])
R i char: replace the i-th character of S with char
I i char: insert character char after the i-th character of S

Output

For each "Q i j" operation, print the answer.

Example

Input:
```
1
madamimadam
7
Q 1 7
Q 4 8
Q 10 11
R 3 a
Q 1 7
I 10 a
Q 2 11
```

Output:
```
5
1
0
2
1
```

Added by:	John Rizzo
Date:	2008-10-04
Time limit:	3s
Source limit:	50000B
Languages:	All
Resource:	Al-Khawarizm 2008

SPOJ Problem Set (classical)

3110. Palindromic Number

Problem code: PALNUM

A postive integer A is called a "palindrome number" if the reverse of the decimal representaion is the same as the original one. Ex. 13231 is a palindrome number, but 13333 is not.
Given a number A(1 <= A <= 1e18), find the number of pairs (a,b) such that a,b are both palindrome numbers, and the sum of a and b is A.
If A is 391, there are 6 ways: 8 + 383 = 383 + 8 = 391 88 + 303 = 303 + 88 = 391 99 + 292 = 292 + 99 = 391

Input

The first Line contains the number of test cases T <= 10. Each test case contains a number A.

Output

Output the number of ways.

Example

Input:
```
1
391
```

Output:
```
6
```

Added by:	John Rizzo
Date:	2008-10-04
Time limit:	3s
Source limit:	50000B
Languages:	All
Resource:	Al-Khawarizm 2008

SPOJ Problem Set (classical)

3111. Stabards

Problem code: STABARDS

In a galaxy far far away, there exists a silicon based form of life, who call themselves stabards. Unlike humans, the stabards are multi-gendered. Therefore, when two stabards form a partnership (known as mating in earth parlance), one stabard would be the donor of genetic material and the other would be the combinator of the genetic material. Of course, the combinator had the tougher task, to combine the genetic material and create a new stabard.
Early on, the wise stabards realized that due to the increased number of genders, every stabard was likely to waste his time trying to find a suitable partner. After that, due to the tremendous opportunity, every stabard would waste some more time trying to cheat on his partner by forming more partnerships (especially the donors since they had nothing much to do). Therefore, the wise ones made the following rules about partnerships:

1. Each stabard can form at most two partnerships.
2. In order to maintain a sense of balance, a stabard cannot play the same role (donor/combinator) in both partnerships.
3. The partnerships shall be formed such that the total number of them is maximized.

To ensure all rules are being followed, the wise ones send the stabard data every year to earth and wish to know the maximum number of partnerships that can be formed.

Input

For each test case, two integers M (the number of stabard genders) and N (the total number of stabards) are given on the first line. M lines follow, each consisting of M characters. The j-th character on the i-th line denotes what would happen if a stabard of gender i formed a partnership with a stabard of gender j. It will be either
'X' - such a partnership is forbidden.
'D' - stabard of gender i would be the donor.
'C' - stabard of gender i would be the combinator.
After the M lines, N space separated integers are given on a single line. The i-th integer gives the gender of the i-th stabard.
The end of the test cases is given by a line with M and N both being 0. This test case should not be processed. The total number of test cases will be <= 100.

Constraints

$0 < M <= 100$
$0 < N <= 100$
The gender data for stabards will be symmetric and consistent. (i.e. character i on line j will not conflict with character j on line i). Two stabards of the same gender can never partner each other. (The wise ones fear this will pollute the gene pool. Moreover, big fights would break out as who would be the donor.) The gender given for each stabard will be between 0 and M-1 inclusive.

Output

For each test case, a single integer giving the total number of partnerships. Each integer must be on its own line.

Example

Input:

```
2 4
XD
CX
0 0 1 1
3 3
XDC
CXD
DCX
0 1 2
0 0
```

Output:

```
2
3
```

Added by: John Rizzo
Date: 2008-10-04
Time limit: 1s
Source limit: 50000B
Languages: All
Resource: Al-Khawarizm 2008 - Set by humblefool

SPOJ Problem Set (classical)

3112. Strings

Problem code: STSTRING

Given two strings A and B, we define the operator cx on {A,B} for string C as C cx {A,B}.

```
if length(A) < length(C) < length(B), then C satisfies the above operator.
else
      if length(A)=length(C), then C must be lexicographically greater than A.
      if length(B)=length(C), then C must be lexicographically smaller than B.
```

Input

Given two strings A,B with length(A) < = length(B) <= 6. A,B can contain any characters between A and J (capital letters).

Output

Print the number of strings satisfying the above criteria. C must also satisfy criteria of A and B. Any two adjacent characters in string C may neither be the same nor consecutve (i.e. the absoulte difference between the ASCII values of adjacent characters is greater than 1).

Example

Input:

```
A J
AA BCD
ABC DEFG
```

Output:

```
8
129
1770
```

Added by: John Rizzo
Date: 2008-10-04
Time limit: 2s
Source limit:50000B
Languages: All
Resource: Al-Khawarizm 2008 - Set by FameofLight and Challenger

SPOJ Problem Set (classical)

3133. Here We Go(relians) Again

Problem code: GORELIAN

The Gorelians are a warlike race that travel the universe conquering new worlds as a form of recreation. Given their violent, fun-loving nature, keeping their leaders alive is of serious concern. Part of the Gorelian security plan involves changing the traffic patterns of their cities on a daily basis, and routing all Gorelian Government Officials to the Government Building by the fastest possible route.

Fortunately for the Gorelian Minister of Traffic (that would be you), all Gorelian cities are laid out as a rectangular grid of blocks, where each block is a square measuring 2520 rels per side (a rel is the Gorelian Official Unit of Distance). The speed limit between two adjacent intersections is always constant, and may range from 1 to 9 rels per blip (a blip, of course, being the Gorelian Official Unit of Time). Since Gorelians have outlawed decimal numbers as unholy (hey, if you're the dominant force in the known universe, you can outlaw whatever you want), speed limits are always integer values. This explains why Gorelian blocks are precisely 2520 rels in length: 2520 is the least common multiple of the integers 1 through 9. Thus, the time required to travel between two adjacent intersections is always an integer number of blips.

In all Gorelian cities, Government Housing is always at the northwest corner of the city, while the Government Building is always at the southeast corner. Streets between intersections might be one-way or two-way, or possibly even closed for repair (all this tinkering with traffic patterns causes a lot of accidents). Your job, given the details of speed limits, street directions, and street closures for a Gorelian city, is to determine the fastest route from Government Housing to the Government Building. (It is possible, due to street directions and closures, that no route exists, in which case a Gorelian Official Temporary Holiday is declared, and the Gorelian Officials take the day off.)

Gorelian city

The picture above shows a Gorelian City marked with speed limits, one way streets, and one closed street. It is assumed that streets are always traveled at the exact posted speed limit, and that turning a corner takes zero time. Under these conditions, you should be able to determine that the fastest route from Government Housing to the Government Building in this city is 1715 blips. And if the next day, the only change is that the closed road is opened to two way traffic at 9 rels per blip, the fastest route becomes 1295 blips. On the other hand, suppose the three one-way streets are switched from southbound to northbound (with the closed road remaining closed). In that case, no route would be possible and the day would be declared a holiday.

Input

The input consists of a set of cities for which you must find a fastest route if one exists. The first line of an input case contains two integers, which are the vertical and horizontal number of city blocks, respectively. The smallest city is a single block, or 1 by 1, and the largest city is 20 by 20 blocks. The remainder of the input specifies speed limits and traffic directions for streets between intersections, one row of street segments at a time. The first line of the input (after the dimensions line) contains the data for the northernmost east-west street segments. The next line contains the data for the northernmost row of north-south street segments. Then the next row of east-west streets, then

north-south streets, and so on, until the southernmost row of east-west streets. Speed limits and directions of travel are specified in order from west to east, and each consists of an integer from 0 to 9 indicating speed limit, and a symbol indicating which direction traffic may flow. A zero speed limit means the road is closed. All digits and symbols are delimited by a single space. For east-west streets, the symbol will be an asterisk '*' which indicates travel is allowed in both directions, a less-than symbol '<' which indicates travel is allowed only in an east-to-west direction, or a greater-than symbol '>' which indicates travel is allowed only in a west-to-east direction. For north-south streets, an asterisk again indicates travel is allowed in either direction, a lowercase "vee" character 'v' indicates travel is allowed only in a north-to-south directions, and a caret symbol '^' indicates travel is allowed only in a south-to-north direction. A zero speed, indicating a closed road, is always followed by an asterisk. Input cities continue in this manner until a value of zero is specified for both the vertical and horizontal dimensions.

Output

For each input scenario, output a line specifying the integer number of blips of the shortest route, a space, and then the word "blips". For scenarios which have no route, output a line with the word "Holiday".

Example

Input:

```
2 2
9 * 9 *
6 v 0 * 8 v
3 * 7 *
3 * 6 v 3 *
4 * 8 *
2 2
9 * 9 *
6 v 9 * 8 v
3 * 7 *
3 * 6 v 3 *
4 * 8 *
2 2
9 * 9 *
6 ^ 0 * 8 ^
3 * 7 *
3 * 6 ^ 3 *
4 * 8 *
0 0
```

Output:

```
1715 blips
1295 blips
Holiday
```

Added by: Nikola P Borisov
Date: 2008-10-11
Time limit: 5s
Source limit: 50000B
Languages: All
Resource: Mid-Central Regional ACM-ICPC Contest 2007

SPOJ Problem Set (classical)

3166. Permutation Exponentiation

Problem code: PERMSG

Alice, a permutation aficionado, has thought up a permutation of **N** (1 <= **N** <= 100000) elements, **P**. So impressed with herself she has told her friend Bob about **P**.

Normally Alice would call it a day after creating such an impressive permutation but today she decided that she wanted to raise **P** to the power k as well! Unfortunately, after working on the problem for a while she gave up because it was taking too long. Not wanting her efforts to go to waste she once again tells Bob about all the elements she has determined so far. Unfortunately, she neglected to tell Bob the value k.

Bob, very interested in Alice's work, needs your help to try and determine any additional elements of **P**^k. Bob is suspicious of Alice's work so he also asks you to check it for errors.

Input

The first line of input contains **N** (1 <= **N** <= 100000), the number of elements in the permutation. The next line contains a permutation of the integers 0 through **N**-1 each separated by a space. The following line will contain the result of applying the permutation k times with the exception that elements that are not known will be -1 instead.

Output

Print **P**^k as a space separated list on its own line with as many elements as possible determined. If an element can't be determined leave it as -1. If there is no k such that **P**^k has the values given in the input print "Inconsistent" (quotes for clarity) on its own line instead.

Example

Input:
```
4
1 2 3 0
3 -1 -1 -1
```

Output:
```
3 0 1 2
```

Example 2

Input:
```
4
1 2 3 0
3 -1 2 -1
```

Output:
```
Inconsistent
```

Added by:	Mark Gordon
Date:	2008-10-17
Time limit:	3s
Source limit:	50000B
Languages:	All
Resource:	Own problem

SPOJ Problem Set (classical)

3184. Game of Lines

Problem code: LINES

Farmer John has challenged Bessie to the following game: FJ has a board with dots marked at N (2 <= N <= 200) distinct lattice points. Dot i has the integer coordinates X_i and Y_i (-1,000 <= X_i, Y_i <= 1,000).

Bessie can score a point in the game by picking two of the dots and drawing a straight line between them; however, she is not allowed to draw a line if she has already drawn another line parallel to it. Bessie would like to know her chances of winning, so she has asked you to help find the maximum score she can obtain.

Input

There will be multiple test cases. For each case, the first line contains the integer N, and each of the next N lines gives a pair of integers, X_i and Y_i. The file ends with the case N = 0, which should not be processed.

Output

For each test case, print a single integer representing the maximum number of lines Bessie can draw, no two of which are parallel.

Example

Input:
```
4
-1 1
-2 0
0 0
1 1
0
```

Output:
```
4
```

Bessie can draw lines of the following four slopes: -1, 0, 1/3, and 1.

Added by: Neal Wu
Date: 2008-10-19
Time limit: 1s
Source limit:50000B
Languages: All
Resource: USACO Feb 2008

SPOJ Problem Set (classical)

3195. Doors and Penguins

Problem code: DOORSPEN

The organizers of the Annual Computing Meeting have invited a number of vendors to set up booths in a large exhibition hall during the meeting to showcase their latest products. As the vendors set up their booths at their assigned locations, they discovered that the organizers did not take into account an important fact -- each vendor supports either the Doors operating system or the Penguin operating system, but not both. A vendor supporting one operating system does not want a booth next to one supporting another operating system. Unfortunately the booths have already been assigned and even set up. There is no time to reassign the booths or have them moved. To make matter worse, these vendors in fact do not even want to be in the same room with vendors supporting a different operating system. Luckily, the organizers found some portable partition screens to build a wall that can separate the two groups of vendors. They have enough material to build a wall of any length. The screens can only be used to build a straight wall. The organizers need your help to determine if it is possible to separate the two groups of vendors by a single straight wall built from the portable screens. The wall built must not touch any vendor booth (but it may be arbitrarily close to touching a booth). This will hopefully prevent one of the vendors from knocking the wall over accidentally.

Input

The input consists of a number of cases. Each case starts with 2 integers on a line separated by a single space: D and P, the number of vendors supporting the Doors and Penguins operating system, respectively (1 <= D, P <= 500). The next D lines specify the locations of the vendors supporting Doors. This is followed by P lines specifying the locations of the vendors supporting Penguins. The location of each vendor is specified by four positive integers: x1, y1, x2, y2. (x1, y1) specifies the coordinates of the southwest corner of the booth while (x2, y2) specifies the coordinates of the northeast corner. The coordinates satisfy x1 < x2 and y1 < y2. All booths are rectangular and have sides parallel to one of the compass directions. The coordinates of the southwest corner of the exhibition hall is (0, 0) and the coordinates of the northeast corner is (15000, 15000). You may assume that all vendor booths are completely inside the exhibition hall and do not touch the walls of the hall. The booths do not overlap or touch each other. The end of input is indicated by D = P = 0.

Output

For each case, print the case number (starting from 1), followed by a colon and a space. Next, print the sentence: It is possible to separate the two groups of vendors. if it is possible to do so. Otherwise, print the sentence: It is not possible to separate the two groups of vendors. Print a blank line between consecutive cases.

Example

Input:

```
3 3
10 40 20 50
50 80 60 90
30 60 40 70
```

```
30 30 40 40
50 50 60 60
10 10 20 20
2 1
10 10 20 20
40 10 50 20
25 12 35 40
0 0
```

Output:

```
Case 1: It is possible to separate the two groups of vendors.

Case 2: It is not possible to separate the two groups of vendors.
```

Added by: Daniel Gómez Didier
Date: 2008-10-21
Time limit: 1s-2s
Source limit: 50000B
Languages: All
Resource: 2007 PUJ - Circuito de Maratones ACIS / REDIS

SPOJ Problem Set (classical)

3208. Yet Another Longest Palindrome Problem

Problem code: PALIM

A string is called a palindrome if it's the same when read from left to right and from right to left. For example, `"abdba"` is a palindrome, but `"abbaa"` is not.

Given a string, print the length of the longest consecutive sequence of characters occurrences at least once in this string, which is a palindrome.

Input

- Line 1: a string consists of at most 100000 characters. The ASCII code of all characters are between 32 and 127, inclusive.
- Line 2: a magical key(for security purpose).

Output

- Line 1: the length of the longest palindrome.
- Line 2: the magical key.

Example

Input:
```
abaabbabaaba
MAGICAL KEY
```

Output:
```
6
MAGICAL KEY
```

Restriction

Only C++ is allowed in this problem now. **In addition, you will receive `"wrong answer"` if your program don't start with this.** You can't use macro `"#undef"` in your solution as well.

If you want to solve this problem in another language, send me the header file in your language please.

warning: Don't try to access the memory of tester, or I will reject your solution manually, and you will lose the chance to enjoy this problem as well.

Hint

hint of using tester library: you can't read anything from stdin, and you can't print anything as well, your program will be terminated if you called answer().

hint of viewing feedback: You can click on "`wrong answer`" link to view the feedback of judge: whether your solution didn't include the testlib, or failed on sample. (if neither, your solution failed on a further test case)

Notice

update on Oct.24: I had updated the header file for C++, now you will receive "`Runtime Error(NZEC)`" if your solution called isSame() illegally. The submissions with old version of header file are still acceptable.

rejudge on Oct.24: some test cases were added, three submissions were rejudged as TLE instead of AC.

Added by: Jin Bin
Date: 2008-10-23
Time limit: 1s
Source limit: 10240B
Languages: C++
Resource: the (second??) "`interactive`" problem on SPOJ

SPOJ Problem Set (classical)

3249. Typesettin

Problem code: TYPESET

Modern fonts are generally of two varieties: outline fonts, whose glyphs (the individual character shapes) are specified mathematically as a set of curves, and bitmap fonts, whose glyphs are specified as patterns of pixels. Fonts may also include embedded information such as kerning pairs (adjusting the spacing between certain pairs of glyphs, such as "AW", so that they appear spaced correctly), tracking hints (for managing inter-glyph spacing), antialiasing hints (smoothing of pixellated edges), and much more. To be sure, modern fonts are more than a simple collection of shapes, and displaying them properly is a common programming challenge.

For this problem we will concern ourselves with bitmapped fonts and a simple form of typesetting called glyph packing. Essentially, the idea is to pack the glyphs as tightly as possible while maintaining at least one horizontal pixel of separation between glyphs. For example, consider the glyphs shown to the left below for the Roman characters "P" and "J". The figure to the right shows them after glyph packing. Note that they are as close as possible without touching horizontally.

PJ

Here’s another example. In this case, notice that the final glyph cannot be packed at all.

Fiji

After packing, pixels from distinct glyphs may be adjacent diagonally or vertically, but not horizontally. The following example shows that pixels may be adjacent diagonally. The "Love" test case in the example input section shows that they may be adjacent vertically.

two slashes

Glyph packing has the nice property that it’s easy to build "fancy" glyphs into the font so that glyph packing creates special effects with no extra work. Look at the "Toy" example below. The same simple packing process has been applied to these glyphs as to the ones above, but the result is more dramatic:

Toy

Glyph packing has a few caveats, however, one of which we must concern ourselves with for this problem. Consider the example on the left below where a glyph for a hyphen is followed by a glyph for an underscore. Based on our one horizontal pixel of separation rule, how would this pack? Clearly something more is needed, and that something more is hinting within the glyphs themselves. Recall that in actual practice, fonts contain kerning pairs, tracking hints, etc. For our purposes, our hinting will be limited to "invisible" pixels that count as a pixel for the purpose of packing, but not for display. The center image below represents invisible pixels as open dots instead of closed dots. Now the two glyphs can be properly packed, resulting in the output shown on the right.

hidden pixels

Now for the formal definition of a proper packing: (1) Glyphs are packed as close as possible without allowing any pixels from different glyphs to be immediately horizontally adjacent; (2) Given two glyphs, they may not be packed in such a way that any pixel of the leftmost glyph at a given height ends up positioned to the right of any pixel at the same height in the rightmost glyph.

Condition (2) above is easily understood by visualizing two glyphs sitting side by side, separated by a small space. If you "squeeze" them together, condition (2) says that their pixels are not allowed to "pass through" one another. Consider the example to the left below. The center image is not the proper packing, because it violates condition (2) of the formal definition. The image on the right is the proper packing of these glyphs.

hooks

Input

The input for this problem is sets of glyphs to be packed. In a given test case, all glyphs are the same height, and an integer, N, on the first line of the test case specifies this height. The next N lines contain the glyphs to be packed. Empty pixels in a glyph are represented by a dot '.' character. Non-empty pixels are represented by a hash mark '#' for visible pixels, and a zero '0' for invisible pixels. Glyphs are separated by a single column of space characters. The input will always consist of more than one glyph, at least one of which will always contain at least one visible pixel. A glyph will always have at least one non-empty pixel in its leftmost and rightmost column, and every glyph will have at least one non-empty pixel at the same height as at least one other glyph in the input. The minimum dimension of a glyph is 1 x 1, the maximum dimension is 20 x 20, and the maximum number of glyphs that will appear in any test case is 20. Test cases continue until a value of zero is specified for N.

Output

For each test case, first output the number of that test case (starting with 1) on a line by itself. Then output the proper packing of the input glyphs, using the dot '.' character for empty pixels and for invisible pixels, and the hash mark '#' character for visible pixels. Omit leading and trailing empty columns (columns with no visible pixels) so that both the leftmost and rightmost output columns contain at least one visible pixel.

Example

Input:

```
8
###. ...#
#..# ...#
#..# ...#
###. ...#
#... ...#
#... ...#
#... #..#
#... ####
8
############ .... ............
..#......... .... ............
..#......... .##. .........#..#
..#......... #..# .........#..#
..#......... #..# .........#..#
```

```
..#.......... .##. ..........###
.............. .... ............#
.............. .... #############.
8
############# ..............
..#.......... ..............
..#.......... .........#..#
..#.......... .........#..#
..#.......... .........#..#
..#.......... ...........###
.............. ............#
.............. #############.
5
0..0 0..0
0..0 0..0
#### 0..0
0..0 0..0
0..0 ####
5
#.... .###.
#.... #...#
#...# #...#
#...# ....#
.###. ....#
3
### 0.0 ###
#.# 0.0 #.#
### 0.0 ###
3
0.0 ### 0.0
0.0 #.# 0.0
0.0 ### 0.0
8
#.... .... ..... ....
#.... .... ..... ....
#.... .##. #...# .##.
#.... #..# .#.#. #..#
#.... #..# .#.#. #..#
#.... #..# .#.#. ###.
#.... .##. ..#.. #...
##### .... ..#.. .###
0
```

Output:

```
1
###..#
#..#.#
#..#.#
###..#
#....#
#....#
#.#..#
#.####
2
##############
..#..........
..#..##..#..#
..#.#..#.#..#
..#.#..#.#..#
..#..##...###
............#
#############.
```

```
3
.....#############
.......#..........
.......#.#..#.....
.......#.#..#.....
.......#.#..#.....
.......#..###.....
............#.....
############......
4
.........
.........
####.....
.........
.....####
5
#......###.
#.....#...#
#...#.#...#
#...#.....#
.###......#
6
###.....###
#.#.....#.#
###.....###
7
###
#.#
###
8
#..............
#..............
#..##.#...#.##.
#.#..#.#.#.#..#
#.#..#.#.#.#..#
#.#..#.#.#.###.
#..##...#..#...
######...#...###
```

Added by: Nikola P Borisov
Date: 2008-10-25
Time limit: 10s
Source limit: 50000B
Languages: All
Resource: Mid-Central Regional ACM-ICPC Contest 2007

SPOJ Problem Set (classical)

3251. Slink

Problem code: SLINK

Slitherlink is a puzzle published by Nikoli, the Japanese company that popularized Sudoku. Slitherlink puzzles are gaining momentum, and books of Slitherlink puzzles have started showing up around the world. The puzzles are simple to understand, but can be challenging to solve. The puzzle is simply a rectangular grid of dots that forms a collection of cells, every cell being either blank or containing an integer from zero to three. The challenge is to connect the dots with line segments to form a cycle (a connected path such that every vertex has precisely two incident edges), in such a way that every cell with a value has exactly the number of incident edges as the digit it contains. Cells with no value may have any number of incident edges. A valid Slitherlink puzzle always contains sufficient non-empty cells to guarantee a unique solution. Below is an example from the Nikoli web site of a Slitherlink puzzle and its solution.

slitherlink example

It was shown by Takayuki Yato at the University of Tokyo that the general Slitherlink problem is NP-complete. (If you are not familiar with this concept, informally it means there is no "efficient" algorithm to solve the problem.) With a slight modification and some simple heuristics, however, programmatic solutions are possible. Our new puzzle, which we will term Slink, differs from Slitherlink only in that the puzzle may not have empty cells. That is, every cell must specify the number of incident edges. Below is the Slitherlink puzzle above converted to Slink (the added numbers are shown in gray). Note that the solution does not change, only the information given in the puzzle itself.

slink example

The heuristics for solving Slink arise from the nature of the puzzle itself. For example, consider a cell containing a zero. There must be no incident edges, therefore all edges incident to all zeros can be immediately removed from consideration as part of the solution path. Consider a three next to a zero. Because all the edges incident to the zero will be eliminated, the common edge shared with the three is also eliminated. But that leaves only three edges around the three, and therefore those three edges must be part of the solution path. The following table specifies the heuristic rules that must be properly applied to solve a Slink puzzle. The "x" characters between vertices mark edges that are not part of the solution path, while line segments between vertices mark edges that form part of the solution. Grey elements are the pattern the rule is based on, black elements indicate the additional edges that should be included or excluded if the rule is matched. Note that the pictured examples are for demonstration purposes only and do *not* illustrate every possible arrangement of the stated rule!

Look here for more pictures

Examples	Rule Specification		Examples	Rule Specification

case 1	The easiest and most obvious of all the rules. Cells containing a zero have no incident edges, so all the edges around a zero should be removed from consideration as part of the solution path.8 8 1 0 1 1 2 2 1 3 3 3 3 3 2 3 3 2 2 2 0 1 1 2 2 0 2 3 1 1 0 1 2 2 2 1 2 3 1 1 0 2 1 2 2 2 2 3 2 1 3 2 1 3 1 1 3 2 1 0 0 2 3 2 3 2 6 6 0 0 1 1 0 0 0 2 2 2 2 0 1 2 0 0 2 1 1 2 0 0 2 1 0 2 2 2 2 0 0 0 1 1 0 0 2 2 2 2 2 2 3 5 3 3 3 2 3 1 2 1 3 2 3 3 2 2 2 0 0		case 2 case 3	If a cell contains the value *n* and only *n* incident edges remain (i.e. have not been eliminated), then the *n* remaining edges must be part of the solution path. Two examples of this occurring are shown here.
case 4 case 5	If a cell contains the value *n* and *n* incident edges have already been included in the path, the remaining edges can be eliminated. Two examples of this occurring are shown here.		case 6 8 8 1 0 1 1 2 2 1 3 3 3 3 3 2 3 3 2 2 2 0 1 1 2 2 0 2 3 1 1 0 1 2 2 2 1 2 3 1 1 0 2 1 2 2 2 2 3 2 1 3 2 1 3 1 1 3 2 1 0 0 2 3 2 3 2 6 6 0 0 1 1 0 0 0 2 2 2 2 0 1 2 0 0 2 1 1 2 0 0 2 1 0 2 2 2 2 0 0 0 1 1 0 0 2 2 2 2 2 2 3 5 3 3 3 2 3 1 2 1 3 2 3 3 2 2 2 0 0	If two 3's are adjacent to one another, the common edge between the cells as well as the outer edges of both cells are part of the solution path. One example of this arrangement occurring is shown here.
case 7	If two 3's occur diagonally adjacent, the opposing corners as shown here must be part of the solution path. One example of such an arrangement is shown here.		case 8	If an edge enters a vertex for which only a single exit remains, that exit must be part of the solution path. One such example is shown here.
case 9	If a vertex has two incident edges, the other edges can be eliminated from consideration as part of the solution path. One such example is shown here.		case 10	If any vertex has three incident edges excluded, the fourth incident edge can be excluded as well. One possible arrangement of this occurring is shown here.
case 11	A 3 for which two of the exits are blocked as shown, such as in a corner of the puzzle, must include the two edges incident to the blocked vertex.		case 12	If the exits at one corner of a 2 are blocked, and one exit at an adjacent vertex around the 2 is also blocked, then the unblocked exit at that adjacent vertex must be part of the solution path. One example of this arrangement is shown here.

case 13	A 1 for which the exit paths at one of its incident vertices are both blocked as shown, such as might occur in the corner of the puzzle, must also eliminate the other two edges incident to that vertex as shown.		case 14	If the solution path enters the corner of a 3, and the exit that goes away from the 3 at that same corner is blocked, then the two edges around the three incident to the opposite corner must be part of the solution path.
case 15	If a 3 and 1 are diagonally adjacent, and the corner of the 3 furthest from the 1 has the exit segments blocked as shown, then the edges incident to the far corner of the 1 becomes blocked. The opposite is also true; if the far corner of the 1 had been blocked, then the exit segments at the far corner of the 3 would become blocked in the same manner.		case 16	If the solution path enters the corner of 2 and the path leading away from the 2 at the same corner is blocked, then if one of the paths leading away from the 2 at the diagonally opposite corner is also blocked, the other edge leading away from the 2 at that same corner must be part of the solution path. One example of this arrangement occurring is shown here.
case 17	If the solution path enters the corner of a 1, and the exit that goes away from the 1 at that same corner is blocked, then the two edges around the three incident to the opposite corner must be eliminated from the solution path.			

Input

The input for this problem is a set of Slink puzzles to be solved. The first line of a Slink problem's input contains two integers, *r* and *c*, separated by a space, the number of rows and the number of columns in the puzzle. The next *r* rows of the input contain *c* integers, space delimited, valued from 0 to 3, which specify the content of the puzzle. The minimum dimension of a puzzle is 2 by 2 cells, and the maximum dimension is 20 by 20 cells. It is guaranteed that a unique solution to every input puzzle exists and can be determined with the above rules if a rule is always applied when it can be applied. A line with values of zero for *r* and *c* marks the end of the input.

Output

The output for this problem is a graphical representation of the Slink puzzle solution. The first data set is 1, the second data set is 2, etc. On a line by itself display the data set number, followed by the solution in exactly the format demonstrated below. Vertical edges are output as the vertical bar 'l' character, horizontal edges are output as dash '-' characters, vertices where the path changes direction are output as plus signs '+', and cell numbers are always displayed with a blank to the left and to the right. Further, surround the entire output with a border made up of hash marks '#' such that the

number in the upper left cell of the puzzle always occurs four positions to the right of the border and three position below the border, and the number in the lower right cell always occurs four positions to the left of the border and three positions above the border.

Example

Input:

```
8 8
1 0 1 1 2 2 1 3
3 3 3 3 2 3 3 2
2 2 0 1 1 2 2 0
2 3 1 1 0 1 2 2
2 1 2 3 1 1 0 2
1 2 2 2 2 3 2 1
3 2 1 3 1 1 3 2
1 0 0 2 3 2 3 2
6 6
0 0 1 1 0 0
0 2 2 2 2 0
1 2 0 0 2 1
1 2 0 0 2 1
0 2 2 2 2 0
0 0 1 1 0 0
2 2
2 2
2 2
3 5
3 3 3 2 3
1 2 1 3 2
3 3 2 2 2
0 0
```

Output:

```
1
#######################################
#                                     #
#                   +---------------+ #
#   1   0   1   1 | 2   2   1   3 | #
# +---+   +---+   |   +---+   +---+ #
# | 3 | 3 | 3 | 3 | 2 | 3 | 3 | 2   #
# |   +---+   +---+   |   +---+     #
# | 2   2   0   1   1 | 2   2   0   #
# +-------+           +-------+     #
#   2   3 | 1   1   0   1   2 | 2   #
# +-------+   +---+           +---+ #
# | 2   1   2 | 3 | 1   1   0   2 | #
# |       +---+   |   +---+       | #
# | 1   2 | 2   2 | 2 | 3 | 2   1 | #
# |   +---+   +---+   |   +---+   | #
# | 3 | 2   1 | 3   1 | 1   3 | 2 | #
# +---+       +---+   |   +---+   | #
#   1   0   0   2 | 3 | 2 | 3   2 | #
#                 +---+   +-------+ #
#                                     #
#######################################
2
##############################
#                            #
#                            #
#   0   0   1   1   0   0    #
```

```
#          +-------+          #
#    0   2 | 2   2 | 2   0    #
#      +---+       +---+      #
#    1 | 2   0   0   2 | 1    #
#      |                 |     #
#    1 | 2   0   0   2 | 1    #
#      +---+       +---+      #
#    0   2 | 2   2 | 2   0    #
#          +-------+          #
#    0   0   1   1   0   0    #
#                             #
#                             #
###############################
3
#############
#           #
# +-------+ #
# | 2   2 | #
# |       | #
# | 2   2 | #
# +-------+ #
#           #
#############
4
#########################
#                       #
# +---+   +---+   +---+ #
# | 3 | 3 | 3 | 2 | 3 | #
# |   +---+   |   |   | #
# | 1   2   1 | 3 | 2 | #
# |   +---+   +---+   | #
# | 3 | 3 | 2   2   2 | #
# +---+   +-----------+ #
#                       #
#########################
```

Added by: Nikola P Borisov
Date: 2008-10-25
Time limit: 5s
Source limit: 50000B
Languages: All
Resource: Mid-Central Regional ACM-ICPC Contest 2007

SPOJ Problem Set (classical)

3253. Electronic Document Security

Problem code: EDS

The Tyrell corporation uses a state-of-the-art electronic document system that controls all aspects of document creation, viewing, editing, and distribution. Document security is handled via *access control lists* (ACLs). An ACL defines a set of entities that have access to the document, and for each entity defines the set of rights that it has. Entities are denoted by uppercase letters; an entity might be a single individual or an entire division. Rights are denoted by lowercase letters; examples of rights are *a* for *append*, *d* for *delete*, *e* for *edit*, and *r* for *read*.

The ACL for a document is stored along with that document, but there is also a separate ACL *log* stored on a separate log server. All documents start with an empty ACL, which grants no rights to anyone. Every time the ACL for a document is changed, a new entry is written to the log. An entry is of the form *ExR*, where *E* is a nonempty set of entities, *R* is a nonempty set of rights, and *x* is either "+", "-", or "=". Entry *E+R* says to grant all the rights in *R* to all the entities in *E*, entry *E-R* says to remove all the rights in *R* from all the entities in *E*, and entry *E=R* says that all the entities in *E* have exactly the rights in *R* and no others. An entry might be redundant in the sense that it grants an entity a right it already has and/or denies an entity a right that it doesn't have. A log is simply a list of entries separated by commas, ordered chronologically from oldest to most recent. Entries are cumulative, with newer entries taking precedence over older entries if there is a conflict.

Periodically the Tyrell corporation will run a security check by using the logs to compute the current ACL for each document and then comparing it with the ACL actually stored with the document. A mismatch indicates a security breach. Your job is to write a program that, given an ACL log, computes the current ACL.

Input

The input consists of one or more ACL logs, each 3-79 characters long and on a line by itself, followed by a line containing only "#" that signals the end of the input. Logs will be in the format defined above and will not contain any whitespace.

Output

For each log, output a single line containing the log number (logs are numbered sequentially starting with one), then a colon, then the current ACL in the format shown below. Note that (1) spaces do not appear in the output; (2) entities are listed in alphabetical order; (3) the rights for an entity are listed in alphabetical order; (4) entities with no current rights are not listed (even if they appeared in a log entry), so it's possible that an ACL will be empty; and (5) if two or more consecutive entities have exactly the same rights, those rights are only output once, after the list of entities.

Example

Input:

```
MC-p,SC+c
YB=rde,B-dq,AYM+e
GQ+tju,GH-ju,AQ-z,Q=t,QG-t
JBL=fwa,H+wf,LD-fz,BJ-a,P=aw
#
```

Output:

```
1:CSc
2:AeBerMeYder
3:
4:BHJfwLPaw
```

Added by: Nikola P Borisov
Date: 2008-10-25
Time limit: 1s
Source limit: 50000B
Languages: All
Resource: Mid-Central Regional ACM-ICPC Contest 2007

SPOJ Problem Set (classical)

3254. Guard

Problem code: GUARD

Guard Placement

The Bluewater Security Company provides guards for clients with valuable possessions. Bluewater has found that clients are interested in having guards posted where they can see everything that is valuable merely by turning their heads, and also like guards to be posted particularly close to particularly valuable items. A sample site layout is shown above. Ignore the three black dots for now. Various locations are labeled and assigned values. For instance location A at coordinates (0,8) is the position of an item with value 4. Locations showing a value 0, like G, do not have a valuable item. The straight lines indicate corridors. For simplicity, corridors are modeled as line segments with 0 width. A guard at an intersection point of several corridors can see and therefore guard the items on each of the corridors. If Bluewater were contracted to supply 3 guards, they might choose to post them at the positions indicated with the small black dots. The guard not at an already labeled position is at (15.5, 6). To model the desire for guards to be closer to items of higher value, Bluewater calculates the risk to a valuable item to be the value of the item times the minimum distance to a guard that can see the item. Even if a guard is close to an item that is around a corner, that guard does not affect the risk to the item, since the guard cannot see around a corner. In the diagram shown, the risks to the items are A: 4x5=20, C: 4x2.5=10, D: 2x0=0, ... The largest risks are for H: 50x7.5=375 and I: 50x7.5=375, so the maximum risk to any one item is 375. With this site layout, no arrangement of 3 guards would provide a lower maximum risk, so this arrangement of 3 guards minimizes the maximum risk. Bluewater would like to be able to tell any client who requests a particular number of guards for a particular site layout, what the minimized maximum risk will be.

Input

The input will consist of one to sixteen data sets, followed by a line containing only 0. On each line the data will consist of blank separated tokens.

The first line of a dataset contains integers p c g, where p is the number of points specified, c is the number of corridors, and g is the number of guards to be placed. Constraints are $1< p < 12$; $0 < c < 12$; $0 < g < 5$.

Next in the dataset are a total of p groups of four tokens, each consisting of a capital letter and three nonnegative integers L x y v indicating the point (x, y) with label L contains an item with value v. If p is no greater than 6, these groups will all be on one line. If p is greater than 6, then the seventh and further groups will be on the next line. Labels will be consecutive letters starting from A. All the numbers are less than 1000. Each of the points is unique. A value of 0 for v means there is no item of value there. The number of locations with items of value will be at least as large as the number of guards.

The last line of a dataset contains c strings of letters, one for each corridor. For each corridor the letters are labels for points along the corridor, in order along the line segment from one end to the other, including both endpoints, all intersection points with other corridors, and all locations on the corridor with a valuable item. Each of the points given in the dataset will lie on at least one of the corridors.

Output

There is one line of output for each data set. If there are not enough guards supplied to be able to see all the valuables, the line is "too few guards". Otherwise the line is an unsigned number r rounded to two places beyond the decimal point, where r is the minimum value over all placements of g guards of the maximum "risk" to the valuables.

The first example dataset matches the illustration above, and the next three examples only vary the number of guards.

Example

Input:

```
11 5 3
A 0 8 4 B 5 8 0 C 14 8 4 D 21 8 2 E 25 8 1 F 5 22 1
G 5 20 0 H 11 12 50 I 20 0 50 J 19 10 5 K 25 4 5
ABCDE AG FGB GHCI JDK
11 5 2
A 0 8 4 B 5 8 0 C 14 8 4 D 21 8 2 E 25 8 1 F 5 22 1
G 5 20 0 H 11 12 50 I 20 0 50 J 19 10 5 K 25 4 5
ABCDE AG FGB GHCI JDK
11 5 1
A 0 8 4 B 5 8 0 C 14 8 4 D 21 8 2 E 25 8 1 F 5 22 1
G 5 20 0 H 11 12 50 I 20 0 50 J 19 10 5 K 25 4 5
ABCDE AG FGB GHCI JDK
11 5 4
A 0 8 4 B 5 8 0 C 14 8 4 D 21 8 2 E 25 8 1 F 5 22 1
G 5 20 0 H 11 12 50 I 20 0 50 J 19 10 5 K 25 4 5
ABCDE AG FGB GHCI JDK
3 3 1
A 0 0 50 B 0 3 60 C 4 0 20
AB CB CA
0
```

Output:

```
375.00
1250.00
too few guards
21.21
150.00
```

Added by: Nikola P Borisov
Date: 2008-10-25
Time limit: 5s
Source limit: 50000B
Languages: All
Resource: Mid-Central Regional ACM-ICPC Contest 2007

SPOJ Problem Set (classical)

3261. Race Against Time

Problem code: RACETIME

As another one of their crazy antics, the N (1 <= N <= 100,000) cows want Farmer John to race against the clock to answer some of their pressing questions.

The cows are lined up in a row from 1 to N, and each one is holding a sign representing a number, A_i (1 <= A_i <= 1,000,000,000). The cows need FJ to perform Q (1 <= Q <= 50,000) operations, which can be either of the following:

- Modify cow i's number to X (1 <= X <= 1,000,000,000). This will be represented in the input as a line containing the letter M followed by the space-separated numbers i and X.
- Count how many cows in the range [P, Q] (1 <= P <= Q <= N) have A_i <= X (0 <= X <= 1,000,000,000). This will be represented in the input as a line containing the letter C followed by the space-separated numbers P, Q, and X.

Of course, FJ would like your help.

Input

The first line gives the integers N and Q, and the next N lines give the initial values of A_i. Finally, the next Q lines each contain a query of the form "M i X" or "C P Q X".

Output

Print the answer to each 'C' query, one per line.

Example

Input:
```
4 6
3
4
1
7
C 2 4 4
M 4 1
C 2 4 4
C 1 4 5
M 2 10
C 1 3 9
```

Output:
```
2
3
4
2
```

FJ has 4 cows, whose initial numbers are 3, 4, 1, and 7. The cows then give him 6 operations; the first asks him to count the how many of the last three cows have a number at most 4, the second asks him to change the fourth cow's number to 1, etc.

Warning: large input/output data.

Added by: Neal Wu
Date: 2008-10-25
Time limit: 1s-8s
Source limit: 50000B
Languages: All

SPOJ Problem Set (classical)

3305. Roman Patrollers

Problem code: SA04C

In ancient times, patrollers were used to ensure that all the cities of the Roman Empire were under control. A patroller's job consisted in continuously visiting the cities of the empire, trying to minimise the interval between two visits to each city. The Military Society (MS) wants to simulate the behavior of one such patroller to see how effective they were.

Each cycle of the simulation corresponds to one time unit. The instantaneous city idleness (ICI) for a city X after T cycles of the simulation is the number of cycles elapsed since the last visit of the patroller to the city X (i.e. the number of time units the city X remained unvisited). All of the cities have initial instantaneous city idleness equal to zero at the start of the simulation. The instantaneous empire idleness (IEI) after each given cycle is the sum of the instantaneous city idleness of all cities after that given cycle. Finally, the empire idleness (EI) for an N-cycle simulation is the sum of the instantaneous empire idleness after each of the N cycles of simulation.

After visiting a city X, the patroller always chooses to visit the neighbour city Y with the highest instantaneous city idleness (if more than one city has the highest idleness, the one with the lowest identifier is chosen). Cities X and Y are neighbour if there is a road linking the two cities directly, without going through any intermediate city. In the beginning of the simulation, the patroller is located in one of the cities, and is given a map of the Roman Empire containing a description of all the roads in the empire, indicating the length (in kilometers) and which two cities each road connects. A road between cities X and Y can be used both to go from X to Y and from Y to X.

Assuming that a patroller travels one kilometer in one time unit (one simulation cycle) and that the time to visit a city is negligible (equal to zero), MS asks you to determine the empire idleness after an N-cycle simulation.

For clarity, consider the example of an empire which contains 3 cities (1, 2 and 3) and two roads of length 1 km. The first road connects cities 1 and 2, while the second road connects cities 2 and 3. Below you find a trace of a 3-cycle simulation for such a scenario, considering that the patroller starts at city 1.

Start of the simulation
Patroller at: 1
ICI1 = 0, ICI2 = 0, ICI3 = 0
IEI = 0
EI = 0
After cycle 1
Patroller at: 2
ICI1 = 1, ICI2 = 0, ICI3 = 1
IEI = 2
EI = 2
After cycle 2
Patroller at: 1
ICI1 = 0, ICI2 = 1, ICI3 = 2
IEI = 3
EI = 5
After cycle 3 Patroller at: 2
ICI1 = 1, ICI2 = 0, ICI3 = 3

IEI = 4
EI = 9
Therefore, for such a scenario, after 3 simulation cycles the empire idleness is 9.

Input

The input consists of several test cases. The first line of a test case contains four integers C,R,N, and S, indicating respectively the quantity of cities in the empire (2 . C . 1000), the number of roads (1 . R . C(C - 1)/2), the number of cycles to be simulated (1 . N . 1000) and the identifier of the starting city of the patroller (1 . S . C). Each city is identified by a distinct integer from 1 to C. Each of the following R lines contains three integers X, Y and D describing a road; X and Y represent cities (1 . X 6= Y . C) and D represents the distance (1 . D . 1000), in kilometers, of the road that connects X and Y directly, without passing through any other city. Each pair of cities X and Y will appear at most once in a road description. You can assume that it is always possible to travel from any city to any other city in the empire using the roads available. The end of input is indicated by C = R = N = S = 0.

Output

For each test case in the input, your program must produce one line containing the empire idleness after the N-cycle simulation.

Example

Input:

```
2 1 1 1
1 2 2
2 1 2 1
1 2 2
2 1 3 1
1 2 2
2 1 4 1
1 2 2
3 2 3 1
1 2 1
2 3 1
0 0 0 0
```

Output:

```
2
4
8
10
9
```

Added by: Daniel Gómez Didier
Date: 2008-11-07
Time limit: 1s-3s
Source limit:50000B
Languages: All
Resource: ACM International Collegiate Programming Contest 2004 - Brazil Sub-Regional

SPOJ Problem Set (classical)

3306. Very Special Boxes

Problem code: SA04D

Special Box Company (SBC) is a small family-owned and family-run business which produces decorated carton boxes for wrapping gifts. The boxes are hand-made, produced individually from fine materials. When accepting an order from a client, they always produce a few more boxes than needed, to keep a stock of boxes to be sold in the future, if needed. Over the years their stock has been growing, with boxes all over the place, and they decided they needed to organize it a bit more. They have therefore made a list registering the dimensions of every box in their stock.
SBC has just received an order from a client that must be delivered tomorrow, so there is no time to produce new boxes. The client wants a certain number N of boxes all of the same size; each box will be used to pack one item of dimensions X, Y and Z. As the carton used in the boxes is very thin, you may assume that a box of size (X, Y, Z) would fit perfectly the item the client wants to wrap. If there are not at least N boxes that fit perfectly, the client wants N boxes that fit the items as tightly as possible. The box size that fits the items as tightly as possible is the one which minimizes the empty space when the item is put inside the box. An item can be rotated in any direction to be accomodated inside a box; therefore, a box of size(X, Y, Z) is as good as a box of size (Y, Z, X), for example.
Can you help SBC finding whether they can fulfill the customer order?

Input

The input consists of several test cases. The first line of a test case contains two integers N and M, indicating respectively the number of boxes the client needs to buy (1 <= N <= 1500) and the number of boxes in the stock list (1 <= M <= 1500). The second line contains three integers X, Y and Z, representing the dimensions of the item the client wants to wrap (0 < X, Y, Z <= 50).
Each of the next M lines contains three integers A, B and C representing the dimensions of a box in the stock list (0 < A, B, C <= 50). A test case with N = 0 indicates the end of the input.
The input must be read from standard input.

Output

For each test case in the input your program must produce one line, containing either:
-> The single word ‘impossible’, in case it is not possible to fulfill the client’s order (because there are not at least N boxes of the same size in stock that can contain the item); or
-> one integer V , which specifies the volume of empty space left when one of the N items packed in one of the boxes chosen.

Example

Input:

```
1 1
2 4 3
2 3 4
2 6
3 1 3
7 4 7
10 8 2
```

```
2 8 10
6 2 9
7 7 4
6 2 9
1 1
3 3 3
1 1 1
0 0
```

Output:

```
0
99
impossible
```

Added by: Daniel Gómez Didier
Date: 2008-11-07
Time limit: 1s-3s
Source limit: 50000B
Languages: All
Resource: ACM International Collegiate Programming Contest 2004 - Brazil Sub-Regional

SPOJ Problem Set (classical)

3307. Hex Tile Equations

Problem code: HEXTILE

An amusing puzzle consists of a collection of hexagonal tiles packed together with each tile showing a digit or '=' or an arithmetic operation '+', '-', '*', or '/'. Consider continuous paths going through each tile exactly once, with each successive tile being an immediate neighbor of the previous tile. The object is to choose such a path so the sequence of characters on the tiles makes an acceptable equation, according to the restrictions listed below. A sequence is illustrated in each figure above. In Figure 1, if you follow the gray path from the top, the character sequence is"6/3=9-7". Similarly, in Figure 2, start from the bottom left 3 to get "3*21+10=73".

There are a lot of potential paths through a moderate sized hex tile pattern. A puzzle player may get frustrated and want to see the answer. Your task is to automate the solution.

The arrangement of hex tiles and choices of characters in each puzzle satisfy these rules:

1. The hex pattern has an odd number of rows greater than 2. The odd numbered rows will all contain the same number of tiles. Even numbered rows will have one more hex tile than the odd numbered rows and these longer even numbered rows will stick out both to the left and the right of the odd numbered rows.
2. There is exactly one '=' in the hex pattern.
3. There are no more than two '*' characters in the hex pattern.
4. There will be fewer than 14 total tiles in the hex pattern.
5. With the restrictions on allowed character sequences described below, there will be a unique acceptable solution in the hex pattern.

To have an acceptable solution from the characters in some path, the expressions on each side of the equal sign must be in acceptable form and evaluate to the same numeric value. The following rules define acceptable form of the expressions on each side of the equal sign and the method of expression evaluation:

6. The operators '+', '-', '*', and '/' are only considered as binary operators, so no character sequences where '+' or '-' would be a unary operator are acceptable. For example "-2*3=-6" and "1 =5+-4" are not acceptable.
7. The usual precedence of operations is not used. Instead all operations have equal precedence and operations are carried out from left to right. For example "44-4/2=2+3*4" is acceptable and "14=2+3*4" is not acceptable.
8. If a division operation is included, the equation can only be acceptable if the division operation works out to an exact integer result. For example "10/5=12/6" and "7+3/5=3*4/6" are acceptable. "5/2*4=10" is not acceptable because the sides would only be equal with exact mathematical calculation including an intermediate fractional result. "5/2*4=8" is not acceptable because the sides of the equation would only be equal if division were done with truncation.
9. At most two digits together are acceptable. For example, "123+1 = 124" is not acceptable.
10. A character sequences with a '0' directly followed by another digit is not acceptable. For example, "3*05=15" is not acceptable.

With the assumptions above, an acceptable expression will never involve an intermediate or final arithmetic result with magnitude over three million.

Input

The input will consist of one to fifteen data sets, followed by a line containing only 0.

The first line of a dataset contains blank separated integers rc, where r is the number of rows in the hex pattern and c is the number of entries in the odd numbered rows. The next r lines contain the characters on the hex tiles, one row per line. All hex tile characters for a row are blank separated. The lines for odd numbered rows also start with a blank, to better simulate the way the hexagons fit together. Properties 1-5 apply.

Output

There is one line of output for each data set. It is the unique acceptable equation according to rules 6-10 above. The line includes no spaces.

Example

Input:

```
5 1
 6
/ 3
 =
9 -
 7
3 3
 1 + 1
* 2 0 =
 3 3 7
5 2
 9 -
* 2 =
 3 4
+ 8 3
 4 /
0
```

Output:

```
6/3=9-7
3*21+10=73
8/4+3*9-2=43
```

Added by: Nikola P Borisov
Date: 2008-11-07
Time limit: 20s
Source limit: 50000B
Languages: All
Resource: Mid-Central Regional ACM-ICPC Contest 2008

SPOJ Problem Set (classical)

3308. The Bridges of San Mochti

Problem code: BRIDGES2

You work at a military training facility in the jungles of San Motchi. One of the training exercises is to cross a series of rope bridges set high in the trees. Every bridge has a maximum capacity, which is the number of people that the bridge can support without breaking. The goal is to cross the bridges as quickly as possible, subject to the following tactical requirements:

One unit at a time!

If two or more people can cross a bridge at the same time (because they do not exceed the capacity), they do so as a unit; they walk as close together as possible, and they all take a step at the same time. It is never acceptable to have two different units on the same bridge at the same time, even if they don't exceed the capacity. Having multiple units on a bridge is not tactically sound, and multiple units can cause oscillations in the rope that slow everyone down. This rule applies even if a unit contains only a single person.

Keep moving!

When a bridge is free, as many people as possible begin to cross it as a unit. Note that this strategy doesn't always lead to an optimal overall crossing time (it may be faster for a group to wait for people behind them to catch up so that more people can cross at once). But it is not tactically sound for a group to wait, because the people they're waiting for might not make it, and then they've not only wasted time but endangered themselves as well.

Periodically the bridges are reconfigured to give the trainees a different challenge. Given a bridge configuration, your job is to calculate the minimum amount of time it would take a group of people to cross all the bridges subject to these requirements.

For example, suppose you have nine people who must cross two bridges: the first has capacity 3 and takes 10 seconds to cross; the second has capacity 4 and takes 60 seconds to cross. The initial state can be represented as (900), meaning that 9 people are waiting to cross the first bridge, no one is waiting to cross the second bridge, and no one has crossed the last bridge. At 10 seconds the state is (630). At 20 seconds the state is (33/3:50/0), where /3:50/ means that a unit of three people is crossing the second bridge and has 50 seconds left. At 30 seconds the state is (06/3:40/0); at 70 seconds it's (063); at 130 seconds it's (027); and at 190 seconds it's (009). Thus the total minimum time is 190 seconds.

Input

The input consists of one or more bridge configurations, followed by a line containing two zeros that signals the end of the input. Each bridge configuration begins with a line containing a negative integer -B and a positive integer P, where B is the number of bridges and P is the total number of people that must cross the bridges. Both B and P will be at most 20. (The reason for putting -B in the input file is to make the first line of a configuration stand out from the remaining lines.) Following are B lines, one for each bridge, listed in order from the first bridge that must be crossed to the last. Each bridge is defined by two positive integers C and T, where C is the capacity of the bridge (the maximum number of people the bridge can hold), and T is the time it takes to cross the bridge (in seconds). C will be at most 5, and T will be at most 100. Only one unit, of size at most C, can cross a bridge at a time; the

time required is always T, regardless of the size of the unit (since they all move as one). The end of one bridge is always close to the beginning of the next, so the travel time between bridges is zero.

Output

For each bridge configuration, output one line containing the minimum amount of time it will take (in seconds) for all of the people to cross all of the bridges while meeting both tactical requirements.

Example

Input:

```
-1 2
5 17
-1 8
3 25
-2 9
3 10
4 60
-3 10
2 10
3 30
2 15
-4 8
1 8
4 30
2 10
1 12
0 0
```

Output:

```
17
75
190
145
162
```

Added by: Nikola P Borisov
Date: 2008-11-07
Time limit: 10s
Source limit:50000B
Languages: All
Resource: Mid-Central Regional ACM-ICPC Contest 2008

SPOJ Problem Set (classical)

3309. Bulletin Board

Problem code: BULLETIN

The ACM Student Chapter has just been given custody of a number of school bulletin boards. Several members agreed to clear off the old posters. They found posters plastered many levels deep. They made a bet about how much area was left clear, what was the greatest depth of posters on top of each other, and how much of the area was covered to this greatest depth. To determine each bet's winner, they made very accurate measurements of all the poster positions as they removed them. Because of the large number of posters, they now need a program to do the calculations. That is your job.

A simple illustration is shown above: a bulletin board 45 units wide by 40 high, with three posters, one with corners at coordinates (10, 10) and (35, 20), another with corners at (20, 25) and (40, 35), and the last with corners at (25, 5) and (30, 30). The total area not covered by any poster is 1300. The maximum number of posters on top of each other is 2. The total area covered by exactly 2 posters is 75.

Input

The input will consist of one to twenty data sets, followed by a line containing only 0. On each line the data will consist of blank separated nonnegative integers.

The first line of a dataset contains integers n w h, where n is the number of posters on the bulletin board, w and h are the width and height of the bulletin board. Constraints are 0 < n <= 100;0 < w<= 50000; 0 < h <= 40000.

The dataset ends with n lines, each describing the location of one poster. Each poster is rectangular and has horizontal and vertical sides. The x and y coordinates are measured from one corner of the bulletin board. Each line contains four numbers xl yl xh and yh, where xl and yl, are the lowest values of the x and y coordinates in one corner of the poster and xh and yh are the highest values in the diagonally opposite corner. Each poster fits on the bulletin board, so 0 <= xl < xh <= w, and 0 <= yl < yh <= h.

Output

There is one line of output for each data set containing three integers, the total area of the bulletin board that is not covered by any poster, the maximum depth of posters on top of each other, and the total area covered this maximum number of times.

Caution: An approach examining every pair of integer coordinates might need to deal with 2 billion coordinate pairs.

Example

Input:

```
3 45 40
10 10 35 20
20 25 40 35
25 5 30 30
1 20 30
5 5 15 25
2 2000 1000
0 0 1000 1000
1000 0 2000 1000
3 10 10
0 0 10 10
0 0 10 10
0 0 10 10
0
```

Output:

```
1300 2 75
400 1 200
0 1 2000000
0 3 100
```

Added by: Nikola P Borisov
Date: 2008-11-07
Time limit: 10s
Source limit: 50000B
Languages: All
Resource: Mid-Central Regional ACM-ICPC Contest 2008

SPOJ Problem Set (classical)

3310. Serial Numbers

Problem code: SERIALN

A manufacturer keeps an ordered table of serial numbers by listing in each row of the table a range of serial numbers along with two corresponding pieces of information called the status code and the transfer code. A four-column table stores information about ranges of serial numbers in this order: starting serial number, ending serial number, status code, transfer code. Serial numbers as well as transfer codes are integers from 1 to 2^{31}-1 (2^{31}-1 = 2147483647), and status codes are a single upper-case letter. The table is maintained in increasing order of serial numbers, serial number ranges are never allowed to overlap, and for any given serial number, the table must always accurately represent the most recent data (status code and transfer code) for that serial number.

Let's say that 100,000 serial numbers are created with a status of "A" and a transfer code of "1". An entry for those serial numbers might look like this:

```
1 100000 A 1
```

This is obviously far more efficient than storing 100,000 individual rows all with the same status and transfer codes. The challenge arises when serial numbers within already defined ranges need to be given different status or transfer codes. For example, if serial number 12345 needs to change to status B, the above table would need to become three separate entries:

```
1 12344 A 1
12345 12345 B 1
12346 100000 A 1
```

Now let's change the transfer code of all serial numbers in the range 12000 to 12999 to 2. This gets us:

```
1 11999 A 1
12000 12344 A 2
12345 12345 B 2
12346 12999 A 2
13000 100000 A 1
```

Now change all existing serial numbers from 10000 to 100000 to status C and transfer code 2:

```
1 9999 A 1
10000 100000 C 2
```

Once created a serial number will never be deleted, but it is possible to have ranges of undefined serial numbers between ranges of defined ones. To demonstrate, let's now set all serial numbers from 1000000 to 1999999 to status Z and transfer code 99:

```
1 9999 A 1
10000 100000 C 2
1000000 1999999 Z 99
```

Finally, the table is always maintained with a minimal number of rows, meaning specifically that there will never be two adjacent rows in the table where one would suffice. For example, consider the following serial number table:

```
1 10 A 1
11 20 A 1
21 30 B 1
```

The first two rows could actually be represented by a single row, meaning that the table above does not have a minimal number of rows. The same data represented by a minimal number of rows would look like this:

```
1 20 A 1
21 30 B 1
```

The following table, however, because the first two rows have non-matching transfer codes, already contains the minimal number of rows:

```
1 10 A 1
11 20 A 2
21 30 B 1
```

Similarly, the following table cannot be reduced further because the first two rows do not represent a continuous series of serial numbers:

```
1 10 A 1
12 20 A 1
21 30 B 1
```

Input

Each input case begins with a single line that is a character string naming the test case. This string contains at most 80 characters. The name "END" marks the end of the input. Following this will be 1 to 100 lines of the form "A B S T", where A, B, and T are integers in the range 1 to 2^{31}-1, S is an uppercase letter, and A<=B. These lines are, in the order they are to be applied, the serial number transactions to be recorded, where A is the start of the serial number range, B is the end of the serial number range, S is the status code, and T is the transfer code. The list of serial number transactions is terminated by a line containing only a 0 (zero) character.

Output

For each input case, echo the test case name to the output on a line by itself, followed by the resulting minimal-rows serial number table that results after all serial number transactions have been applied.

Example

Input:

```
First Example
1 100000 A 1
12345 12345 B 1
0
And Another
1 100000 A 1
12345 12345 B 1
12000 12999 A 2
```

```
12345 12345 B 2
0
Test Case Three
1 100000 A 1
12345 12345 B 1
12000 12999 A 2
12345 12345 B 2
10000 100000 C 2
0
Example Four
1 100000 A 1
12345 12345 B 1
12000 12999 A 2
12345 12345 B 2
10000 100000 C 2
1000000 1999999 Z 99
0
Example 5
1 10 A 1
21 30 B 1
11 20 A 1
0
Example 6
21 30 B 1
1 10 A 1
11 20 A 2
0
Example 7
12 20 A 1
21 30 B 1
1 10 A 1
0
END
```

Output:

```
First Example
1 12344 A 1
12345 12345 B 1
12346 100000 A 1
And Another
1 11999 A 1
12000 12344 A 2
12345 12345 B 2
12346 12999 A 2
13000 100000 A 1
Test Case Three
1 9999 A 1
10000 100000 C 2
Example Four
1 9999 A 1
10000 100000 C 2
1000000 1999999 Z 99
Example 5
1 20 A 1
21 30 B 1
Example 6
1 10 A 1
11 20 A 2
21 30 B 1
```

```
Example 7
1 10 A 1
12 20 A 1
21 30 B 1
```

Added by: Nikola P Borisov
Date: 2008-11-07
Time limit: 10s
Source limit: 50000B
Languages: All
Resource: Mid-Central Regional ACM-ICPC Contest 2008

SPOJ Problem Set (classical)

3314. Umnozak

Problem code: UMNOZAK

The digit-product of a positive integer is the product of the number's decimal digits. For example, the digit-product of 2612 is 2 . 6 . 1 . 2 = 24.

The self-product of a number is the product of the number and its digit-product. For example, the self-product of 2612 is 2612 . 24 = 62688.

Write a program that, given two positive integers A and B ($1 <= A <= B < 10^{18}$), calculates the number of positive integers whose self-product is between A and B, inclusive.

Input

The first line of input contains the integer T (1 <= T <= 20). The next T lines each contain a pair of integers A and B.

Output

For each test case, print a line with the number of positive integers whose self-product is between A and B.

Example

Input:
```
3
20 30
145 192
2224222 2224222
```

Output:
```
2
4
1
```

For the second example, the self-products of the numbers 19, 24, 32, and 41 are 171, 192, 192 and 164, respectively.

Added by: Neal Wu
Date: 2008-11-08
Time limit: 10s
Source limit: 50000B
Languages: All
Resource: Croatian Olympiad in Informatics 2008

SPOJ Problem Set (classical)

3322. Doubled Numbers

Problem code: DOUBLE

Some numbers have a curious property: when rotating the number to the right, the new number is the double of the original. Rotating the number to the right means choosing the last digit and moving it to the beginning of the number, as the leftmost digit. For example, the number 421052631578947368 when rotated to the right gives 842105263157894736, which is the double of the original. These numbers are expressed in the decimal system. In any numeral system such numbers exist, for example, in the binary (base 2) numeral system, numbers 01 and 0101 have this property. Note that leading zeros are required in this case.

Write a program that, for any given base B (2 <= B <= 250), finds the smallest number in that numeral system which has this property. Try not to precompute the numbers, the solution is very easy and pretty.

Input

The input consists of several test cases. The first line contains an integer T (T <= 20), the number of test cases. The following T lines contain one number B, each.

Output

For each number B, output one or more numbers separated by a blank space that represent the digits in base B (written as decimal numbers) of the smallest number which has this property.

Example

Input:

```
3
2
10
35
```

Output:

```
0 1
0 5 2 6 3 1 5 7 8 9 4 7 3 6 8 4 2 1
11 23
```

Output explanation

```
In  example #1
The initial number (when converted to decimal system):
0 * 2^1 + 1 * 2^0 = 1
After applying the rotation:
1 * 2^1 + 0 * 2^0  = 2.

In example #3
```

```
The initial number (when converted to decimal system):
11 * 35^1 + 23 * 35^0  =  408
After applying the rotation:
23 * 35^1 + 11 * 35^0  = 816.
```

Added by: Reinier César Mujica Hdez
Date: 2008-11-10
Time limit: 2s
Source limit: 3072B
Languages: All
Resource: OCI Olimpiads Cuban in Informatics 2008 day 1

SPOJ Problem Set (classical)

3347. Cestarine

Problem code: HIGHWAY

In a single day, N of Luka's trucks travel a specific highway. The highway has a number of exits and entrances. An exit with a particular number is in the same location as the entrance with that number.

Upon entering the highway, a truck driver receives a ticket which indicates the entrance he used. When exiting, the driver pays a toll equal to the absolute difference of the entrance and exit numbers. For example, if a ticket says he used entrance 30, then exiting at exit 12 will cost him 18.

Luka has figured out a way to save toll money that his company daily spends. Any two drivers can meet on the highway and exchange tickets, even if their routes don't overlap. Tickets can be exchanged an arbitrary number of times.

However, a driver cannot use an exit if his ticket says he used the same entrance, since that would be suspicious.

Write a program that calculates the least total amount of tolls that the drivers can achieve by exchanging tickets.

Input

On the first line of the input is the integer T (1 <= T <= 5), the number of test cases. T cases follow, each beginning with the single integer N (2 <= N <= 100,000). Each of the next N lines contains two integers between 1 and 1,000,000,000 inclusive, representing the entrance and exit numbers of a truck. Note that no two trucks will have the same entrance or exit numbers.

Output

For each test case, output the least total amount of tolls Luka's company must pay.

Example

Input:

```
2
3
3 65
45 10
60 25
3
5 5
6 7
8 8
```

Output:

```
32
5
```

In the first example, the first and third drivers will exchange tickets. After this, the second and third drivers exchange tickets. After this, the drivers will have the tickets 60, 3, 45, respectively. The total amount in tolls is |65 - 60| + |10 - 3| + |25 - 45| = 32.

Warning: large input/output data.

Added by: Neal Wu
Date: 2008-11-13
Time limit: 1s-8s
Source limit:50000B
Languages: All
Resource: Croatian Open 07/08 - Contest 6

SPOJ Problem Set (classical)

3359. Stack

Problem code: STACK

Alan loves to construct a stack of building bricks. His stack consists of many cuboids with square base. All cuboids have the same height 1. Alan puts the consecutive cuboids one over another.

Recently in math class, the concept of volume was introduced to Alan. Consequently, he wants to compute the volume of his stack now. The lengths of cuboids bases (from top to bottom) are constructed by Alan in the following way:

- Length of edge of the first square is one. i.e. $\mathbf{a_1} = 1$.
- Next, Alan fixes the length of the edge of the second square $\mathbf{a_2}$.
- Next, Alan calculates the length $\mathbf{a_n}$ (n > 2) by $\mathbf{2*a_2*a_{n-1} - a_{n-2}}$. Do not ask why he chose such a formula; let us just say that he is a really peculiar young fellow.

For example, if Alan fixes $\mathbf{a_2} = 2$, then $\mathbf{a_3} = 7$. If Alan fixes $\mathbf{a_2} = 1$, then $\mathbf{a_n} = 1$ holds for all n.

Now Alan wonders if he can calculate the volume of stack of **N** consecutive building bricks. Help Alan and write the program that computes this volume. Since it can be quite large, it is enough to compute the answer modulo given natural number **m**.

Input

The input contains several test cases. The first line contains the number t (t <= 100000) denoting the number of test cases. Then t test cases follow. Each of them is given in a separate line containing three integers $\mathbf{a_2}$, **N**, **m** (1 <= $\mathbf{a_2}$, **m** <= 10^9, 2 <= **N** <= 10^9) separated by a single space.

Output

For each test case compute the volume of stack of **N** consecutive bricks constructed by Alan according to steps 1 to 3 and output its remainder modulo **m**.

Example

Input:
```
3
2 3 100
1 4 1000
3 3 1000000000
```

Output:
```
54
4
299
```

Warning: large input/output data, be careful with certain languages

Warning: A naive algorithm won't terminate in even 2 minutes.

Added by: Blue Mary
Date: 2008-11-15
Time limit: 43s
Source limit: 50000B
Languages: All except: C99 strict
Resource: ACM Central European Programming Contest, Wrocław 2008

SPOJ Problem Set (classical)

3360. Digital Image Recognition

Problem code: IMGREC2

Description

According to Wikipedia, image processing is any form of signal processing for which the input is an image, such as photographs or frames of video; the output of image processing can be either an image or a set of characteristics or parameters related to the image. Most image-processing techniques involve treating the image as a two-dimensional signal and applying standard signal-processing techniques to it.

The task you are facing here is a relatively easy one (compared to our general conception of image processing!). Given a black-and-white image of size R * C with some digits (and possibly other shapes) on it, your program needs to figure out the digits written on the image. Specifically, the digits drawn on the graph will adhere to the following rules:

1) Digits are drawn with a series of *strokes*. A **stroke** can be regarded as a **rectangle** of any size on the image, and its edges will always be parallel to either **x-axis** or **y-axis**. The number of strokes required to draw each digit will be exactly as follows:

0	1	2	3	4	5	6	7	8	9
4	1	5	4	3	5	5	2	5	5

Refer to the **figure below** if you are unclear about how the digits are drawn.

2) Although the **width** of strokes used to draw a digit might be **different**, the **outer shapes of digits** will strictly follow those specified in the **figure below**.

3) In order for a digit to be recognizable, **all** parts (**strokes** and **joints**) presented in the graph below must also be clearly **distinguishable** in the image.

(Refer to the last sample test case if you are unsure about this requirement; in that test case, when the middle stroke of 2 is omitted, the number should not be considered as recognizable.)

4) You may assume that the image is not rotated, and there is **no noise** in the input.

[IMAGE]

Please output the sum of digits recognizable in the graph. In the case that no characters is recognizable, please output 0 instead.

Input

There are multiple test cases in the input file.

Each test case starts with two integers, R and C (1 <= R,C <= 500), specifying the number of rows / columns of the graph. Each of the following R lines contains consecutive C characters ("0" or "1"), describing the image to be processed.

Two successive test cases are separated by a blank line. A case with R = 0, C = 0 indicates the end of the input file, and should not be processed by your program.

Output

For each test case, please print a single integer, the sum of recognizable numbers. See the sample output for format details.

Example

Input:

```
5 12
001101011111
000101000011
000101001111
001101000011
000000000111

5 3
111
010
110
010
110

6 14
11111000011111
11001000000011
11111001000000
11111001001110
11001011001010
11111000001110

5 2
11
01
11
01
11

6 9
111100111
000100001
000100011
011100010
010000011
011110000

0 0
```

Output:

```
Case #1: 4
```

```
Case #2: 0
Case #3: 15
Case #4: 3
Case #5: 2
```

Added by: Blue Mary
Date: 2008-11-15
Time limit: 3s
Source limit: 50000B
Languages: All except: C99 strict
Resource: ACM/ICPC Asian Regional Contest, Hangzhou 2008

SPOJ Problem Set (classical)

3363. Svada

Problem code: SVADA

The local zoo has acquired a large open garden in which animals may freely move as in their natural habitats and entertain visitors with their usual shenanigans.

The most popular animals are monkeys. With their climbing and jumping and other skills, they delight old and young visitors alike.

One species of monkey has specialized in climbing tall trees and picking off coconuts. Another species has specialized in breaking them open.

There are N monkeys of the first type (numbered 1 through N) and M monkeys of the second type (numbered 1 through M).

Monkey k of the first type takes Ak seconds to find a good spot on the tree, after which it picks off its first coconut. After that the monkey produces a new coconut every Bk seconds.

Monkey k of the second type takes Ck seconds to find a good tool for opening the coconuts, after which it opens its first coconut. After that the monkey opens another coconut every Dk seconds.

Unfortunately, the second type of monkey is extremely aggressive so the two types may not be in the garden at the same time. Therefore, zoo keepers will chase away the first type of monkeys as soon as they have picked off all the coconuts. Similarly, if monkeys of the same type stay too long after opening all the coconuts, fights will ensue. Because of that, zoo keepers will send them away as soon as they have opened all the coconuts.

The zoo keepers first arrive immediately after all coconuts have been picked, and again immediately after the monkeys open them all. The time needed for monkeys to enter or leave the garden is also negligibly small.

Tomislav especially likes the second type of monkey, but can never guess when to arrive in order to see them. Help him calculate the time when the second type arrives if he knows the total time that monkeys spent in the garden, but does not know the number of coconuts in the garden.

Input

The first line contains the integer T (1 <= T <= 1 000 000 000), the total time that monkeys spent in the garden, in seconds.

The next line contains the integer N (1 <= N <= 100), the number of monkeys of the first type.

Each of the following N lines contains two integers Ak and Bk (1 <= Ak, Bk <= 1 000 000 000), how fast monkey k of the first type is.

The next line contains the integer M (1 <= M <= 100), the number of monkeys of the second type.

Each of the following M lines contains two integers Ck and Dk (1 <= Ck, Dk <= 1 000 000 000), how fast monkey k of the second type is.

Output

Output the number of seconds between the arrival of the first type of monkeys and the arrival of the second type.

Example

Input:
```
20
2
3 2
1 3
3
3 1
4 1
5 1
```

Output:
```
13
```

Added by: Race with time
Date: 2008-11-16
Time limit: 1s
Source limit:50000B
Languages: All
Resource: COCI 2008-2009

SPOJ Problem Set (classical)

3372. Round Table

Problem code: ROUNDT

We have a round table with 2*N seats (1<= N <= 2000). The seats are numbered with the numbers from 1 to 2*N in order. This round table is in a round room with two doors. Door 1 is between seats 1 and 2N on the table, and door 2 is between seats N and N+1. We have 2N guests that we want to seat. Each guest has a id from 1 to 2N and we want to seat him/her exactly at the seat corresponding to his id. All the guest are split into two groups of N and each group is waiting in front of a door in a line. This room unfortunately has the problem that if you let a person go from door 1 and he/she has to seep at position p, to get to his seat he has to make either all people seating on seats from 1 to p-1 or all people seating on seats p+1 to 2N to get up to let him in. This presents you with the problem of how to let the guests in to cause minimum number of getting ups. The only thing you control is from which door to let the next guest. The guests that are waiting in front of door 1 can only get in from door 1 the same is true with door 2.

Input

On the first line there will one integer - N. Two more lines of input follow each with N integers in the range [1, 2N] - the guests waiting in front of door 1 and door 2 respectively. The first people in the list is the first to enter the room.

Output

Single integer the total min number of stand-ups that will happen if we choose the best sequence of letting guest in.

Example

Input:
```
3
4 5 3
6 2 1
```

Output:
```
3
```

Added by: Nikola P Borisov
Date: 2008-11-17
Time limit: 1s-10s
Source limit:50000B
Languages: All

SPOJ Problem Set (classical)

3373. Permutation Code

Problem code: PERMCODE

As the owner of a computer forensics company, you have just been given the following note by a new client:

I, Albert Charles Montgomery, have just discovered the most amazing cypher for encrypting messages. Let me tell you about it.

To begin, you will need to decide on a set of symbols, call it S, perhaps with the letters RATE. The size of this set must be a power of 2 and the order of the symbols in S is important. You must note that R is at position 0, A at 1, T at 2, and E at 3. You will also need one permutation P of all those symbols, say TEAR. Finally you will need an integer, call it x. Together, these make up the key. Given a key, you are now ready to convert a plaintext message M of length n (M[0], M[1]... M[n-1]), that has some but not necessarily all of the symbols in S, into a cyphertext string C, also of length n (C[0], C[1],...C[n-1]), that has some but not necessarily all of the symbols in S.

The encrypting algorithm computes C as follows:

1. Calculate an integer d as the remainder after dividing the integer part of ($n^{1.5}$ + x) by n. This can be expressed more succinctly as d = (int)($n^{1.5}$ + x) % n, where "%" is the remainder operator.
2. Set C[d] to be the symbol in S whose position is the same as the position of M[d] in P.
3. For each j != d in 0..n-1, set C[j] to be the symbol in S whose position is the value obtained by xor-ing the position of M[j] in P with the position of M[(j+1) % n] in S. Note that the bitwise xor operator is "^" in C, C++, and Java.

For example, consider this scenario where S=RATE, P=TEAR, x=102, M=TEETER, and n=6. To compute d, first calculate $6^{1.5}$ + 102 = 116.696938, then take the remainder after dividing by 6. So d = 116 % 6 = 2. The following table shows the steps in filling in the cyphertext C. Note that the order of the steps is not important.

	0	1	2	3	4	5	
S =	R	A	T	E			
P =	T	E	A	R			
M =	T	E	E	T	E	R	
C =	E						M[0] is T, T is at P[0]. M[1] is E, E is at S[3]. C[0] = S[0 xor 3] = S[3]
	E	T					M[1] is E, E is at P[1]. M[2] is E, E is at S[3]. C[1] = S[1 xor 3] = S[2]
	E	T	A				2 is d. M[2] is E, E is at P[1], so C[2] = S[1]
	E	T	A	E			M[3] is T, T is at P[0]. M[4] is E, E is at S[3]. C[3] = S[0 xor 3] = S[3]
	E	T	A	E	A		M[4] is E, E is at P[1]. M[5] is R, R is at S[0]. C[4] = S[1 xor 0] = S[1]
	E	T	A	E	A	A	M[5] is R, R is at P[3]. M[0] is T, T is at S[2]. C[5] = S[3 xor 2] = S[1]

I have included additional examples of encrypted messages at the end of this note for you to experiment with. However, first, I need to tell you about the decryption algorithm.

Unfortunately, the next page of the note, with the decrypting algorithm, is completely unreadable because it is covered with huge, overlapping, messy ink blots. Given your considerable skill in unravelling puzzles, your task is to write the decoder based on your knowledge of the encoding algorithm.

Input

The input for the decoder consists of one or more sets of {key, encrypted message} pairs. The key is on 3 separate lines. The first line contains the single integer x, 0 < x < 10,000; the second line contains the string S; and the third line contains the string P, which will be a permutation of *S*. The length of S (and therefore P) will always be one of the following powers of two: 2, 4, 8, 16, or 32. Following the key is a line containing the encrypted message string C, which will contain at least one and at most sixty characters. The strings S, P, and C will not contain whitespace, but may contain printable characters other than letters and digits. The end of the input is a line which contains the single integer 0.

Output

For each input set print the decrypted string on a single line, as shown in the sample output.

Example

Input:

```
102
RATE
TEAR
ETAEAA
32
ABCDEFGHIJKLMNOPQRSTUVWXYZ._!?,;
;ABCDEFGHIJKLMNOPQRSTUVWXYZ._!?,
MOMCUKZ,ZPD
1956
ACEHINT_
ACTN_IHE
CIANCTNAAIECIA_TAI
0
```

Output:

```
TEETER
HELLO_WORLD
THE_CAT_IN_THE_HAT
```

Added by: Nikola P Borisov
Date: 2008-11-17
Time limit: 1s
Source limit: 50000B
Languages: All
Resource: Mid-Central Regional ACM-ICPC Contest 2004

SPOJ Problem Set (classical)

3374. Scavenger Hunt

Problem code: SCAVHUNT

Bill has been the greatest boy scout in America and has become quite a superstar because he always organized the most wonderful scavenger hunts (you know, where the kids have to find a certain route following certain hints). Bill has retired now, but a nationwide election quickly found a successor for him, a guy called George. He does a poor job, though, and wants to learn from Bill's routes. Unfortunately Bill has left only a few notes for his successor. Bill never wrote his routes completely, he only left lots of little sheets on which he had written two consecutive steps of the routes. He then mixed these sheets and memorized his routes similarly to how some people learn for exams: practicing again and again, always reading the first step and trying to remember the following. This made much sense, since one step always required something from the previous step. George however would like to have a route written down as one long sequence of all the steps in the correct order. Please help him make the nation happy again by reconstructing the routes.

Input

The first line contains the number of scenarios. Each scenario describes one route and its first line tells you how many steps (3 <= S <= 333) the route has. The next S - 1 lines each contain one consecutive pair of the steps on the route separated by a single space. The name of each step is always a single string of letters.

Output

The output for every scenario begins with a line containing "Scenario #i:", where i is the number of the scenario starting at 1. Then print S lines containing the steps of the route in correct order. Terminate the output for the scenario with a blank line.

Example

Input:

```
2
4
SwimmingPool OldTree
BirdsNest Garage
Garage SwimmingPool
3
Toilet Hospital
VideoGame Toilet
```

Output:

```
Scenario #1:
BirdsNest
Garage
SwimmingPool
OldTree
```

```
Scenario #2:
VideoGame
Toilet
Hospital
```

Added by: Daniel Gómez Didier
Date: 2008-11-17
Time limit: 3s
Source limit:50000B
Languages: All
Resource: 2007 - Circuito de Maratones ACIS / REDIS

SPOJ Problem Set (classical)

3375. Stamps

Problem code: STAMPS

Everybody hates Raymond. He's the largest stamp collector on planet earth and because of that he always makes fun of all the others at the stamp collector parties. Fortunately everybody loves Lucy, and she has a plan. She secretly asks her friends whether they could lend her some stamps, so that she can embarrass Raymond by showing an even larger collection than his. Raymond is so sure about his superiority that he always tells how many stamps he'll show.And since Lucy knows how many she owns, she knows how many more she needs. She also knows how many friends would lend her some stamps and how many each would lend. But she's like to borrow from as few friends as possible and if she needs too many then she'd rather not do it at all. Can you tell her the minimum number of friends she needs to borrow from?

Input

The first line contains the number of scenarios. Each scenario describes one collectors party and its first line tells you how many stamps (from 1 to 1000000) Lucy needs to borrow and how many friends (from 1 to 1000) offer her some stamps. In a second line you'll get the number of stamps (from 1 to 10000) each of her friends id offering.

Output

The output for every scenario begins with a line containing "Scenario #i:", where i is the number of the scenario starting at 1. Then print a single line with the minimum number of friends Lucy needs to borrow stamps from. If it's impossible even if she borrows everything from everybody, write impossible. Terminate the output for the scenario with a blank line.

Example

Input:

```
3
100 6
13 17 42 9 23 57
99 6
13 17 42 9 23 57
1000 3
314 159 265
```

Output:

```
Scenario #1:
3

Scenario #2:
2

Scenario #3:
impossible
```

Added by: Daniel Gómez Didier
Date: 2008-11-17
Time limit: 3s
Source limit:50000B
Languages: All
Resource: 2007 - Circuito de Maratones ACIS / REDIS

SPOJ Problem Set (classical)

3376. Parking Lot

Problem code: PARKINGL

At a certain college, a small parking lot is arranged in a rectangular shape, with 20 spaces numbered 1, 2, 3..... 19, 20. Traffic flow is one way in a counter- clockwise direction. The lot looks something like this:

[IMAGE]

Note that the first position encountered upon entering is 1 and the last is 20. Cars may exit or continue to drive in a counter-clockwise direction. The following assumptions apply to this problem:
* At the start, class is in session and the lot is full (all 20 spaces are occupied by parked cars).
* In addition to the (20) cars already parked in the lot, K autos are in the lot waiting for positions to become available (1 <= K <= 20).
* Each waiting auto is positioned behind one of the occupied spaces. When a position empties, the space is filled either by the car waiting at that position or, if no car is waiting at that position, by the closest car, bearing in mind that the traffic flow is one way. (There is sufficient room at each position for the car parked in that position to leave and the car waiting at that position to then park.)
* When an auto advances N positions to a free spot, all other cars advance N positions. Since the lot is circular, advancing 4 positions from position 18 means advancing to position 2.
* None of the waiting cars exits.

Input

Write a program that reads data from standard input. Input consist of a line indicating the number of datasets, a blank line, and the datasets separated by a blank line. Each dataset is in two parts. The first part consists of integers, one per line beginning in column 1, representing initial positions of waiting autos. An integer 99 signals the end of this part of the data. The second part consists of integers, in the same format, representing positions vacated. Positions are vacated in the order in which their numbers appear in the second part of the data.

Output

The output of each dataset should consist a series of lines giving, for each initial (waiting) car position, the initial position and the final position of that car based on the description and assumptions stated above. The output lines must appear in the same order as the order of the initial positions given in the input. Print a blank line between datasets.

Example

Input:
```
1

6
19
17
13
```

```
1
99
1
3
20
16
```

Output:

```
Original position 6 parked in 16
Original position 19 parked in 3
Original position 17 did not park
Original position 13 parked in 20
Original position 1 parked in 1
```

Added by: Daniel Gómez Didier
Date: 2008-11-17
Time limit: 3s
Source limit: 50000B
Languages: All
Resource: 2007 PUJ - Circuito de Maratones ACIS / REDIS

SPOJ Problem Set (classical)

3377. A Bug's Life

Problem code: BUGLIFE

Professor Hopper is researching the sexual behavior of a rare species of bugs. He assumes that they feature two different genders and that they only interact with bugs of the opposite gender. In his experiment, individual bugs and their interactions were easy to identify, because numbers were printed on their backs.
Given a list of bug interactions, decide whether the experiment supports his assumption of two genders with no homosexual bugs or if it contains some bug interactions that falsify it.

Input

The first line of the input contains the number of scenarios. Each scenario starts with one line giving the number of bugs (at least one, and up to 2000) and the number of interactions (up to 1000000) separated by a single space. In the following lines, each interaction is given in the form of two distinct bug numbers separated by a single space. Bugs are numbered consecutively starting from one.

Output

The output for every scenario is a line containing "Scenario #i:", where i is the number of the scenario starting at 1, followed by one line saying either "No suspicious bugs found!" if the experiment is consistent with his assumption about the bugs' sexual behavior, or "Suspicious bugs found!" if Professor Hopper's assumption is definitely wrong.

Example

Input:

```
2
3 3
1 2
2 3
1 3
4 2
1 2
3 4
```

Output:

```
Scenario #1:
Suspicious bugs found!
Scenario #2:
No suspicious bugs found!
```

Added by: Daniel Gómez Didier
Date: 2008-11-17
Time limit: 1s-2s
Source limit: 50000B
Languages: All
Resource: 2007 PUJ - Circuito de Maratones ACIS / REDIS

SPOJ Problem Set (classical)

3378. Mirrored Pairs

Problem code: MIRRORED

The letters b and d are mirror images of each other, as are p and q. No other pairs of letters are mirrors, except for letters like H that are mirrors of themselves, and what's the interest in a pair that's just two of the same letter? We refuse to count self-mirrors as mirrored pairs.

Input

Input is a list of lines with two characters on each line. Your program should end immediately when it encounters a line with two spaces.

Output

The first line of output should contain only Ready. For each pair of characters (prior to a pair of spaces), print the line Mirrored pair if the characters are mirrors, otherwise print the line Ordinary pair.

Example

Input:

```
Fr
qp
HH
db

pq
```

Output:

```
Ready
Ordinary pair
Mirrored pair
Ordinary pair
Mirrored pair
```

Added by: Daniel Gómez Didier
Date: 2008-11-17
Time limit: 1s
Source limit: 50000B
Languages: All
Resource: 2007 PUJ - Circuito de Maratones ACIS / REDIS

SPOJ Problem Set (classical)

3379. String Shuffle

Problem code: SSHUFFLE

Given three strings consisting of just lowercase letters, count the number of ways that the third string can be constructed by combining two subsequences from the first two strings.
One derives a subsequence of the string by deleting zero or more characters from a string. For example, "", "a", "b", "c", "ab", "ac", "bc", and "abc" are all the subsequence strings of "abc". (Note that the empty string, "", is a subsequence of any string.)
The two subsequences are combined to make a third string by shuffling them together. That is, the relative order of the letters from the subsequence cannot be changed in the target string; but the two subsequences can be interleaved arbitrarily. For example, consider the two subsequences "abc" and "de". By combining them, one can get the following strings: "abcde", "abdce", "abdec", "adbce", "adbec", "adebc", "dabce", "dabec", "daebc", and "deabc".

Input

The first line of the input contains a single integer t that indicates the number of test cases. Each test case contains 3 strings, each containing only lowercase characters. The length of each string is between 1 and 60, inclusive.

Output

For each test case, output a line with a single integer that denotes the number of ways that one can construct the third string from the first two strings as described above.

Example

Input:

```
3
abc abc abc
aa aa aa
abbcd bccde abcde
```

Output:

```
8
10
18
```

Added by: Daniel Gómez Didier
Date: 2008-11-18
Time limit: 3s
Source limit:50000B
Languages: All
Resource: 2008 PUJ - Circuito de Maratones ACIS / REDIS

SPOJ Problem Set (classical)

3380. Tourist

Problem code: TOURIST

A lazy tourist wants to visit as many interesting locations in a city as possible without going one step further than necessary. Starting from his hotel, located in the north-west corner of city, he intends to take a walk to the south-east corner of the city and then walk back. When walking to the south-east corner, he will only walk east or south, and when walking back to the north-west corner, he will only walk north or west. After studying the city map he realizes that the task is not so simple because some areas are blocked. Therefore he has kindly asked you to write a program to solve his problem.
Given the city map (a 2D grid) where the interesting locations and blocked areas are marked, determine the maximum number of interesting locations he can visit. Locations visited twice are only counted once.

Input

The first line in the input contains the number of test cases (at most 20). Then follow the cases. Each case starts with a line containing two integers, W and H (2 <= W , H <= 100), the width and the height of the city map. Then follow H lines, each containing a string with W characters with the following meaning:
. Walkable area
* Interesting location (also walkable area)
# Blocked area
You may assume that the upper-left corner (start and end point) and lower-right corner (turning point) are walkable, and that a walkable path of length H + W - 2 exists between them.

Output

For each test case, output a line containing a single integer: the maximum number of interesting locations the lazy tourist can visit.

Example

Input:

```
2
9 7
*........
.....**#.
..**...#*
..####*#.
.*.#*.*#.
...#**...
*........
5 5
.*.*.
*###.
*.*.*
.###*
.*.*.
```

Output:

```
7
8
```

Added by:	Daniel Gómez Didier
Date:	2008-11-18
Time limit:	3s
Source limit:	50000B
Languages:	All
Resource:	2007 PUJ - Circuito de Maratones ACIS / REDIS

SPOJ Problem Set (classical)

3381. Highways

Problem code: HIGHWAYS

A number of cities are connected by a network of highways. Each highway is bidirectional and connects two cities, with a given travel time. What is the shortest time to get from a given city to another given city?

Input

The first line of input contains the number of test cases.
Each test case starts with a line containing the number of cities n (2 <= n <= 100000), the number of highways m (1 <= m <= 100000), the starting city and the ending city. Cities are numbered from 1 to n.
Then m lines follow, each describing one highway. The decription consists of the two distinct city numbers and the time in minutes to travel along the highway. The time will be between 1 and 1000.

Output

For each test case output a single line containing the minimum time it takes to get from the start to the destination. If no connection exists, output NONE.

Example

Input:

```
2
4 2 1 4
1 2 5
3 4 5
4 4 1 4
1 2 5
2 3 5
3 4 5
4 2 6
```

Output:

```
NONE
11
```

Added by: Daniel Gómez Didier
Date: 2008-11-18
Time limit: 3s
Source limit:50000B
Languages: All
Resource: Circuito de Maratones ACIS / REDIS

SPOJ Problem Set (classical)

3382. Monster Trap

Problem code: MONSTER

Once upon a time when people still believed in magic, there was a great wizard Aranyaka Gondlir. After twenty years of hard training in a deep forest, he had finally mastered ultimate magic, and decided to leave the forest for his home.
Arriving at his home village, Aranyaka was very surprised at the extraordinary desolation. A gloom had settled over the village. Even the whisper of the wind could scare villagers. It was a mere shadow of what it had been.
What had happened? Soon he recognized a sure sign of an evil monster that is immortal. Even the great wizard could not kill it, and so he resolved to seal it with magic. Aranyaka could cast a spell to create a monster trap: once he had drawn a line on the ground with his magic rod, the line would function as a barrier wall that any monster could not get over. Since he could only draw straight lines, he had to draw several lines to complete a monster trap, i.e., magic barrier walls enclosing the monster. If there was a gap between barrier walls, the monster could easily run away through the gap.
For instance, a complete monster trap without any gaps is built by the barrier walls in the left figure, where "M" indicates the position of the monster. In contrast, the barrier walls in the right figure have a loophole, even though it is almost complete.

[IMAGE]

Your mission is to write a program to tell whether or not the wizard has successfully sealed the monster.

Input

The input consists of multiple data sets, each in the following format.

[IMAGE]

The first line of a data set contains a positive integer n, which is the number of the line segments drawn by the wizard. Each of the following n input lines contains four integers x, y, x , and y , which represent the x- and y-coordinates of two points (x, y) and (x , y) connected by a line segment. You may assume that all line segments have non-zero lengths. You may also assume that n is less than or equal to 100 and that all coordinates are between -50 and 50, inclusive.
For your convenience, the coordinate system is arranged so that the monster is always on the origin (0, 0). The wizard never draws lines crossing (0, 0).
You may assume that any two line segments have at most one intersection point and that no three line segments share the same intersection point. You may also assume that the distance between any two intersection points is greater than 10^-5 .
An input line containing a zero indicates the end of the input.

Output

For each data set, print "yes" or "no" in a line. If a monster trap is completed, print "yes". Otherwise, i.e., if there is a loophole, print "no".

Example

Input:

```
8
-7 9 6 9
-5 5 6 5
-10 -5 10 -5
-6 9 -9 -6
6 9 9 -6
-1 -2 -3 10
1 -2 3 10
-2 -3 2 -3
8
-7 9 5 7
-5 5 6 5
-10 -5 10 -5
-6 9 -9 -6
6 9 9 -6
-1 -2 -3 10
1 -2 3 10
-2 -3 2 -3
0
```

Output:

```
yes
no
```

Added by: Daniel Gómez Didier
Date: 2008-11-18
Time limit: 3s
Source limit: 50000B
Languages: All
Resource: 2008 PUJ - Circuito de Maratones ACIS / REDIS

SPOJ Problem Set (classical)

3385. Yoda Goes Palindromic !

Problem code: YODA

According to a very famous web site, which in this case we will trust, defines a palindrome as ‘a word, phrase, verse, or sentence that reads the same backward or forward’. For example, the phrase A man, a plan, a canal, Panama! is a palindrome. Actually, writing texts consisting of only palindromes is part of a literary technique called constrained writing.
Now imagine the wise Yoda, the master of all, whose proficiency putting words together in sentences is one of his well-kwnown abilities. He is now interested in enriching his long- lasting, and maybe boring, inactivity periods by ‘composing’ palindromic sentences. That is, he has plans to use only palindromic sentences for his chats. For this matter, he needs to practice. The first task in his practice plan is to count all the palindromes that can be arranged out of a collection of characters.
Today, you will be Yoda’s assistant for this first task. Your only mission is to, given a sequence of characters, determine how many palindromes can be obtained with some of the characters in the sequence; you will only take into account uppercase or lowercase letters. Put in other way, you need to determine how many permutations of a give sequence of characters are palindromes. Your solution will help definitively master Yoga.

Input

The input consists of several test cases, one per line. For each test case, the input consist of a sequence of ASCII characters.

Output

For each test case you should print in a single line, and according to the order of the test cases, the total number of palindromes generated by the input sequence of ASCII characters. For your purpose, you should only consider uppercase or lowercase characters appearing in the input; any other character should be ignored in the calculations. Uppercase and lowercase characters are not considered different; for example, A and a should not be considered different. In any case, the total number of palindromes will not exceed the number e^43 , where e is approximately 2.71828. Remember that the empty sequence is a palindrome itself.

Example

Input:

```
A man, a plan, a canal, Panama!
arD,R!A
B.a.C1/
12[';. =1
```

Output:

```
15120
2
0
1
```

Added by: Daniel Gómez Didier
Date: 2008-11-18
Time limit: 4s
Source limit:50000B
Languages: All
Resource: 2007 U.Nacional - Circuito de Maratones ACIS / REDIS

SPOJ Problem Set (classical)

3386. Contest System Quality Assurance Tester

Problem code: QUALITY

Write a program to score a small, three-problem programming contest. Each input line contains six space-separated integers representing raw score data. The first three integers are in the range 0 . . . 100000. They represent seconds taken to solve the first, second, and third problems, respec- tively. Zero seconds indicates that a problem has not been solved. The last three integers are in the range 0 . . . 100, representing the attempts taken to solve the first, second, and third problems, respectively. Every failed attempt is penalized with 20 minutes, but only for problems that are eventually solved.
Each output line should begin with the string team, followed by a single space, the input line number, a colon, a single space, the number of solved problems, a comma, a single space, and the total number of seconds including penalties it took for the solved problems.

Input:
```
0 777 0 4 1 1
1 1 1 1 1 1
```

Output:
```
team 1: 1, 777
team 2: 3, 3
```

Added by: Daniel Gómez Didier
Date: 2008-11-18
Time limit: 2s
Source limit:50000B
Languages: All
Resource: 2007 U.Nacional - Circuito de Maratones ACIS / REDIS

SPOJ Problem Set (classical)

3387. Changing Maze

Problem code: CHMAZE

Luke Skywalker and his sister/love interest Leia are trying to get through a killer maze. And I mean killer! Every time step, the boundaries change. If our twins/lovebirds ever visit a square the same time a boundary appears, they're toast. There is no need to panic; the Force will guide them through the maze, and they will not die. However, the Force needs to know what advice to give and is therefore asking you for help.
Luke and Leia begin in the northwest corner of a maze. They want to make it to the southeast corner of the maze. At any given time step, Luke and Leia can move one square north, south, east, or west, or they can stay where they are. At every time step, the boundaries of the maze change: there is a finite list of patterns; if Luke and Leia are still in the maze when the list of patterns is exhausted, the maze cycles through again from the beginning of the list. You need to compute whether Luke and Leia can make it to the southeast corner of the maze, and, if so, the minimum number of time steps necessary for them to get there. Remember, the Force is counting on you! If you give the Force bad advice, we'll have to wait around for A Newer Hope and Force Knows how long that could take!

Input

The input consists of several test cases. Each case (but the last) will begin with a line containing three decimal integers. The first is the number of rows in the maze; the second is the number of columns in the maze; the third is the number of patterns in the list. The first two numbers will be inclusively between 1 and 20; the third will be inclusively between 1 and 10. The integers will be separated by exactly one space and will be followed by one . Immediately following this line will be a number of patterns, equal to the number specified on the first line. Each pattern will consist of r lineas each containing c characters, where r is the number of rows and c is the number of columns indicated on the first line. Each character will be either 0 (indicating no boundary) or 1 (indicating a boundary). Each line will be terminated by , and an extra will follow each pattern. The northwest corner of the first pattern will always be zero, since Luke and Leia will be starting from there. The last case will be three zeros, separated by exactly one space and followed by exactly one . This case is not to be processed; it indicates the end of input.

Output

The output cases are to appear in the same order in wich they appear in the input. Each output case should be of the form *Case c: Luke and Leia can escape in s steps.* or of the form *Case c: Luke and Leia cannot escape.* c and s are decimal integers. c in the number of the case being processed (starting with 1) and s is the minimum number of time steps Luke and Leia require to reach the southeast corner. Each line should be terminated by exactly one .

Example

Input:

```
5 5 1
00000
00000
00000
00000
00000
5 5 2
00000
00000
00000
00000
00000
01110
01110
11111
01110
01110
0 0 0
```

Output:

```
Case 1: Luke and Leia can escape in 8 steps.
Case 2: Luke and Leia cannot escape.
```

Added by: Daniel Gómez Didier
Date: 2008-11-18
Time limit: 3s
Source limit:50000B
Languages: All
Resource: 2007 U.Nacional - Circuito de Maratones ACIS / REDIS

SPOJ Problem Set (classical)

3388. Double Near Palindromes

Problem code: DNPALIN

C-3PO is an expert in pretty much every language. His conversation with R2-D2 are always fun to observe un that R2-D2 speaks in Droid, C-3PO speak in English, and they understand each other perfectly! Anyway, humans and droids both enjoy playing word games. A palindrome is a word or sequence of one or more letters that reads the same forwards and backwards. A near palindrome is a word or sequence that can be changed to or kept a palindrome by changing exactly one letter to a different letter. For example. BAT is a near palindrome, since changing the T to a B woluld make the word a palindrome: BAB. PEEP is not a near palindrome: although PEEP is palindrome, changing any letter would remove its palindrome status. A double near palindrome is a word or sequence that consist of two near palindromes concatenated together. For example, BATMAN is a double near palindrome, since BAT and MAN are both near palindromes. Given a list of words, you are to determine wich words are double near palindromes and wich are not.

Input

The input consists of one or more words. All words (except the last) will be inclusively between 1 and 25 letters long and will consist of entirely of capital letters. The last word will be *END* and is not be processed; it simply indicates the end of the input. There may be any number of spaces and characters before, after, and between words.

Output

The output cases are to appear in the same order in wich they appear in the input. For each input case, you are to print either w is a double near palindrome. or w is not a double near palindrome. wichever is appropriate, where w is the input word. Exactly one should follow each output case (meaning there should be no blank lines in the output).

Example

Input:

```
BATMAN
CONSTANTINOPLE
*END*
```

Output:

```
BATMAN is a double near palindrome.
CONSTANTINOPLE is not a double near palindrome.
```

Added by: Daniel Gómez Didier
Date: 2008-11-18
Time limit: 3s
Source limit:50000B
Languages: All
Resource: 2007 U.Nacional - Circuito de Maratones ACIS / REDIS

SPOJ Problem Set (classical)

3389. The Knights of the Round Circle

Problem code: KNIGHTSR

A group of Jedi Knights is having a competition. One of the Knights at random stands within a circle. The other Knights, in a random order, challenge him. If a challenger de- feats the Knight of the Round Circle, that Knight must leave the contest. The challenger then becomes the new Knight of the Round Circle and, as such, will face all subsequent challengers until he is defeated. If the current Knight of the Round Circle wins the chal- lenge, he stays within the circle and the challenger must leave the competition. The Knight within the circle at the end of the competition is deemed the winer. You may assume that no two Knights have exactly the same skill and that a stronger Knight will always defeat a weaker Knight.
Suppose there are three Knights in the competition. If the strongest one happens to stand in the circle first, he will not be defeated, so no one will ever leave the circle. If the weakest one happens to be first in the circle, he will be kicked out after his first match. If th e Knight that defeated him was the strongest, he will win the final challenge as well (so only one Knight will ever leave the circle),otherwise the strongest Knight will kick the middle Knight out of the circle during his challenge (so that two Knights leave the circle). If the middle Knight stands in the circle first, he will be the only one kicked out of the circle, no matter what order the other two come at him. All in all, an average of 5/6 or 0.83 Knight will leave the circle during the competition.
You are to compute the average number of Knights to leave the circle during a competition given the number of Knights in the competition.

Input

The input consists of several lines. Each line (but the last) will contain one positive decimal integer no larger than 10000. This integer is followed by exactly one . These integers represent the number of Knights in the competition. The last line will contain one zero, followed by . This line is not to be processed; it merely signifies the end of the input.

Output

The output cases are to appear in the same order in wich they appear in the input. For each case, you are to print With c competitors, a Jedi Knight will be replaced approximately t times. c is the number of competitors in this case and should be a decimal integer. t is the average number of times a Jedi Knight leaves the circle and should be a floating point decimal number with exactly two digits following the decimal point. There should always be at least one digit before the decimal point (use 0.50 rather .50, for example) The statement should be followed by two ’s, wich is to say that a blank line should follow every output case.

Example

Input:

```
3
1000
0
```

Output:

```
With 3 competitors, a Jedi Knight will be replaced approximately 0.83 times.

With 1000 competitors, a Jedi Knight will be replaced approximately 6.49 times.
```

Added by: Daniel Gómez Didier
Date: 2008-11-18
Time limit: 3s
Source limit: 50000B
Languages: All
Resource: 2007 U.Nacional - Circuito de Maratones ACIS / REDIS

SPOJ Problem Set (classical)

3390. Tribe Council

Problem code: TRIBE2

The Kazooba tribe contains N men, who want to gather for a tribe council. The men are all descendants of a single one, who is the chief of the tribe. Since the people of this tribe are quite long lived, every person's ancestors, including the chief, are alive and will be attending the council.

The men arrive at the council in some particular order and they are seated as they arrive. The men sit in a row of parallel tables. The first table is at the base of the Holy Mountain and the row extends indefinitely to the West (away from the Mountain). Unfortunately, each man doesn't get along with his father and his sons, so he doesn't want to sit at the same table with any of them. On the other hand, each man wants to be as close to divinity as possible, so, when he arrives at the council, he goes to the table closest to the Holy Mountain, where none of his children or father is already seated.

As usual, the Gods of the tribe have a certain inclination for grandeur. Therefore, they would like to command the men to arrive in such an order, as to maximize the number of tables that are occupied. Unfortunately, they do not have any inclination for programming, so they have ordered you to write a program for that.

Write a program that determines the maximum number of tables that can end up being occupied.

Input

The first line of the input file contains an integer N, representing the number of men in the tribe. All men are numbered from 1 to N; the chief is number 1. Each of the following lines describe a father-son relation and contains two integers A B separated by a blank, meaning that person number A is the father of the person number B. The relations described in the file are correct (for instance, there are no "cycles" and each person is a descendent of the chief).

Output

The output file should contain a single line with the maximum number of tables that can be occupied.

Example

Input:

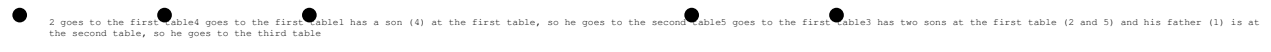
```
5
1 4
3 2
1 3
3 5
```

Output:

```
3
```

Explanation:

If men arrive in the order 2 4 1 5 3, the following happens:



Constraints

- 1 < N <= 100 000
- Any table has unlimited capacity.

Note: The problem statement and test data was updated on 22-11-2008. I'm sorry for the inconvenience.

Added by: Ajay Somani
Date: 2008-11-19
Time limit: 2s
Source limit:50000B
Languages: All
Resource: BOI 2003

SPOJ Problem Set (classical)

3393. Knot or Not

Problem code: NOTOKNOT

Disentangling of string loops is a classical 3D- puzzle. A designer of one of these puzzles wants to know if two given string loops can be disentangled or not.
The designer says that a string loop configuration that can be disentangled is a Notknot. In fact, he says that knots are those string loops that can not be disentangled.
Your task is to write a program to help the puzzle designer deciding if some configurations can or cannot be disentangled. In the problems that you are going to solve, loops are defined by straight segments in the space between points with integer coordinates. Then, a loop is described with a list of p points with integral coordinates for some p >= 3.

Input

Input consists of N test cases (1 <= N <= 1000). The number N is given in the first line of the input.
Each test case contains the description of two loops, each one in a line. A description for a p points loop (3 <= p <= 20) is given in one input line with 3p integer numbers separated by blanks:

x1 y1 z1 x2 y2 z2 . . . xp yp zp

what represents the loop:

(x1 , y1 , z1) - (x2 , y2 , z2) - . . . - (xp , yp , zp) - (x1 , y1 , z1)

where there is a straight segment between adjacent coordinates.

Output

For each analyzed case, one line classifying the case: Notknot or Knot.

Example

Input:

```
3
10 0 0 0 0 -10 0 10 0 0 0 10
5 0 0 -10 -10 0 0 5 0 10 10 0
10 0 0 0 0 -10 0 10 0 0 0 10
15 0 0 -10 -10 0 0 5 0 10 10 0
1 0 0 0 1 0 0 0 1
10 10 0 0 10 0 0 0 10 10 0 0 25 5 0 3 3 0
```

Output:

```
Notknot
Knot
Notknot
```

Added by: Daniel Gómez Didier
Date: 2008-11-19
Time limit: 4s
Source limit:50000B
Languages: All
Resource: 2008 U.Catolica & U.Central - Circuito de maratones ACIS / REDIS

SPOJ Problem Set (classical)

3394. Lagrange's Four-Square Theorem

Problem code: LAGRANGE

The fact that any positive integer has a representation as the sum of at most four positive squares (i.e. squares of positive integers) is known as Lagrange's Four-Square Theorem. The first published proof of the theorem was given by Joseph-Louis Lagrange in 1770. Your mission however is not to explain the original proof nor to discover a new proof but to show that the theorem holds for some specific numbers by counting how many such possible representations there are. For a given positive integer n, you should report the number of all representations of n as the sum of at most four positive squares. The order of addition does not matter, e.g. you should consider 4^2 + 3^2 and 3^2 + 4^2 are the same representation.
For example, let's check the case of 25. This integer has just three representations 1^2+2^2+2^2+4^2, 3^2 + 4^2, and 5^2. Thus you should report 3 in this case. Be careful not to count 4^2 + 3^2 and 3^2 + 4^2 separately.

Input

The input is composed of at most 255 lines, each containing a single positive integer less than 2^15 , followed by a line containing a single zero. The last line is not a part of the input data.

Output

The output should be composed of lines, each containing a single integer. No other characters should appear in the output. The output integer corresponding to the input integer n is the number of all representations of n as the sum of at most four positive squares.

Example

Input:
```
1
25
2003
211
20007
0
```

Output:
```
1
3
48
7
738
```

Added by: Daniel Gómez Didier
Date: 2008-11-19
Time limit: 2s
Source limit:50000B
Languages: All
Resource: 2008 U.Catolica & U.Central - Circuito de maratones ACIS / REDIS

SPOJ Problem Set (classical)

3405. Almost Shortest Path

Problem code: SAMER08A

Finding the shortest path that goes from a starting point to a destination point given a set of points and route lengths connecting them is an already well known problem, and it's even part of our daily lives, as shortest path programs are widely available nowadays.
>
> Most people usually like very much these applications as they make their lives easier. Well, maybe not that much easier.
>
> Now that almost everyone can have access to GPS navigation devices able to calculate shortest paths, most routes that form the shortest path are getting slower because of heavy traffic. As most people try to follow the same path, it's not worth it anymore to follow these directions.
>
> With this in his mind, your boss asks you to develop a new application that only he will have access to, thus saving him time whenever he has a meeting or any urgent event. He asks you that the program must answer not the shortest path, but the almost shortest path. He defines the almost shortest path as the shortest path that goes from a starting point to a destination point such that no route between two consecutive points belongs to any shortest path from the starting point to the destination.
>
> For example, suppose the figure below represents the map given, with circles representing location points, and lines representing direct, one-way routes with lengths indicated. The starting point is marked as S and the destination point is marked as D. The bold lines belong to a shortest path (in this case there are two shortest paths, each with total length 4). Thus, the almost shortest path would be the one indicated by dashed lines (total length 5), as no route between two consecutive points belongs to any shortest path. Notice that there could exist more than one possible answer, for instance if the route with length 3 had length 1. There could exist no possible answer as well.
>
>

subir imagenes

Input

The input contains several test cases. The first line of a test case contains two integers N ($2 <= N <= 500$) and M ($1 <= M <= 10^4$), separated by a single space, indicating respectively the number of points in the map and the number of existing one-way routes connecting two points directly. Each point is identified by an integer between 0 and N -1. The second line contains two integers S and D, separated by a single space, indicating respectively the starting and the destination points (S != D; $0 <= S, D < N$).
>
> Each one of the following M lines contains three integers U, V and P (U != V; $0 <= U, V < N$; $1 <= P <= 10^3$), separated by single spaces, indicating the existence of a one-way route from U to V with distance P. There is at most one route from a given point U to a given point V, but notice that the existence of a route from U to V does not imply there is a route from V to U, and, if such road exists, it

can have a different length. The end of input is indicated by a line containing only two zeros separated by a single space.

Output

For each test case in the input, your program must print a single line, containing `-1` if it is not possible to match the requirements, or an integer representing the length of the almost shortest path found.
>
>

Example

Input:

```
7 9
0 6
0 1 1
0 2 1
0 3 2
0 4 3
1 5 2
2 6 4
3 6 2
4 6 4
5 6 1
4 6
0 2
0 1 1
1 2 1
1 3 1
3 2 1
2 0 3
3 0 2
6 8
0 1
0 1 1
0 2 2
0 3 3
2 5 3
3 4 2
4 1 1
5 1 1
3 0 1
0 0
```

Output:

```
5
-1
6
```

Added by: Diego Satoba
Date: 2008-11-23
Time limit: 2s
Source limit: 50000B
Languages: C C++ JAVA
Resource: South American Regional Contests 2008

SPOJ Problem Set (classical)

3406. Bases

Problem code: SAMER08B

What do you get if you multiply 6 by 9? The answer, of course, is 42, but only if you do the calculations in base 13.

Given an integer $B >= 2$, the *base B numbering system* is a manner of writing integers using only digits between 0 and B -1, inclusive. In a number written in base B, the rightmost digit has its value multiplied by 1, the second rightmost digit has its value multiplied by B, the third rightmost digit has its value multiplied by B^2, and so on.

Some equations are true or false depending on the base they are considered in. The equation 2+2=4, for instance, is true for any $B >= 5$ - it does not hold in base 4, for instance, since there is no digit '4' in base 4. On the other hand, an equation like 2+2=5 is never true.

Write a program that given an equation determines for which bases it holds.

Input

Each line of the input contains a test case; each test case is an equation of the form "`EXPR=EXPR`", where both "`EXPR`" are arithmetic expressions with at most 17 characters.

All expressions are valid, and contain only the characters '`+`', '`*`' and the digits from '`0`' to '`9`'. No expressions contain leading plus signs, and no numbers in it have leading zeros.

The end of input is indicated by a line containing only "=".

Output

For each test case in the input your program should produce a single line in the output, indicating for which bases the given equation holds.

If the expression is true for infinitely many bases, print "`B+`", where B is the first base for which the equation holds.

If the expression is valid only for a finite set of bases, print them in ascending order, separated by single spaces.

If the expression is not true in any base, print the character '`*`'.

Example

`Input:`

```
6*9=42
10000+3*5*334=3*5000+10+0
2+2=3
2+2=4
```

```
0*0=0
=
```

Output:

```
13
6 10
*
5+
2+
```

Added by: Diego Satoba
Date: 2008-11-23
Time limit: 1s
Source limit: 50000B
Languages: C C++ JAVA
Resource: South American Regional Contests 2008

SPOJ Problem Set (classical)

3407. Candy

Problem code: SAMER08C

Little Charlie is a nice boy addicted to candies. He is even a subscriber to All Candies Magazine and was selected to participate in the International Candy Picking Contest.

In this contest a random number of boxes containing candies are disposed in *M* rows with *N* columns each (so, there are a total of *M* ×*N* boxes). Each box has a number indicating how many candies it contains.

The contestant can pick a box (any one) and get all the candies it contains. But there is a catch (there is always a catch): when choosing a box, all the boxes from the rows immediately above and immediately below are emptied, as well as the box to the left and the box to the right of the chosen box. The contestant continues to pick a box until there are no candies left.

The figure bellow illustrates this, step by step. Each cell represents one box and the number of candies it contains. At each step, the chosen box is circled and the shaded cells represent the boxes that will be emptied. After eight steps the game is over and Charlie picked 10+9+8+3+7+6+10+1 = 54 candies.
>
>

subir imagenes

>
> For small values of *M* and *N*, Charlie can easily find the maximum number of candies he can pick, but when the numbers are really large he gets completely lost. Can you help Charlie maximize the number of candies he can pick?

Input

The input contains several test cases. The first line of a test case contains two positive integers *M* and *N* (1 <= *M* ×*N* <= 10^5), separated by a single space, indicating the number of rows and columns respectively. Each of the following *M* lines contains *N* integers separated by single spaces, each representing the initial number of candies in the corresponding box. Each box will have initially at least 1 and at most 10^3 candies.

The end of input is indicated by a line containing two zeroes separated by a single space.

Output

For each test case in the input, your program must print a single line, containing a single value, the integer indicating the maximum number of candies that Charlie can pick.

Example

Input:

```
5 5
1 8 2 1 9
1 7 3 5 2
1 2 10 3 10
8 4 7 9 1
7 1 3 1 6
4 4
10 1 1 10
1 1 1 1
1 1 1 1
10 1 1 10
2 4
9 10 2 7
5 1 1 5
0 0
```

Output:

```
54
40
17
```

Added by: Diego Satoba
Date: 2008-11-23
Time limit: 2s
Source limit: 50000B
Languages: C C++ JAVA
Resource: South American Regional Contests 2008

SPOJ Problem Set (classical)

3408. DNA Sequences

Problem code: SAMER08D

Thomas, a computer scientist that works with DNA sequences, needs to compute longest common subsequences of given pairs of strings. Consider an alphabet S of letters and a word $w=a_1 a_2 \ldots a_r$, where a_i (- S, for i = 1, 2, ...,r. A *subsequence* of w is a word $x=a_{i_1} a_{i_2} \ldots a_{i_s}$ such that $1 <= i_1 < i_2 < \ldots < i_s <= r$. Subsequence x is a *segment* of w if $i_{j+1}=i_j + 1$, for j = 1,2, ...,s -1. For example the word `ove` is a segment of the word `lovely`, whereas the word `loly` is a subsequence of `lovely`, but not a segment.

A word is a *common subsequence* of two words w_1 and w_2 if it is a subsequence of each of the two words. A *longest common subsequence* of w_1 and w_2 is a common subsequence of w_1 and w_2 having the largest possible length. For example, consider the words w_1=`lovxxelyxxxxx` and w_2=`xxxxxxxlovely`. The words w_3=`lovely` and w_4=`xxxxxxx`, the latter of length 7, are both common subsequences of w_1 and w_2. In fact, w_4 is their longest common subsequence. Notice that the empty word, of length zero, is always a common subsequence, although not necessarily the longest.

In the case of Thomas, there is an extra requirement: the subsequence must be formed from common segments having length K or more. For example, if Thomas decides that K=3, then he considers `lovely` to be an acceptable common subsequence of `lovxxelyxxxxx` and `xxxxxxxlovely`, whereas `xxxxxxx`, which has length 7 and is also a common subsequence, is not acceptable. Can you help Thomas?

Input

The input contains several test cases. The first line of a test case contains an integer K representing the minimum length of common segments, where $1 <= K <= 100$. The next two lines contain each a string on lowercase letters from the regular alphabet of 26 letters. The length l of each string satisfies the inequality $1 <= l <= 10^3$. There are no spaces on any line in the input. The end of the input is indicated by a line containing a zero.

Output

For each test case in the input, your program must print a single line, containing the length of the longest subsequence formed by consecutive segments of length at least K from both strings. If no such common subsequence of length greater than zero exists, then `0` must be printed.

Example

Input:

```
3
lovxxelyxxxxx
xxxxxxxlovely
1
```

```
lovxxelyxxxxx
xxxxxxxlovely
3
lovxxxelxyxxxx
xxxlovelyxxxxxxx
4
lovxxxelyxxx
xxxxxxlovely
0
```

Output:

```
6
7
10
0
```

Added by: Diego Satoba
Date: 2008-11-23
Time limit: 10s
Source limit: 50000B
Languages: C C++ JAVA
Resource: South American Regional Contests 2008

SPOJ Problem Set (classical)

3409. Electricity

Problem code: SAMER08E

Martin and Isa stopped playing crazy games and finally got married. It's good news! They're pursuing a new life of happiness for both and, moreover, they're moving to a new house in a remote place, bought with most of their savings.

Life is different in this new place. In particular, electricity is very expensive, and they want to keep everything under control. That's why Martin proposed to keep a daily record of how much electricity has been consumed in the house. They have an electricity meter, which displays a number with the amount of KWh (kilowatt-hour) that has been consumed since their arrival.

At the beginning of each day they consult the electricity meter, and write down the consumption. Some days Martin does it, and some days Isa does. That way, they will be able to look at the differences of consumption between consecutive days and know how much has been consumed.

But some days they simply forget to do it, so, after a long time, their register is now incomplete. They have a list of dates and consumptions, but not all of the dates are consecutive. They want to take into account only the days for which the consumption can be precisely determined, and they need help.

Input

The input contains several test cases. The first line of each test case contains one integer N indicating the number of measures that have been taken ($2 <= N <= 10^3$). Each of the N following lines contains four integers D, M, Y and C, separated by single spaces, indicating respectively the day ($1 <= D <= 31$), month ($1 <= M <= 12$), year ($1900 <= Y <= 2100$), and consumption ($0 <= C <= 10^6$) read at the beginning of that day. These N lines are increasingly ordered by date, and may include leap years. The sequence of consumptions is strictly increasing (this is, no two different readings have the same number). You may assume that D, M and Y represent a valid date.

Remember that a year is a leap year if it is divisible by 4 and not by 100, or well, if the year is divisible by 400.

The end of input is indicated by a line containing only one zero.

Output

For each test case in the input, your program must print a single line containing two integers separated by a single space: the number of days for which a consumption can be precisely determined, and the sum of the consumptions for those days.

Example

Input:

```
5
9 9 1979 440
29 10 1979 458
30 10 1979 470
1 11 1979 480
2 11 1979 483
3
5 5 2000 6780
6 5 2001 7795
7 5 2002 8201
8
28 2 1978 112
1 3 1978 113
28 2 1980 220
1 3 1980 221
5 11 1980 500
14 11 2008 600
15 11 2008 790
16 12 2008 810
0
```

Output:

```
2 15
0 0
2 191
```

Added by: Diego Satoba
Date: 2008-11-23
Time limit: 1s
Source limit: 50000B
Languages: C C++ JAVA
Resource: South American Regional Contests 2008

SPOJ Problem Set (classical)

3410. Feynman

Problem code: SAMER08F

Richard Phillips Feynman was a well known American physicist and a recipient of the Nobel Prize in Physics. He worked in theoretical physics and also pioneered the field of quantum computing. He visited South America for ten months, giving lectures and enjoying life in the tropics. He is also known for his books "Surely You're Joking, Mr. Feynman!" and "What Do You Care What Other People Think?", which include some of his adventures below the equator.

His life-long addiction was solving and making puzzles, locks, and cyphers. Recently, an old farmer in South America, who was a host to the young physicist in 1949, found some papers and notes that is believed to have belonged to Feynman. Among notes about mesons and electromagnetism, there was a napkin where he wrote a simple puzzle: "how many different squares are there in a grid of $N \times N$ squares?".

In the same napkin there was a drawing which is reproduced below, showing that, for N=2, the answer is 5.
>
>

subir imagenes

Input

The input contains several test cases. Each test case is composed of a single line, containing only one integer N, representing the number of squares in each side of the grid (1 <= N <= 100).

The end of input is indicated by a line containing only one zero.

Output

For each test case in the input, your program must print a single line, containing the number of different squares for the corresponding input.

Example

Input:
```
2
1
8
0
```

Output:
```
5
1
204
```

Added by: Diego Satoba
Date: 2008-11-23
Time limit: 1s
Source limit: 50000B
Languages: C C++ JAVA
Resource: South American Regional Contests 2008

SPOJ Problem Set (classical)

3411. Pole Position

Problem code: SAMER08G

In car races, there is always a high pole next to the finish line of the track. Before the race starts, the pole is used to display the starting grid. The number of the first car in the grid is displayed at the top of the pole, the number of the car in second place is shown below that, and so on.

During the race, the pole is used to display the current position of each car: the car that is winning the race has its number displayed at the top of the pole, followed by the car that is in second place, and so on.

Besides showing the current position of a car, the pole is also used to display the number of positions the cars have won or lost, relative to the starting grid. This is done by showing, side by side to the car number, an integer number. A positive value *v* beside a car number in the pole means that car has won *v* positions relative to the starting grid. A negative value *v* means that car has lost *v* positions relative to the starting grid. A zero beside a car number in the pole means the car has neither won nor lost any positions relative to the starting grid (the car is in the same position it started).

>
>

subir imagenes

>
>

We are in the middle of the Swedish Grand Prix, the last race of the World Championship. The race director, Dr.Shoo Makra, is getting worried: there have been some complaints that the software that controls the pole position system is defective, showing information that does not reflect the true race order.

Dr.Shoo Makra devised a way to check whether the pole system is working properly. Given the information currently displayed in the pole, he wants to reconstruct the starting grid of the race. If it is possible to reconstruct a valid starting grid, he plans to check it against the real starting grid. On the other hand, if it is not possible to reconstruct a valid starting grid, the pole system is indeed defective.

Can you help Dr.Shoo Makra?

Input

The input contains several test cases. The first line of a test case contains one integer N indicating the number of cars in the race ($2 <= N <= 10^3$). Each of the next N lines contains two integers C and P, separated by one space, representing respectively a car number ($1 <= C <= 10^4$) and the number of positions that car has won or lost relative to the starting grid ($-10^6 <= P <= 10^6$), according to the pole system. All cars in a race have different numbers.

The end of input is indicated by a line containing only one zero.

Output

For each test case in the input, your program must print a single line, containing the reconstructed starting grid, with car numbers separated by single spaces. If it is not possible to reconstruct a valid starting grid, the line must contain only the value `-1`.

Example

Input:

```
4
1 0
3 1
2 -1
4 0
4
22 1
9 1
13 0
21 -2
3
19 1
9 -345
17 0
7
2 2
8 0
5 -2
7 1
1 1
9 1
3 -3
0
```

Output:

```
1 2 3 4
-1
-1
5 8 2 3 7 1 9
```

Added by: Diego Satoba
Date: 2008-11-23
Time limit: 1s
Source limit: 50000B
Languages: C C++ JAVA C#
Resource: South American Regional Contests 2008

SPOJ Problem Set (classical)

3412. Higgs Boson

Problem code: SAMER08H

It's been 100 years since the detection of the first Higgs boson and now particle physics is a mainstream subject in all high schools. Obviously, kids love the fact that they can create tiny black holes using only their portable particle accelerators and show off to their friends and colleagues. Although the creation of big black holes that could swallow the whole planet is possible even with these portable particle accelerators, the devices are programmed to only thrown particles when this undesirable side effect is impossible.

Your granddaughter is trying to create her own black holes with a portable accelerator kit, which is composed of two small particle accelerators that throw, each one, a boson-sized particle. Both particles are thrown at the same time, and a black hole appears when the particles collide. However, your granddaughter doesn't know how much time she'll have to wait before this happens. Fortunately, each accelerator can predict the particle's trajectory, showing four integer values into its display, called *A*, *B*, *C* and *D*. Each value can be replaced into the following equations:

$$r = At + B$$

$$\text{th} = Ct + D$$

in order to determine the trajectory of the particle, in polar coordinates. The radius (r) is represented in distance units and the angle (th) in degrees. The time (t) is given in time units and it is always a rational value which can be represented by an irreducible fraction. Your granddaughter knows that in polar coordinates a point has infinite representations. In general, the point (*r*, th) can be represented as (r, th +-$k \times 360^o$) or (-r, th +-$(2k + 1) \times 180^o$), where *k* is any integer. Besides, the origin ($r = 0$) can be represented as (0, th) for any th.

Using these parameters informed by each particle accelerator, your granddaughter wants to determine whether the particles will eventually collide and, if they do, the time when they will collide. After the first collision it is impossible to predict the particle's trajectory, therefore, only the first possible collision should be considered.

Although your granddaughter is really intelligent and has a deep knowledge of particle physics, she does not know how to program computers and is looking for some notes in her grandfather's (or grandmother's) ICPC notebook (don't forget, she is *your* granddaughter!). Fortunately for you, there is a note on your notebook which says that you wrote that code during the 2008 ICPC South America Regional Contest (or, to be more specific, *this* contest).

Input

The input consists of several test cases, one per line. Each test case contains eight integer numbers separated by single spaces, A_1, B_1,C_1, D_1, A_2, B_2, C_2, D_2 ($-10^4 <= A_1, B_1, C_1, D_1, A_2, B_2, C_2, D_2 <= 10^4$). The first four input values (A_1, B_1,C_1, D_1) correspond to the four parameters displayed

by the first portable particle accelerator and the following input values (A_2,B_2, C_2, D_2) correspond to the four parameters displayed by the second portable particle accelerator when both particles are thrown. The end of the input is represented by $A_1 = B_1 = C_1 = D_1 = A_2 = B_2 = C_2 = D_2 = 0$, which should not be processed as a test case, since these are the values displayed by the particle accelerators when a big black hole would be created if the particles were trown. Although the end of input is represented by a line with eight zeroes, note that the number zero is a possible input value.

Output

For each test case, your program must output a line containing two non-negative integers t_a and t_b separated by a single space. If there is no possibility of collision, $t_a = t_b = 0$, otherwise, t_a/t_b must be an irreducible fraction representing the earliest collision time. Even if the fraction results in an integer value, you still must output the number 1 as the denominator (see samples below).

Example

Input:

```
1 1 180 0 2 0 180 360
10 10 360 0 -24 18 180 72
5 5 180 0 -12 9 10 40
-9 5 5 180 2 5 5 180
0 0 0 0 0 0 0 0
```

Output:

```
1 1
0 0
4 17
0 1
```

Added by: Diego Satoba
Date: 2008-11-23
Time limit: 1s
Source limit:50000B
Languages: C C++ JAVA
Resource: South American Regional Contests 2008

SPOJ Problem Set (classical)

3413. Traveling Shoemaker Problem

Problem code: SAMER08I

Once upon a time there was a very peaceful country named Nlogonia. Back then, Poly the Shoemaker could come to the country and travel freely from city to city doing his job without any harassment. This task was very easy, as every city in Nlogonia had a direct road to every other city in the country. He could then easily travel the whole country visiting each city exactly once and fixing everybody's shoes.

But not anymore. The times have changed and war has come to Nlogonia. The age when people could travel freely is over.

Confederations identified by colors were formed among the cities all over the country, and now each city belongs to at least one and at most two confederations. When trying to enter a city, you must give to the border officer a ticket from one of the confederations this city belongs to. When leaving the city, you receive a ticket from the other confederation the city belongs to (i.e. different from the one you gave when entering) or from the same confederation if the city only belongs to one.

As Poly the Shoemaker is a long time friend of Nlogonia, he is allowed to choose a ticket and a city he wants to enter as the first city in the country, but after that he must obey the confederations rules. He wants to do the same routine he did before, visiting each city exactly once in Nlogonia, but now it's not easy for him to do this, even though he can choose where to start his journey.

For example, suppose there are four cities, labeled from 0 to 3. City 0 belongs to confederations *red* and *green*; city 1 belongs only to *red*; city 2 belongs to *green* and *yellow*; and city 3 belongs to *blue* and *red*. If Poly the Shoemaker chooses to start at city 0, he can enter it carrying either the *red* or the *green* ticket and leave receiving the other. Should he choose the *red* ticket, he will leave with a *green* ticket, and then there is only city 2 he can travel to. When leaving city 2 he receives the *yellow* ticket and now can't go anywhere else. If he had chosen the *green* ticket as the first he would receive the *red* one when leaving, and then he could travel to cities 1 or 3. If he chooses city 3, when leaving he will receive the *blue* ticket and again can't go anywhere else. If he chooses city 1, he receives the *red* ticket again when leaving (city 1 belongs only to the *red* confederation) and can only travel to city 3 and will never get to city 2. Thus, it is not possible to visit each city exactly once starting at city 0. It is possible, however, starting at city 2 with the *yellow* ticket, leaving the city with the *green* ticket, then visiting city 0, leaving with *red* ticket, then visiting city 1, leaving with *red* ticket again and, at last, visiting city 3.

As you can see, it got really difficult for Poly the Shoemaker to accomplish the task, so he asks you to help him. He wants to know if it's possible to choose a city to start such that he can travel all cities from Nlogonia exactly once.

Can you help Poly the Shoemaker?

Input

The input contains several test cases. The first line of a test case contains two integers N and C, separated by one space, indicating respectively the number of cities ($1 <= N <= 500$) and confederations ($1 <= C <= 100$) in the country. Each of the next C lines describes a confederation. It starts with one integer K ($0 <= K <= N$) and then K integers representing the cities which belong to this confederation. All integers are separated by single spaces and cities are numbered from 0 to N -1. Each city will appear at least once and at most twice and no city will be repeated on the same confederation.

The end of input is indicated by a line containing two zeroes separated by a single space.

Output

For each test case in the input, your program must print a single line, containing the integer `-1` if it's not possible to match the requirements or one integer representing the city where Poly the Shoemaker can start his journey. If there are multiple correct answers, print the smallest one.

Example

Input:

```
4 4
1 3
3 0 1 3
2 0 2
1 2
3 4
1 0
3 0 1 2
1 1
1 2
3 4
1 1
2 1 0
2 0 2
1 2
0 0
```

Output:

```
2
-1
1
```

Added by: Diego Satoba
Date: 2008-11-23
Time limit: 1s
Source limit:50000B
Languages: C C++ JAVA
Resource: South American Regional Contests 2008

SPOJ Problem Set (classical)

3414. Bora Bora

Problem code: SAMER08J

Bora Bora is a simple card game for children, invented in the South Pacific Island of the same name. Two or more players can play, using a deck of standard cards. Cards have the usual ranks: Ace, 2, 3, 4, 5, 6, 7, 8, 9, 10, Jack, Queen and King. Each card has also one of four suits: Clubs, Diamonds, Hearts and Spades.

Players sit on a circle around the table and play by turns. The next player to play may be the one to the left (clockwise direction) or to the right (counter-clockwise direction) of the current player, depending on the cards played, as we will see. At the start, the direction of play is clockwise.

The deck is shuffled and each player is dealt a hand of cards. The remaining of the deck is placed, face down, on the table; this is called the *stock* pile. Then the first (topmost) card is removed from the stock and placed on the table, face up, starting another pile, called the *discard* pile.

The objective of the game is for a player to discard all his cards. At each turn, a player discards at most one card. A card can be discarded only if it has the same rank or the same suit as the topmost card on the discard pile. A player discards a card by placing it, face up, in the discard pile (this card becomes the topmost). If a player does not have a suitable card to discard on his turn, he must draw one card from the stock and add it to his hand; if he can discard that card, he does so, otherwise he does nothing else and his turn ends. A player always discards the highest valued card he possibly can. The *value* of a card is determined first by the card rank and then by the card suit. The rank order is the rank itself (Ace is the lowest, King is the highest), and the suit order is, from lowest to highest, Clubs, Diamonds, Hearts and Spades. Therefore, the highest valued card is the King of Spades and the lowest valued card is the Ace of Clubs. As an example, a Queen of Diamonds has a higher value than a Jack (any suit) but has a lower value than a Queen of Hearts.

Some of the discarded cards affect the play, as follows:

- when a Queen is discarded, the direction of play is reversed: if the direction is clockwise, it changes to counter-clockwise, and vice-versa;
- when a Seven is discarded, the next player to play must draw two cards from the stock (the number of cards in his hand increases by two), and misses his turn (does not discard any card);
- when an Ace is discarded, the next player to play must draw one card from the stock (the number of cards in his hand increases by one), and misses his turn (does not discard any card);
- when a Jack is discarded, the next player to play misses his turn (does not discard any card).

Notice that the penalty for the first card in the discard pile (the card draw from the stock at the beginning) is applied to the first player to play. For example, if the first player to play is *p* and the first card on the discard pile is an Ace, player *p* draws a card from the stock and does not discard any card on his first turn. Also notice that if the first card is a Queen, the direction of play is reversed to counter-clockwise, but the first player to play remains the same.

The winner is the player who first discards all his cards (the game ends after the winner discards his last card).

Given the description of the shuffled deck and the number of players, write a program to determine who will win the game.

Input

The input contains several test cases. The first line of a test case contains three integers *P*, *M* and *N*, separated by single spaces, indicating respectively the number of players ($2 <= P <= 10$), the number of cards distributed to each of the players at the beginning of the game ($1 <= M <= 11$) and the total number of cards in the shuffled deck ($3 <= N <= 300$). Each of the next *N* lines contains the description of one card. A card is described by one integer *X* and one character *S*, separated by one space, representing respectively the card rank and the card suite. Card ranks are mapped to integers from 1 to 13 (Ace is 1, Jack is 11, Queen is 12 and King is 13). Card suits are designated by the suit's first letter: '`C`' (Clubs), '`D`' (Diamonds), '`H`' (Hearts) or '`S`' (Spades).

Players are identified by numbers from 1 to *P*, and sit on a circle, in clockwise direction, 1, 2 ...*P*, 1. The first $P \times M$ cards of the deck are dealt to the players: the first *M* cards to the first player (player 1), the next *M* to the second player (player 2), and so on. After dealing the cards to the players, the next card on the deck - the $(P \times M + 1)$-th card - is used to start the discard pile, and the remaining cards form the stock. The $(P \times M + 2)$-th card to appear on the input is the topmost card on the stock, and the last card to appear on the input (the *N*-th card) is the bottommost card of the stock (the last card that can be drawn). Player 1 is always the first to play (even when the card used to start the discard pile is a Queen). All test cases have one winner, and in all test cases the number of cards in the deck is sufficient for playing to the end of the game.

The end of input is indicated by a line containing only three zeros, separated by single spaces.

Output

For each test case in the input, your program must print a single line, containing the number of the player who wins the game.

Example

Input:

```
2 2 10
1 D
7 D
1 S
3 C
13 D
1 S
5 H
12 D
7 S
2 C
3 2 11
1 S
7 D
11 D
3 D
7 D
```

```
3 S
11 C
8 C
9 H
6 H
9 S
3 3 16
1 H
10 C
13 D
7 C
10 H
2 S
2 C
10 S
8 S
12 H
11 C
1 C
1 C
4 S
5 D
6 S
0 0 0
```

Output:

```
1
3
2
```

Added by: Diego Satoba
Date: 2008-11-23
Time limit: 1s
Source limit: 50000B
Languages: C C++ JAVA
Resource: South American Regional Contests 2008

SPOJ Problem Set (classical)

3415. Shrinking Polygons

Problem code: SAMER08K

A polygon is said to be *inscribed* in a circle when all its vertices lie on that circle. In this problem you will be given a polygon inscribed in a circle, and you must determine the minimum number of vertices that should be removed to transform the given polygon into a *regular polygon*, i.e., a polygon that is equiangular (all angles are congruent) and equilateral (all edges have the same length).

When you remove a vertex v from a polygon you first remove the vertex and the edges connecting it to its adjacent vertices w_1 and w_2, and then create a new edge connecting w_1 and w_2. Figure (a) below illustrates a polygon inscribed in a circle, with ten vertices, and figure (b) shows a pentagon (regular polygon with five edges) formed by removing five vertices from the polygon in (a).
>
>

subir imagenes

>
>

In this problem, we consider that any polygon must have at least three edges.

Input

The input contains several test cases. The first line of a test case contains one integer N indicating the number of vertices of the inscribed polygon ($3 <= N <= 10^4$). The second line contains N integers X_i separated by single spaces ($1 <= X_i <= 10^3$, for $0 <= i <= N$ -1). Each X_i represents the length of the arc defined in the inscribing circle, clockwise, by vertex i and vertex (i+1) mod N. Remember that an *arc* is a segment of the circumference of a circle; do not mistake it for a *chord*, which is a line segment whose endpoints both lie on a circle.

The end of input is indicated by a line containing only one zero.

Output

For each test case in the input, your program must print a single line, containing the minimum number of vertices that must be removed from the given polygon to form a regular polygon. If it is not possible to form a regular polygon, the line must contain only the value `-1`.

Example

Input:

```
3
1000 1000 1000
6
1 2 3 1 2 3
```

```
3
1 1 2
10
10 40 20 30 30 10 10 50 24 26
0
```

Output:

```
0
2
-1
5
```

Added by: Diego Satoba
Date: 2008-11-23
Time limit: 1s
Source limit: 50000B
Languages: C C++ JAVA
Resource: South American Regional Contests 2008

SPOJ Problem Set (classical)

3420. Falling Ice

Problem code: FALLINGI

Imagine disks of ice falling, one at a time, into a box, each ending up at the lowest point it can reach without overlapping or moving previous disks. Each disk then freezes into place, so it cannot be moved by later disks. Your job is to find the overall height of the final combination of disks.

So that the answer is unique, assume that any disk reaching the bottom of the box rolls as far to the left as possible. Also the data is chosen so there will be a unique lowest position for any disk that does not reach the bottom. The data is also such that there are no "perfect fits": each disk that lands will be in contact with only two other points, on previous circles or the sides of the box. The illustrations above show white filled disks labeled with the order in which they fall into their boxes. The gray circle in the fourth illustration is not intended to be a disk that fell in. The gray disk is included to demonstrate a point: the gray disk is the same size as disk 2, so there is space for disk 2 on the very bottom of its box, but disk 2 cannot reach that position by falling from the top. It gets caught on disk 1 and the side of the box.

One way to find the top intersection point of two intersecting circles is as follows. Suppose circle 1 has center (x1, y1) and radius r1, and suppose circle 2 has center (x2, y2), and radius r2. Also assume that circle 1 is to the left of circle 2, i.e., x1 < x2. Let

dx = x2 - x1,
dy = y2 - y1,
D = sqrt(dx*dx + dy*dy),
E = (r1*r1 - r2*r2 + D*D)/(2*D),
F = sqrt(r1*r1 - E*E);

then the upper intersection point is (x1 + (E*dx - F*dy)/D, y1 + (F*dx + E*dy)/D).

Input

The input consists of one or more data sets, followed by a line containing only 0 that signals the end of the input. Each data set is on a line by itself and contains a sequence of three or more blank-separated positive integers, in the format w, n, d_1, d_2, d_3, ..., d_n, where w is the width of the box, n is the number of disks, and the remaining numbers are the diameters of the disks, in the order in which they fall into the box. You can assume that w < 100, that n < 10, and that each diameter is less than w.

Output

For each data set, output a single line containing the height of the pile of disks, rounded to two places beyond the decimal point.

The example data matches the illustrations above.

Example

Input:

```
10 3 5 2 3
8 2 5 5
11 3 10 2 4
9 3 4 4 6
10 6 5 4 6 3 5 2
0
```

Output:

```
5.00
9.00
12.99
9.58
14.19
```

Added by: Nikola P Borisov
Date: 2008-11-24
Time limit: 60s
Source limit: 50000B
Languages: All
Resource: Mid-Central Regional ACM-ICPC Contest 2006

SPOJ Problem Set (classical)

3426. Ouroboros Snake

Problem code: OROSNAKE

Ouroboros is a mythical snake from ancient Egypt. It has its tail in its mouth and continously devours itself.
The Ouroboros numbers are binary numbers of 2^n bits that have the property of "generating" the whole set of numbers from 0 to 2^n - 1. The generation works as follows: given an Ouroboros number, we place its 2^n bits wrapped in a circle. Then, we can take 2^n groups of n bits starting each time with the next bit in the circle. Such circles are called Ouroboros circles for the number n. We will work only with the smallest Ouroboros number for each n.
Example: for n = 2, there are only four Ouroboros numbers. These are 0011;0110;1100; and 1001. In this case, the smallest one is 0011. Here is the Ouroboros circle for 0011:

[IMAGE]

The table describes the function o(n;k) which calculates the k-th number in the Ouroboros circle of the smallest Ouroboros number of size n. This function is what your program should compute.

Input

The input consists of several test cases. For each test case, there will be a line containing two integers n and k (1<=n<=15; 0<=k<2^n). The end of the input file is indicated by a line containing two zeros. Don’t process that line.

Output

For each test case, output o(n;k) on a line by itself.

Example

Input:
```
2 0
2 1
2 2
2 3
0 0
```

Output:
```
0
1
3
2
```

Added by: Daniel Gómez Didier
Date: 2008-11-24
Time limit: 3s
Source limit: 50000B
Languages: All
Resource: PUJ Internal Contest

SPOJ Problem Set (main)

3436. Histogram

Problem code: HIST2

In statistics, a histogram is a graphical display of tabulated frequencies, shown as bars. It shows what proportion of cases fall into each of several categories. It is a polygon composed of a sequence of rectangles aligned at a common base line. In this problem all rectangles have a width of unit length. But their heights are distinct. Some permutation of the heights will give the maximum perimeter. Your task is to find the maximum perimeter of the histogram and the number of permutations that give the maximum perimeter.

[IMAGE]

In the image Figure (a) shows a histogram with heights {1,2,3,4} (1st sample testcase) and has a perimeter of 16 units. Figure (b) shows one of the permutations {3,1,2,4} having the maximum perimeter of 20 units.

Input

Input consists of multiple test cases. Each test case describes a histogram and starts with an integer **N**, 2 <= N <= 15, denoting the number of rectangles it is composed of. Next line consists of N space separated positive integers representing the heights of the rectangles. All heights are distinct and less than or equal to 100. N=0 indicates the end of tests. There are atmost 50 test cases.

Output

For each test case output the maximum possible perimeter of the histogram and the number of permutations that give maximum perimeter in a single line, separated by a single space.

Example

Input:

```
4
1 2 3 4
3
2 6 5
0
```

Output:

```
20 8
24 2
```

Added by:	u.swarnaprakash
Date:	2008-11-29
Time limit:	4s
Source limit:	50000B
Languages:	All
Resource:	Kurukshetra 09 OPC

SPOJ Problem Set (classical)

3442. The last digit

Problem code: LASTDIG

Nestor was doing the work of his math class about three days but he is tired of make operations a lot and he should deliver his task tomorrow. His math's teacher gives two numbers a and b. The problem consist in find the last digit of the potency of base a and index b. Help Nestor with his problem. You are given two integer numbers: the base a (0 < a < 20) and the index b (0 <= b <= 2,147,483,000). You have to find the last digit of a^b.

Input

The first line of input contains an integer t, the number of test cases (t <= 30). t test cases follow. For each test case will appear a and b separated by space.

Output

For each test case output an integer per line representing the result.

Example

Input:
```
2
3 10
6 2
```

Output:
```
9
6
```

Added by: Jose Daniel Rdguez
Date: 2008-12-01
Time limit: 1s
Source limit: 700B
Languages: All
Resource: Mine

SPOJ Problem Set (classical)

3459. SkyScrapers

Problem code: CEPC08B

In a seaside village, there is an avenue of skyscrapers. Each skyscrapers is 100m wide and has certain height. Due to very high price of parcels, any two consecutive skyscrapers are adjacent. The avenue lies close to the beach so the street is exactly at the sea level. Unfortunately, this year, due to the global warming, the sea level started to increase by one meter each day. If the skyscraper height is no greater than the current sea level, it is considered flooded. A region is a maximal set of non-flooded, adjacent skyscrapers. This term is of particular importance, as it is sufficient to deliver goods (like current, carrots or cabbages) to any single skyscraper in each region. Hence, the city major wants to know how many regions there will be in the hard days that come. An example of an avenue with 5 skyscrapers after 2 days is given below.

[IMAGE]

Input

The input contains several test cases. The first line contains an integer t (t <= 15) denoting the number of test cases. Then t test cases follow. Each of them begins with a line containing two numbers n and d (1 <= n, d <= 10^6), n is the number of skyscrapers and d is the number of days which the major wants
to query. Skyscrapers are numbered from left to right. The next line contains n integers $h_1, h_2, \ldots, h_n$
where $1 <= h_i <= 10^9$ is the height of skyscraper i. The third line of a single test case contains d numbers
t_j such that $0 <= t_1 < t_2 < \ldots < t_{d-1} < t_d <= 10^9$.

Output

For each test case output d numbers $r_1, r_2, \ldots, r_d$, where r_j is the number of regions on day t_j .

Example

Input:

```
2
3 3
1 2 3
1 2 3
5 3
1 3 5 1 3
0 2 4
```

Output:

```
1 1 0
1 2 1
```

Added by:	Robert Rychcicki
Date:	2008-12-06
Time limit:	10s
Source limit:	50000B
Languages:	All
Resource:	Central European Regional Contest 2008

SPOJ Problem Set (classical)

3462. The Skatepark´s New Ramps

Problem code: RAMP

The local skating park has been given a financial incentive by the city to make the park interesting for skaters of all levels. The park wants to use the incentive to build a series of ramps, somewhat resembling a mountain range. When talking to some of the volunteers in the committee responsible for the project, you find out they're having dificulties deciding about the best conguration of the ramps. They know the number of ramps to be built, and for each ramp they agree on the range of the height for that ramp. They are still discussing exactly how high each ramp should be, since they can't afford to have them all at their highest, but they do want to spend all of the budget. This is the most important issue in the debate: they can't agree whether they want the differences between the ramps to be small, to give the full ride a more consistent feeling, or as big as possible, to create a more diverse set of challenges.

You also notice they don't really have a good idea what the possibilities are, leaving them stranded in 'what-if' discussions. You decide to help them out by showing them the options they have, both the ones where the dierence between the highest and lowest ramp is kept as small as possible, as well as the one where that difference is as much as possible. Since the committee is mainly bickering over the allowable differences, you decide to start out by just presenting them the minimum and maximum difference between the highest and lowest ramp. Luckily, the park has a lot of space, so you won't need to take the placement of the ramps into account. All ramps have the same inclination, which is such that a ramp of height h will have a length $4h$ (measuredflat, not over the ramp).

Input

The first line of input consists of the integer number n, the number of test cases. For each test case:

A line with the integer number r (2 <= r <= 10000), the number of ramps the park will place;

A line with the integer number m (0 <= m <= 200000000), the number of cubic meters of concrete the park has money for;

r lines with two numbers, l and t (0.00 <= l <= t <= 100.00), separated by one space, the minimum and maximum height in meters of the r-th ramp.

You may assume all ramps are made entirely of concrete, and shaped as 1 meter wide prisms, with a triangle with two equal sides as base. A series of ramps within the given constraints and using all concrete is guaranteed to exist.

Output

For each test case, the output contains one line with two numbers, separated by one space: the minimum difference between the highest and lowest ramp and the maximum difference between the highest and the lowest ramp. These numbers are rounded to two decimals.

Example

Input:

```
1
3
36
1.00 4.00
1.00 4.00
1.00 4.00
```

Output:

```
0.00 3.00
```

Added by: Blue Mary
Date: 2008-12-07
Time limit: 1s
Source limit: 50000B
Languages: All except: C99 strict
Resource: Benelux Algorithm Programming Contest 2008

SPOJ Problem Set (classical)

3465. Drive through MegaCity

Problem code: DRIVE

MegaCity of the future is a rectangular grid of streets. Each intersection has integer Cartesian coordinates x and y. To get from intersection a with coordinates x_a, y_a to intersection b with coordinates x_b, y_b you need to drive exactly $|x_a - x_b| + |y_a - y_b|$ blocks. Usually, it takes 10 time units to drive one block, so one can easily compute the time it takes to get from a to b. However, traffic jams that occur in MegaCity turn estimation of minimal driving time into a complex problem that you have to solve.

Traffic jams in MegaCity affect a rectangular area that is specified by coordinates of its bottom-left and top-right corners. The time to travel one block in the traffic jam is specified. All streets that are strictly inside the rectangular region are affected by the traffic jam. Sometimes, it is better to drive around the traffic jams to save time, but sometimes it is better to drive through some traffic jams as shown in the example - 17 blocks are driven outside of traffic jams, taking 10 time units per block, and 2 blocks in the light traffic jam with 11 time units per block.

[IMAGE]

Input

Multiple test cases. The number of them is given in the very first line. For each test case:

The first line contains four integer numbers x_a, y_a, x_b and y_b, coordinates of the start and finish intersections. The second line contains a single integer number n ($0 <= n <= 1000$) which specifies the number of traffic jams. The following n lines describe traffic jams. Each traffic jam is described by five integer numbers $x_{1,i}$, $y_{1,i}$, $x_{2,i}$, $y_{2,i}$ and t_i, where first four numbers are coordinates of the bottom-left and top-right corners of the jammed area ($x_{1,i} < x_{2,i}$, $y_{1,i} < y_{2,i}$), and t_i ($10 < t_i <= 10^8$) is the time it takes to travel one block inside this traffic jam. All coordinates are from 0 to 10^8 inclusive. Areas of traffic jams neither intersect nor touch each other. Start and finish points are different and do not lie inside nor on the border of any traffic jam.

Output

For each test case:

A single integer - the minimal driving time from intersection a to intersection b.

Example

Input:

```
1
1 6 15 3
4
2 1 3 7 44
5 2 10 4 33
```

```
8 5 11 9 22
12 1 14 8 11
```

Output:

```
192
```

Warning: A naive algorithm may not run in time!

Note: In Sphere Online Judge system, "Memory Limit Exceeded" will be shown as "Runtime Error(other)", with the 0.00 second run-time & 92-200k memory used, or "Runtime Error(SIGSEGV)" with 250M memory used.

Added by: Blue Mary
Date: 2008-12-07
Time limit: 60s
Source limit: 50000B
Languages: All except: C99 strict
Resource: ACM Northeastern European Regional Contest 2008

SPOJ Problem Set (classical)

3476. Deposit

Problem code: DEPOSIT

Banks offer deposit schemes of various kinds to attract customers. In an r -year progressively reducing recurring deposit scheme (PRRDS) of a bank, a customer is required to deposit progressively reducing amounts every year for r years. Depending on the duration r of the scheme and the total amount T deposited in r years, the bank offers to return on maturity, i.e., after the expiry of r years, an amount R , which is equal to k times the amount deposited in the first year. The bank ensures that the return R looks attractive by making a suitable choice of k ; k being a natural number.

In a PRRDS, the amount to be deposited in each but the last two years is exactly equal to the sum of amounts to be deposited in the next two years. The amounts to be deposited in the last two years, say x in the last year and y in the last but one year, are progressively reducing (x, y > 0;y > x) and are determined so that the total amount deposited in r years is exactly equal to the specified amount T . Assume that all deposits are in whole number of Rupees.

Write a program for the bank, so that given r , k and T , the program computes x and y for which the return R is maximum. For example in a 4-year scheme with r = 4 , k = 3 and T = 500 , the progressively reducing recurring deposits 248, 126, 122 and 4, ensures the maximum return R = 744 with x = 4 and y = 122 .

Input

The input may contain multiple test cases.

For each test case there is only one input line. The line gives values of r , k and T . Assume that r is not greater than 20.

A line containing a zero ‘0’ as the first character follows the last case.

Output

For each test case there is only one output line. The line gives the computed values of x , y and R .

Example

Input

```
4 3 500
5 3 10000
6 4 8000
8 5 12000
0
```

Output

```
4 122 744
5 1425 12855
1 666 13332
1 363 23635
```

Added by: Walrus
Date: 2008-12-09
Time limit: 1s
Source limit: 50000B
Languages: All
Resource: ACM Kanpur 2007

SPOJ Problem Set (classical)

3477. Baby

Problem code: BABY

A baby tries to solve the well-known eight-queen puzzle: the problem of placing eight chess queens on an 8×8 chessboard so that no two queens share the same row, column, or diagonal. The baby understands the concept of row and column quite well but diagonal is not very clear to her. As a result she succeeds placing eight queens on the board so that no two queens share the same row or column but there remains the possibility that some queens share the same diagonal.

Given baby's queens (a solution by the baby) and a valid eight-queen solution, it is possible to move baby's queens to positions of queens in the valid solution. Assume that in a single move, a queen can be moved one unit row-wise or column-wise into an unoccupied position.

Write a program to find the minimum number of moves required to move baby's queens to positions of queens in the valid solution. The program should be usable for a more general n -queen puzzle where n queens are placed on an n×n chessboard, 4 <= n <= 16 . Assume that rows and columns of the chessboard are numbered 1, 2,..., n .

Input

The input consists of multiple test cases.

Each case begins with a line containing the integer n .

Each of the next two lines contains a sequence of n integers. Integers in the first line represent column numbers of baby's queens appearing in rows 1, 2,..., n respectively. In the same way, the second line contains column numbers of queens in the given valid solution. A space separates two consecutive integers in the sequence.

A line containing a zero '0' as the first character follows the last case.

Output

For each test case, print the minimum number of moves required.

Example

Sample Input

```
4
1 2 3 4
3 1 4 2
4
3 2 4 1
3 1 4 2
5
5 3 1 4 2
5 3 1 4 2
5
```

```
1 5 2 4 3
3 1 4 2 5
0
```

Sample Output

```
6
2
0
8
```

Added by: Walrus
Date: 2008-12-09
Time limit: 1s
Source limit: 50000B
Languages: All
Resource: ACM Kanpur 2007

SPOJ Problem Set (classical)

3483. Begin

Problem code: BEGIN

Begin a sequence of distinct natural numbers Ni , i = 0, 1, 2,... , with the number B (= N0) ; generate numbers Ni , i = 1, 2,... , recursively and end the sequence with the last generated number E . The characteristic of numbers and the process for generation are stated below:

* Each number in the sequence contains an even number of decimal digits and is of the form f1d1f2d2fk...dk where d1, d2,..., dk , are k distinct digits in increasing order and each fj is a non-zero digit.

* For i = 0, 1, 2,... , if Ni = f1d1f2d2...fkdk then Ni+1 = F1D1F2D2...FKDK , where K$ \ge$k ; D1, D2,..., DK , are distinct digits that occur in Ni and appear in increasing order in Ni+1 ; and FJ is the frequency of DJ in Ni , for J = 1, 2,..., K . For example if Ni = 102335 then Ni+1 = 1011122315 .

Write a program to find for a given E , the longest sequence of numbers that ends with E and begins with the smallest B .

Again consider an example; if E =1011122315 then the required sequence of numbers is 303355 103325 1011122315.

Input

The input may contain multiple test cases.

Each test case contains only one input, viz., E .

The input terminates when a line containing 0 appears as a test case.

Output

For each test case, print the longest sequence of numbers that ends with E and begins with the smallest B . Use space to separate two consecutive numbers in the sequence.

Example

Sample Input

```
1011122315
22
112213
0
```

Sample Output

```
303355 103325 1011122315
22
13 1113 3113 2123 112213
```

Added by: Walrus
Date: 2008-12-09
Time limit: 1s-10s
Source limit: 50000B
Languages: All
Resource: ACM Kanpur 2006

SPOJ Problem Set (classical)

3484. Crossbits

Problem code: CROSSBIT

Crossbits are like Crosswords; instead of entering words you enter binary bits 01 in a Crossbit under certain given conditions, assuming that a solution exists. An empty Crossbit of size N is an empty grid of size N×N.

Given a natural number N , consider entering N^2 binary bits in an empty Crossbit, satisfying the following conditions:

- Each square in the grid contains either a 0-bit or a 1-bit with no 1-bit in two major diagonals.
- The total number of 1-bit in each row / column is exactly equal to K , K being a given natural number less than N.
- A 0-bit has at least another adjacent 0-bit either in the same row or in the same column.
- The Crossbit represents the N2 -bit binary number B formed by placing bits in the 1st , the 2nd , ... the Nth row from left to right.

You are required to write a program that enters bits in an empty Crossbit so that the Crossbit represents the least binary number B for given N and K .

As an illustration consider the case with N = 4 and K = 1 . The Crossbit shown below represents the least binary number B = 0010100000010100 of 16 bits satisfying the specified conditions.

```
0       0       1       0
1       0       0       0
0       0       0       1
0       1       0       0
```

Input

The input may contain multiple test cases.

For each test case parameters N and K of the Crossbit are given in one line. Assume that N does not exceed 10.

The input terminates with a line containing 0 as input.

Output

For each test case, print the Crossbit in N rows; each row contains N bits with a space between two neighbouring bits. Keep a blank line after the last output line of each test case.

Example

Sample Input

```
4 1
6 2
0
```

Sample Output

```
0 0 1 0
1 0 0 0
0 0 0 1
0 1 0 0

0 0 0 1 1 0
1 0 0 1 0 0
0 0 0 0 1 1
1 1 0 0 0 0
0 0 1 0 0 1
0 1 1 0 0 0
```

Added by: Walrus
Date: 2008-12-09
Time limit: 1s
Source limit: 50000B
Languages: All
Resource: ACM Kanpur 2006

SPOJ Problem Set (classical)

3486. Elimination

Problem code: ELIM

Elimination of contestants from a live IQ contest on a TV channel is decided in phases.

Initially at phase 0, N contestants, where $N = 2^n$, $n < 10$, are selected through a special online IQ contest in which a total of M ($M > N$) contestants participate. The contestants are identified by distinct registration numbers 1, 2,..., M . The selected contestants are ranked distinctly from 1 to N according to their performance in the online contest. They are qualified to participate in the live contest.

In the p^{th} phase, p = 1, 2,..., n, K_p contestants participate in the live contest, where $K_p = 2^{n-p+1}$. On the basis of response to questions presented during the show, $K_p/2$ of K_p contestants are ranked distinctly from 1 to $K_p/2$. These $K_p/2$ contestants qualify to participate in the next phase. At the n^{th} phase there are only two contestants and the one selected at this phase is the winner of the contest.

You are required to write a program that identifies the winner of the contest, given the following information:

- INFO_1: Registration numbers of N contestants who are selected through the online IQ contest, in order of the rank in the online IQ contest, and
- INFO_2: A total of N - 1 qualified contestants in different phases; K_2 in phase 1, K_3 in phase 2, ... , and K_{n+1} in phase n . Qualified contestants of different phases appear in order of phases, i.e., phase 1, phase 2, ... , phase n . Further, qualified contestants in a phase, say phase p , appear in the order of the rank in the phase, i.e., the rank in phase p . A qualified contestant of a phase, say phase p , is identified by his/her rank in the previous phase, i.e., in phase p - 1 .

Input

Input may contain multiple test cases. For each case there are two input lines.

The first line gives N integers representing INFO_1 while the second line gives N - 1 integers representing INFO_2.

In each input line integers are separated by space. The input terminates with a line containing `0` as input.

Output

For each test case there is only one output line. The line prints the registration number of the winner of the contest.

Sample Input

```
23 18 6 20
4 2 2
29 57 4 33 5 12 16 18
7 1 5 3 2 1 1
0
```

Sample Output

```
18
29
```

Added by: Walrus
Date: 2008-12-09
Time limit: 1s
Source limit: 50000B
Languages: All
Resource: ACM Kanpur 2006

SPOJ Problem Set (classical)

3488. The Top-Code

Problem code: TOPCODE

A word is a string of two or more letters while a code is a string of one or more distinct words in lexicographic order. Thus a string of letters may represent either a word or a code. An optimum code is a code that contains the maximum number of words.

For a given string of letters there may be one or more optimum codes. The optimum code of top priority is the optimum code that appears at the top when all optimum codes are arranged in lexicographic order.

Given a string of letters, you are required to write a program that finds the following:

- The total number of words, *m* in an optimum code,
- The total number of optimum codes, *n* represented by the string,
- The optimum code of top priority, the top-code.

Input

Input consists of multiple test cases.

For each test case there is only one line of input. It contains a string of at most 100 letters.

A line consisting of a single letter terminates input.

Output

For each test case, present output in two lines.

The first line gives the two integers *m* and *n* defined above. The next line gives the optimum code of top priority, the top-code.

Sample Input

```
aaaaaa
words
lexicographic
a
```

Sample Output

```
2 1
aa aaaa
```

```
1 1
words
3 2
lexic og raphic
```

Kanpur-Kolkata 2004-2005

Added by: Walrus
Date: 2008-12-09
Time limit: 1s
Source limit: 50000B
Languages: All
Resource: ACM Kanpur 2004

SPOJ Problem Set (classical)

3490. Hidden Triangle

Problem code: HIDTRI

Assume that each '0' or '1' in the array represents a point on a plane and the distance between each pair of neighbouring points row wise or column wise is unity. Assume further that every neighbouring pair of 1's, row wise, column wise or diagonally is connected by a line segment. Two line segments emerging from a point, either join together to form a longer line segment or form an angle of 45°, 90° or 135°, thus forming right-angled isosceles triangles. The existence of hidden right-angled isosceles triangles in an array is illustrated in the figure below.

\epsfbox{p3258.eps}

Input

Input consists of multiple test cases.

For each test case the first line gives three integers: the case number *k*, the number of rows *m* and the number of columns *n* of the given array. A space appears between two neighbouring integers.

Each of the next *m* lines gives a string of 0's and 1's of length *n*; the *i*-th line gives the *i*-th row of the array.

Input terminates with a value zero for case number *k*.

Output

For each test case, display output in one line. The line contains the case number *k* and the area of the largest right-angled isosceles triangle hidden in the array. The area is a real number with one digit after the decimal point. If a triangle does not exist then output '0.0' as the area.

Sample Input

```
1 3 3
101
100
101
2 4 6
001001
010101
111111
000001
0
```

Sample Output

```
1 0.0
2 4.0
```

Kanpur-Kolkata 2004-2005

Added by: Walrus
Date: 2008-12-09
Time limit: 1s-3s
Source limit: 50000B
Languages: All
Resource: ACM Kanpur 2004

SPOJ Problem Set (classical)

3492. Braille Transcription

Problem code: BRAILLE

Visually handicapped people use Braille system of codes for reading and writing. The basic Braille symbol (or cell) is composed of six dots arranged in two vertical columns, each column being three dots high. This pattern produces 64 one-cell symbols with character index #($2^0a_1 + 2^1a_2 + 2^2a_3 + 2^3a_4 + 2^4a_5 + 2^5a_6$), where $a_i = 1$, if dot i is embossed up and zero otherwise. The positions of these dots are numbered as follows:

A cell without numbers A cell with numbers

\epsfbox{p3254.eps}

Many Braille codes are in use; one such is the Simple Braille System (SBS). In SBS alphabets in lower case are default characters; letters '`a`, `b`,..., `z`' have character indices '`1`, `2`, ..., `26`' respectively. The ''letter sign'' (dots 5-6) may be used optionally before a string of alphabets in lower case. Each numeric digit 0-9 requires one cell, with character index the digit itself. However the ''number sign'' (dots 3-4-5-6) is added before a string of numerals. In order to revert back to normal alphabets after a string of numerals, the ''letter sign'' is used. The blank cell is used as a space in alphabetic context and zero in numeric context. For example the numerals 1 - 10 are the same as the first ten letters of the alphabet, index of #`a` being 1 and #`j` being 10. SBS allows the contraction of '`th`', by a single cell with dots 1-4-5-6. A two-cell contraction for '`tion`' is allowed with dots 5-6 and dots 1-3-4-5.

There are no single-cell codes for capital letters in SBS. So a ''capital sign'' (dot 6) is inserted before a capital letter. Two ''capital signs'' are inserted to indicate that the string of alphabets that follows is capitalized. To revert back to normal alphabets (or numerals), the ''letter sign'' (or the ''number sign'') is used. SBS ignores punctuation marks altogether.

You are required to write a program for SBS transcription that converts an SBS code to English.

Input

Input consists of multiple test cases.

In each test case there is an SBS code. It is given in three input lines containing a certain number of Braille cells. Each Braille cell is represented by a 3×2 array of `0`'s and `1`'s, appearing in an odd and the next even numbered column of the three input lines, where '`1`' is used for an embossed dot and '`0`' otherwise.

A line that is not a part of an SBS code, containing a single zero in column 1 terminates input.

Output

For each test case, output the SBS code in English, in one line. In case the input does not conform to SBS rules stated above output simply the sign '?'.

Sample Input

```
110010
011100
011110
110110
010100
011110
000011
000101
010110
00001110
00000100
01010110
0
```

Sample Output

```
?
th5
Tion
THE
```

Kanpur-Kolkata 2004-2005

Added by:	Walrus
Date:	2008-12-09
Time limit:	1s
Source limit:	50000B
Languages:	All
Resource:	ACM Kanpur 2004

SPOJ Problem Set (classical)

3495. The Nobel Thief

Problem code: NBLTHIEF

The Nobel Prize of Kabiguru Rabindranath Thagore was stolen from a museum of Viswa Bharati University in West Bengal. The Central Bureau of Investigation (CBI) has been assigned the job to investigate the crime and recover the prize. CBI has identified some suspects and linked each one of them to a distinct subset of a set of clues.

Let there be p suspects and q clues. Integers 1 to p identify suspects while q distinct letter-codes identify clues. The clues are of varying importance. The alphabetic order of letter-codes determines the priority order in the clues; letter-codes '`a`' to '`z`' vary from highest to lowest priority.

Let L_0 be the set of all suspects. Based on a clue '$ \alpha$ ', a subset L of L_0 may be split into two disjoint subsets $L_{+ \scriptstyle \alpha}$ and $L_{- \scriptstyle \alpha}$. The subset $L_{+ \scriptstyle \alpha}$ includes all elements of L that are linked to a subset of clues containing '$ \alpha$ ' and $L_{- \scriptstyle \alpha}$ includes all other elements of L. Let n, $n_{+ \scriptstyle \alpha}$, and $n_{- \scriptstyle \alpha}$ denote respectively the total number of elements in L, $L_{+ \scriptstyle \alpha}$ and $L_{- \scriptstyle \alpha}$. The discriminatory power of a clue '$ \alpha$ ' to discriminate suspects in L is defined by $ \delta_{\alpha}^{}$ = - ($| n_{+ \scriptstyle \alpha} - n_{- \scriptstyle \alpha} |$)

Let L^* denote a set of disjoint subsets of L_0, each subset containing two or more elements. The discriminatory power $ \delta_{\alpha}^{*}$ of a clue '$ \alpha$ ' to discriminate suspects in subsets contained in L^* is defined to be the sum of all $ \delta_{\alpha}^{}$'s corresponding to each subset in L*.

CBI wants to select a set D of dominant clues so that the presence or absence of some or all of these clues can identify each suspect uniquely. The selection is to be done in stages.

Let L^* contain only L_0 initially. At each stage of selection a clue '$ \beta$ ' with highest discriminatory power $ \delta_{\beta}^{*}$ is selected. Selecting the clue '$ \beta$ ' with highest priority breaks tie, if any. After a selection of '$ \beta$ ' each L in L^* is split into disjoint subsets $L_{+ \scriptstyle \beta}$ and ' $L_{- \scriptstyle \beta}$ ' all resulting subsets with less than two elements are dropped from L^*. The process of selection continues until L^* becomes empty. All the clues thus selected form the set of dominant clues D.

You are required to write a program to find the set of dominant clues D.

Input

Input may contain multiple test cases. For each test case input is given in one line. It contains an integer k representing the case number and a certain number of strings of clues. The i-th string represents the subset of clues to which the i-th suspect is linked. A space separates two consecutive fields in input.

Input terminates with an input 0 for the case number *k*.

Output

For each test case, present output in one line. The line contains the case number *k* and a string of letters. The letters in the string correspond to the clues in *D* and appear in the order of their selection.

Sample Input

```
1 cbx cpxb bc brc
2 bac adce cbd d
0
```

Sample Output

```
1  xpr
2  ab
```

Kanpur-Kolkata 2004-2005

Added by: Walrus
Date: 2008-12-09
Time limit: 1s
Source limit:50000B
Languages: All
Resource: ACM Kanpur 2004

SPOJ Problem Set (classical)

3543. Matrica

Problem code: MATRICA

A matrix is a rectangular table of letters. A square matrix is a matrix with an equal number of rows and columns. A square matrix M is called symmetric if its letters are symmetric with respect to the main diagonal ($M_{i,j} = M_{j,i}$ for all pairs of i and j).

For example, the following two matrices are symmetric:

```
AAB          AAA
ACC          ABA
BCC          AAA
```

However, the following two are not:

```
ABCD          AAB
ABCD          ACA
ABCD          DAA
ABCD
```

Given a collection of available letters, you are to output a subset of columns in the lexicographically smallest symmetric matrix which can be composed using all the letters. If no such matrix exists, output "IMPOSSIBLE".

To determine if matrix A is lexicographically smaller than matrix B, consider their elements in row-major order (as if you concatenated all rows to form a long string). If the first element in which the matrices differ is smaller in A, then A is lexicographically smaller than B.

Input

The first line of input contains two integers N (1 <= N <= 30000) and K (1 <= K <= 26). N is the dimension of the matrix, while K is the number of distinct letters that will appear.

Each of the following K lines contains an uppercase letter and a positive integer, separated by a space.

The integer denotes how many corresponding letters are to be used. For example, if a line says "A 3", then the letter A must appear three times in the output matrix.

The total number of letters will be exactly N^2. No letter will appear more than once in the input.

The next line contains an integer P (1 <= P <= 50), the number of columns that must be output.

The last line contains P integers, the indices of columns that must be output. The indices will be between 1 and N inclusive, given in increasing order and without duplicates.

Output

If it is possible to compose a symmetric matrix from the given collection of letters, output the required columns on N lines, each containing P character, without spaces. Otherwise, output "IMPOSSIBLE" (quotes for clarity).

Example

Input:
```
3 3
A 3
B 2
C 4
3
1 2 3
```

Output:
```
AAB
ACC
BCC
```

Input:
```
4 5
E 4
A 3
B 3
C 3
D 3
2
2 4
```

Output:
```
AC
BE
DE
ED
```

Input:
```
4 6
F 1
E 3
A 3
B 3
C 3
D 3
4
1 2 3 4
```

Output:
```
IMPOSSIBLE
```

Warning: large input/output data.

Note: The test data for this problem consist of the official test cases from the contest, as well some cases of my own.

Added by: Neal Wu
Date: 2008-12-17
Time limit: 1s-1.5s
Source limit:50000B
Languages: All
Resource: Croatian Open 08/09 - Contest 3

SPOJ Problem Set (classical)

3544. Binary Search Tree

Problem code: BST

A binary search tree is a tree in which every node has at most two children nodes (a left and a right child). Each node has an integer written inside it. If the number X is written inside a node, then the numbers in its left subtree are less than X and the numbers in its right subtree are greater than X.

You will be given a sequence of integers between 1 and N (inclusive) such that each number appears in the sequence exactly once. You are to create a binary search tree from the sequence, putting the first number in the root node and inserting every other number in order. In other words, run insert (X, root) for every other number:

```
insert (number X, node N)
          increase the counter C by 1
          if X is less than the number in node N
                    if N has no left child
                              create a new node with the number X and set it to be the left child of node N
                    else
                              insert (X, left child of node N)
          else (X is greater than the number in node N)
                    if N has no right child
                              create a new node with the number X and set it to be the right child of node N
                    else
                              insert (X, right child of node N)
```

Write a program that calculates the value of the counter C after every number is inserted. The counter is initially 0.

Input

The first line contains the integer N (1 <= N <= 300 000), the length of the sequence.

The remaining N lines contain the numbers in the sequence, integers in the interval [1, N]. The numbers will be distinct.

Output

Output N integers, each on its own line, the values of the counter C after each number is inserted into the tree.

Example

Input:

```
8
3
5
1
6
8
7
2
4
```

Output:

```
0
1
2
4
7
11
13
15
```

Warning: large input/output data.
Warning: A naive algorithm may not run in time; do not simply implement the above algorithm.

Note: The test data for this problem consist of the official test cases from the contest, as well some cases of my own.

Added by: Neal Wu
Date: 2008-12-17
Time limit: 1s-5s
Source limit: 50000B
Languages: All
Resource: Croatian Open 08/09 - Contest 3

SPOJ Problem Set (classical)

3545. Najkraci

Problem code: NAJKRACI

A road network in a country consists of N cities and M one-way roads. The cities are numbered 1 through N. For each road we know the origin and destination cities, as well as its length.

We say that the road F is a continuation of road E if the destination city of road E is the same as the origin city of road F. A path from city A to city B is a sequence of road such that origin of the first road is city A, each other road is a continuation of the one before it, and the destination of the last road is city B. The length of the path is the sum of lengths of all roads in it.

A path from A to B is a shortest path if there is no other path from A to B that is shorter in length.

Your task is to, for each road, output how many different shortest paths containing that road, modulo 1 000 000 007.

Input

The first line contains two integers N and M (1 <= N <= 1500, 1 <= M <= 5000), the number of cities and roads.

Each of the following M lines contains three positive integers O, D and L. These represent a one-way road from city O to city D of length L. The numbers O and D will be different and L will be at most 10000.

Output

Output M integers, each on its own line - for each road, the number of different shortest paths containing it, modulo 1 000 000 007. The order of these numbers should match the order of roads in the input.

Example

Input:
```
4 4
1 2 5
2 3 5
3 4 5
1 4 8
```

Output:
```
2
3
2
1
```

Input:
```
5 8
1 2 20
```

```
1 3 2
2 3 2
4 2 3
4 2 3
3 4 5
4 3 5
5 4 20
```

Output:

```
0
4
6
6
6
7
2
6
```

Note: The test data for this problem consist of the official test cases from the contest, as well some cases of my own.

Added by: Neal Wu
Date: 2008-12-17
Time limit: 1s-10s
Source limit: 50000B
Languages: All
Resource: Croatian Open 08/09 - Contest 3

SPOJ Problem Set (classical)

3576. Boy Scouts

Problem code: BOYSCOUT

Boy Scouts of New England organize Scout Olympics every year. They ask each team to perform certain tasks, sum up their points, announce winners, and then stay up all night by the fire plying guitar and singing scout songs. This year, Scouts decided to organize the Olympics in one of the most beautiful forests in Maine. There will be only one difficult task. A team picks one tree as a starting point, then goes to another tree in a straight line, then to another, etc. until they come back to the starting one. They win as many points as there are trees on their route. However, they are only allowed to move in a counter-clockwise manner, i.e. after reaching a tree, they can only rotate to the left by less than 180 degrees. Furthermore, when they reach the starting tree again, they should be able to repeat the same route, still going couter-clockwise. More specifically, their path should trace out the boundary of a convex polygon. As they don't bring laptops to the Olympics, the Boy Scouts wants you to compute the maximum possible score a team can achieve.

Input

The first line of the input contains a single integer N (3 <= N <= 100), which is the number of trees in the forest. Each of the next N lines contain two real numbers x and y separated by a space character (-10^6 <= x, y <= 10^6), that represent coordinates of one tree. Coordinates are given with at most two decimal digits. There are no three colinear trees.

Output

Output one integer, the maximum number of points a team can score, followed by a newline.

Example

Input:

```
5
0 0
1.5 -0.25
0 -1
-1 0.5
0.5 1
```

Output:

```
4
```

Added by: Jelani Nelson (Minilek)
Date: 2008-12-22
Time limit: 2s
Source limit:50000B
Languages: All
Resource: MIT Individual Contest 2008

SPOJ Problem Set (main)

3577. Parity

Problem code: PARITY

You are given n binary strings s_1,...,s_n, each of the same length m. Along with each s_i you are given a bit b_i. You are also given some nonnegative integer k and want to know whether there exists a subset S of {0,1,...,m-1} of size at most k such that for each i=1,2,...,n, the bit b_i is the XOR of the bits of s_i at the indices in S. The s_i are 0-indexed strings. Recall that the XOR of a set of bits is 1 if the number of bits equal to 1 is odd, else the XOR is 0 (in particular, the XOR of an empty set of bits is 0). For example, if s_1 = 1010 and S = {0,3}, then b_1 would be 1 (the first bit of s_1) XOR'd with 0 (the last bit of s_1), which is 1. Given n, k, and the strings s_1,...,s_n and their corresponding b_i, find a set S of size at most k which produces the given b_i. You should also detect when no such S exists.

Input

The first line contains n and k, space-separated (1 <= n <= 64, 0 <= k <= 10). n lines then follow, where the ith line contains s_i, followed by a space, then b_i. In a given test case all strings s_i are of the same length m (1 <= m <= 50). k will not be bigger than m.

Output

If no set S of size at most k exists producing the given b_i, output -1 followed by a newline. Otherwise, on the first line output the size of a possible S. If the size of that S is not 0, on the second line, output a space-separated list of the indices in S, followed by a newline. If there exist multiple valid S to be output, you can output any one of your choosing.

Example

Input:
```
3 1
111 1
001 0
011 1
```

Output:
```
1
1
```

Added by: Jelani Nelson (Minilek)
Date: 2008-12-22
Time limit: 5s-30s
Source limit:50000B
Languages: All
Resource: MIT Individual Contest 2008

SPOJ Problem Set (main)

3578. Hashing

Problem code: HASH

Consider the hash function h(y) = a*y + b (mod m) which maps each integer to some integer between 0 and m-1. You are given x,n,c,d and are curious how many of the hash values h(x),h(x+1),...,h(x+n) land in the interval [c,d].

Input

The first line contains a positive integer t, the number of test cases (1 <= t <= 10^5). t lines then follow, where the ith line gives the values a,b,x,n,c,d,m, space-separated, for the ith test case. All given values are non-negative. Also, 1 <= m <= 10^{15}, c <= d < m, a,b < m, x+n <= 10^{15}, and a*(x+n) + b <= 10^{15}.

Output

For each test case in order output the number of i, 0 <= i <= n, such that c <= a*(x+i) + b (mod m) <= d in that test case, followed by a newline.

Example

Input:

```
2
2 3 1 3 0 1 7
1 0 0 8 0 8 9
```

Output:

```
1
9
```

Added by: Jelani Nelson (Minilek)
Date: 2008-12-22
Time limit: 30s
Source limit:50000B
Languages: All
Resource: MIT Individual Contest 2008

SPOJ Problem Set (main)

3579. Disjoint Paths

Problem code: DISJPATH

One of your classes has K students in it who really don't like each other. In fact, they dislike each other so much that they want to find routes to class that don't cross at any intersection so that they won't ever see each other outside of class. Can you find such routes?

Input

The input file will contain multiple cases. The first line of each case is K N, where K is the number of routes you need to find and N is the number of intersections in MIT's floor plan. The intersections are numbered 1,...,N. This is followed by N lines, one for each intersection, containing the indices of the adjacent intersections, separated by spaces. (This means that if the line for intersection 2 contains a 3, then the line for intersection 3 will contain a 2.) Every intersection is adjacent to at least one other intersection. Each case is followed immediately by the next case. The end of the input is indicated by a line containing only "0 0". You may assume that 1 <= K <= 100 and 2 <= N <= 5000. The students all start at intersection 1 and their class is at intersection 2.

Output

For each case, output the case number, in the format "Case #:", followed by a newline. If there are K non-intersecting routes from the start (1) to the end (2), then this must be followed K lines, each one giving a list of corners, in order, on one such route from 1 to 2. If not, then output one line with the word "Impossible". Output a blank line after each test case.

Example

Input:

```
3 5
3 4 5
3 4 5
1 2
1 2
1 2
4 5
3 4 5
3 4 5
1 2
1 2
1 2
0 0
```

Output:

```
Case 1:
1 3 2
1 4 2
```

```
1 5 2

Case 2:
Impossible
```

Added by:	Jelani Nelson (Minilek)
Date:	2008-12-22
Time limit:	30s
Source limit:	50000B
Languages:	All
Resource:	MIT Individual Contest 2008

SPOJ Problem Set (classical)

3580. Company

Problem code: COMPANY

In Plumsoft company, there is a hierarchy among employees, i.e. some of them are bosses to the others. Person A is in charge of person B if there is a sequence of employees P_1 = A, P_2, ..., P_k = B, such that P_1 is P_2's boss, P_2 is P_3's boss, ..., and P_{k-1} is P_k's boss. As Plumsoft is a pretty sane company, you can assume that no two employees can be in charge of each other. The management wants to cut the costs of meetings (they eat a lot of food), so they plan to minimize the number of "A is boss of B" relations by keeping only some of the existing ones. However they want to keep all "A is in charge of B" relations. Please, help them to successfully make this transition.

Input

The first line of the input contains two integers N and M separated by a space character (1 <= N <= 1000, 1 <= M <= 10000). N is the number of employees, and M is the number of "boss" relations in the company. Employees are labeled with numbers 1 through N. Each of the next M lines contain two labels A and B separated by a space character, meaning that A is a boss of B.

Output

In the first line of the output, write a single number M_{min}, which is the minimum number of "boss" relations that the company has to keep. In the next M_{min} lines write the relations that are kept. In each line, write two labels A and B separated by a space character, meaning that A is still a boss of B. If there are multiple solutions, write any of them. Relations can be listed in any order. Each line of the output should be followed by a newline.

Example

Input:

```
5 8
3 5
1 4
4 3
1 3
4 5
1 2
1 5
2 3
```

Output:

```
5
3 5
1 4
4 3
1 2
2 3
```

Added by:	Jelani Nelson (Minilek)
Date:	2008-12-22
Time limit:	2s
Source limit:	50000B
Languages:	All
Resource:	MIT Individual Contest 2008

SPOJ Problem Set (main)

3581. Tree Similarity

Problem code: TREESIM

You are given two labeled and ordered rooted trees T and T' and would like to calculate the *distance* from T to T', which is the minimum number of operations you can perform on T to make it *equivalent* to T'. For each operation you can choose to do one of three things:

1. change the label of one node in T
2. delete a non-root node in T
3. insert a new node in T at a position somewhere below its root

Recall the trees T and T' are ordered, which means that if a non-leaf node has c children, its children are ordered from 1 to c. That is, there is a 1st child, a 2nd child, etc., all the way up to a cth child. When we say a tree X is equivalent to a tree Y, we mean the root of X should have the same label as the root of Y, their roots should have the same number of children (call it c), and the subtree rooted at the ith child of the root of X should be equivalent to the subtree rooted at the ith child of the root of Y for i=1,2,...,c. We now describe what we mean by deletion and insertion of non-root nodes in T. When deleting a non-root node w with d children, let u be its parent and suppose w is u's ith child. Then the first child of w becomes u's ith child, the second child of w becomes u's (i+1)st child, etc. For j < i, the jth child of u remains the same, but for all j > i, the child which was formerly the jth child of u now becomes its (j+d-1)st child (they get "shifted over" due to the insertion of w's children into u's child list). To insert a non-root node w into the tree, we can choose any node u to be its parent, and we can choose any contiguous subsequence (possibly empty) of u's children to become w's children, putting w in their place. When inserting a node, we can give it any label we want at the time of insertion. The root of T can never be deleted, and you can never insert a new node above the root to become the old root's parent. You can, however, change the label of the root.

[IMAGE]

Input

The first line contains n and m separated by a space, the sizes of the trees T and T', respectively (1 <= n,m <= 60). The next n lines describe T. On the ith line is a description of the ith node in the tree: its label, the number of children it has, then a list of its children in order from first to last, all space-separated. The next m lines similarly describe T'. Labels are nonnegative integers fitting in a 32-bit signed integer. The root of each tree is the node which is not the child of any other node in the tree.

Output

On a single line output the minimum number of operations that can be performed on T to make it equivalent to T', followed by a newline.

Example

Input:

```
3 2
6 0
1 2 0 2
4 0
2 1 1
4 0
```

Output:

```
2
```

Added by: Jelani Nelson (Minilek)
Date: 2008-12-22
Time limit: 2s
Source limit: 50000B
Languages: All
Resource: MIT Individual Contest 2008

SPOJ Problem Set (main)

3582. Restaurant Tab

Problem code: RSTAURNT

After eating dinner at a restaurant with some friends, you determine how much money each person owes. Each of you has some cash and some change, but very few of you have exact change. Can you make change for each other so that each person ends up paying the exact right amount?

Input

The input file will contain multiple cases. The first line of each case is N, the number of people at the table. This is followed by N lines, one for each i (- [1,N], containing

x_i $c_{i,1}$ $c_{i,5}$ $c_{i,10}$ $c_{i,25}$ $c_{i,100}$ $c_{i,500}$ $c_{i,1000}$ $c_{i,2000}$ $c_{i,5000}$ $c_{i,10000}$

where x_i is the amount in cents that person i owes and $c_{i,v}$ is the number of coins or bills worth v cents that person i starts out with. For example, person 1 has $c_{1,1}$ pennies, $c_{1,5}$ nickels, etc. Each case is followed immediately by the next case. The end of the input is indicated by a line containing only a zero. You may assume that no person owes more money than they have (i.e. $x_i <= S_j\ j*c_{i,j}$) and that the total amount of money in cents that everyone starts with fits in a signed 32-bit integer. You may also assume that N <= 100000.

Output

For each case, output the case number, in the format "Case #:" (where # is the case number, starting at 1), followed by a space, followed by "YES" if all of the money can be rearranged so that each person ends up paying the correct amount and "NO" if not.

Example

Input:

```
1
10 0 0 1 0 0 0 0 0 0 0
2
0 0 0 0 0 2 0 0 0 0 0
500 0 0 0 0 0 1 0 0 0 0
1
100 4 0 2 3 0 1 0 0 0 0
0
```

Output:

```
Case 1: YES
Case 2: YES
Case 3: NO
```

Added by:	Jelani Nelson (Minilek)
Date:	2008-12-22
Time limit:	30s
Source limit:	50000B
Languages:	All
Resource:	MIT Individual Contest 2008

SPOJ Problem Set (classical)

3591. Patting Heads

Problem code: PATHEADS

It's Bessie's birthday and time for party games! Bessie has instructed the N (1 <= N <= 100,000) cows conveniently numbered 1..N to sit in a circle (so that cow i [except at the ends] sits next to cows i-1 and i+1; cow N sits next to cow 1). Meanwhile, Farmer John fills a barrel with one billion slips of paper, each containing some integer in the range 1..1,000,000.

Each cow i then draws a number A_i (1 <= A_i <= 1,000,000) (which is not necessarily unique, of course) from the giant barrel. Taking turns, each cow i then takes a walk around the circle and pats the heads of all other cows j such that her number A_i is exactly divisible by cow j's number A_j; she then sits again back in her original position.

The cows would like you to help them determine, for each cow, the number of other cows she should pat.

Input

- Line 1: A single integer: N.
- Lines 2..N+1: Line i+1 contains a single integer: A_i.

Output</h3

- **Lines 1..N: On line i, print a single integer that is the number of other cows patted by cow i.**

Example

Input:

```
5
2
1
2
3
4
```

Output:

```
2
0
2
1
3
```

The first cow pats the second and third cows; the second cows pats no cows; etc.

Added by:	**Neal Wu**
Date:	**2008-12-26**
Time limit:	**5s**
Source limit:	**50000B**
Languages:	**All**
Resource:	**USACO Dec 2008**

SPOJ Problem Set (classical)

3678. Cattle Bruisers

Problem code: CATTLEB

Canmuu is out for revenge after being utterly defeated by Bessie in paintball and has challenged Bessie to a video game.

In this game, Bessie starts out at point (BX, BY) in the coordinate grid (-1,000 <= BX <= 1,000; -1000 <= BY <= 1,000), and tries to escape, starting at time 0. She moves continuously at a velocity of (BVX, BVY) units/second (-100 <= BVX <= 100; -100 <= BVY <= 100). Thus, at time 1 she will be at point (BX + BVX, BY + BVY); at time 1.5 she will be at (BX + 1.5*BVX, BY + 1.5*BVY).

Unfortunately, Canmuu has sent N (1 <= N <= 50,000) cattle bruisers to pursue Bessie. At time t=0, cattle bruiser i is at position (X_i, Y_i) (-1,000 <= X_i <= 1,000; -1,000 <= Y_i <= 1,000) with velocity (VX_i, VY_i) units/second (-1,000 <= VX_i <= 1,000; -1,000 <= VY_i <= 1,000).

Each cattle bruiser carries a "proximity" weapon to fire at Bessie; the weapon can hurt Bessie when the cattle bruiser is no further than R (1 <= R <= 2,500) units from her.

Bessie has a shield to protect herself from these attacks. However, she does not want to waste any of her shield's power, so she would like to know the maximum number of cattle bruisers within firing range for any (potentially non-integer) time t >= 0.

In order to avoid precision errors with real numbers, it is guaranteed that the answer produced will be the same whether the attack range is decreased to R-0.0001 or increased to R+0.0001.

Input

* Line 1: Six space-separated integers: N, R, BX, BY, BVX, and BVY * Lines 2..N+1: Line i+1 contains four space-separated integers: X_i, Y_i, VX_i, and VY_i

Output

* Line 1: Print a single integer denoting the maximum number of cattle bruisers within attack range at any point in time.

Example

Input:

```
3 1 0 0 0 2
0 -3 0 4
1 2 -1 1
1 -2 2 -1
```

Input details:

Bessie starts at point (0, 0) and is moving at 2 units per second in the (positive) y-direction. There are 3 cattle bruisers, the first of which starts at point (0, -3) and travels 4 units per second in the y-direction. The maximum distance for a cattle bruiser to be in range of Bessie is 1 unit.

```
Output:
```

```
2
```

Output details:

At time 1.5, Bessie is at point (0, 3), and the three bruisers are at points (0, 3), (-0.5, 3.5), and (4, -3.5). The first two cattle bruisers are within 1 unit of Bessie, while the third will never be within 1 unit of Bessie, so 2 is the most achievable.

Added by: Mir Wasi Ahmed
Date: 2009-01-07
Time limit: 1s
Source limit:50000B
Languages: All
Resource: USACO Holiday Special Contest 2009, Problemsetter - Neal Wu

SPOJ Problem Set (classical)

3679. Moo University - Emergency Pizza Order

Problem code: MOOPIZZA

Moo U's cafeteria has run out of hay and so must order pizzas for the C (1 <= C <= 1,000) calves attending Moo U. Conveniently, a large pizza from the local pizzeria, Pizza Farm, serves exactly one calf.

Pizza Farm is willing to make a pizza for each calf, but, due to the size of the order, has three constraints on the order:

* Although Pizza Farm has long list of T (1 <= T <= 30) vegetarian toppings, each of the pizzas must have exactly K (1 <= K <= T) toppings
* No topping on a pizza can be duplicated (a pizza cannot have onions and onions, for example).
* No two pizzas in the order can have the same set of toppings.

For example, if pizza 1 has onions, green peppers, pineapples, and wheat grass, then it can be the only pizza with that exact set of toppings, although pizza 2 might have onions, green peppers, pineapples, and also olives.

For ordering purposes, the toppings are numbered 1..T.

The calves at Moo U are very picky when it comes to their pizza toppings. Some calves might not like all of the toppings available. A calf will eat a pizza only she likes every single one of the toppings on that pizza. Determine the maximum number of calves that can be fed.

Input

* Line 1: Three integers: C, T, and K. * Lines 2..C+1: Each line of space-separated integers describes which toppings one of the calves likes. The first integer on a line is the number of topping the calf likes. The remaining integers on the line are the toppings that the calf likes.

Output

* Line 1: A single integer, the maximum number of calves that can be fed.

Example

Input:

```
3 2 1
2 2 1
1 1
1 2
```

Input details:

There are three calves. Pizza Farm has two toppings and each pizza must have exactly one topping. The first calf likes both of the toppings, the second calf likes only the first topping, and the third calf likes only the second topping.

Output:

```
2
```

Output details:

There are only two pizzas that can be made: a pizza with topping 1 and a pizza with topping 2. If the first pizza is given to the first calf (since she likes topping 1) and the second pizza to the third calf (since she likes topping 2), two calves will be fed. There is no way to feed all three calves.

Added by: Mir Wasi Ahmed
Date: 2009-01-07
Time limit: 1s
Source limit:50000B
Languages: All
Resource: USACO 2004 March Green Division, Problemsetter - Hal Burch

SPOJ Problem Set (main)

3693. Maximum Sum

Problem code: KGSS

You are given a sequence A[1], A[2], ..., A[N] (0 <= A[i] <= 10^8 , 2 <= N <= 10^5). There are two types of operations and they are defined as follows:

Update:

This will be indicated in the input by a 'U' followed by space and then two integers i and x.

U i x, 1 <= i <= N, and x, 0 <= x <= 10^8.

This operation sets the value of A[i] to x.

Query:

This will be indicated in the input by a 'Q' followed by a single space and then two integers i and j.

Q x y, 1 <= x < y <= N.

You must find i and j such that x <= i, j <= y and i != j, such that the sum A[i]+A[j] is maximized. Print the sum A[i]+A[j].

Input

The first line of input consists of an integer **N** representing the length of the sequence. Next line consists of N space separated integers A[i]. Next line contains an integer **Q**, Q <= 10^5, representing the number of operations. Next Q lines contain the operations.

Output

Output the maximum sum mentioned above, in a separate line, for each Query.

Example

Input:

```
5
1 2 3 4 5
6
Q 2 4
Q 2 5
U 1 6
Q 1 5
U 1 7
Q 1 5
```

Output:

```
7
9
11
12
```

Warning: large Input/Output data, be careful with certain languages

---

Added by:	u.swarnaprakash
Date:	2009-01-10
Time limit:	2s-4s
Source limit:	50000B
Languages:	All
Resource:	Kurukshetra 09 OPC

SPOJ Problem Set (main)

3713. Primitive Root

Problem code: PROOT

In the field of Cryptography, prime numbers play an important role. We are interested in a scheme called "Diffie-Hellman" key exchange which allows two communicating parties to exchange a secret key. This method requires a prime number **p** and **r** which is a primitive root of p to be publicly known. For a prime number p, r is a primitive root if and only if it's exponents r, r^2, r^3, ... , r^{p-1} are distinct (mod p).

Cryptography Experts Group (CEG) is trying to develop such a system. They want to have a list of prime numbers and their primitive roots. You are going to write a program to help them. Given a prime number p and another integer r < p , you need to tell whether r is a primitive root of p.

Input

There will be multiple test cases. Each test case starts with two integers **p** ($p < 2^{31}$) and **n** (1 <= n <= 100) separated by a space on a single line. p is the prime number we want to use and n is the number of candidates we need to check. Then n lines follow each containing a single integer to check. An empty line follows each test case and the end of test cases is indicated by p=0 and n=0 and it should not be processed. The number of test cases is atmost 60.

Output

For each test case print "YES" (quotes for clarity) if r is a primitive root of p and "NO" (again quotes for clarity) otherwise.

Example

Input:

```
5 2
3
4

7 2
3
4

0 0
```

Output:

```
YES
NO
YES
NO
```

Explanation

In the first test case 3^1, 3^2 , 3^3 and 3^4 are respectively 3, 4, 2 and 1 (mod 5). So, 3 is a primitive root of 5.

4^1, 4^2 , 4^3 and 4^4 are respectively 4, 1, 4 and 1 respectively. So, 4 is not a primitive root of 5.

Added by: u.swarnaprakash
Date: 2009-01-14
Time limit: 3s
Source limit:50000B
Languages: All
Resource: Kurukshetra 09 OPC

SPOJ Problem Set (classical)

3723. Snooker

Problem code: SNOOKER

Consider a rectangular snooker table with pockets only at the 4 corners of the rectangle as shown in the image below. Consider all integer points on the boundary of the table. At each point, except the four corners (four pockets), you are allowed to hit the ball at an angle of **45** degrees from the side of the table from which you are hitting the ball.

From any point on the boundary you can hit the ball in two directions and they are considered to be two different ways. For instance in the image shown below, from the point S the ball can be hit in two ways as shown.

Given the dimensions of the board your task is to find the number of ways in which the ball can be hit so that it *eventually* reaches one of the four holes.

Consider the ball to be of negligible size, like a point. Also assume that the ball does not lose energy due to collisions or friction - it keeps moving until it drops into a hole.

[IMAGE]

Input

The input has multiple test cases. Each test case consists of two space separated integers **M** and **N**, 2 <= M,N <= 10^5, representing the dimensions of the table. M=N=0 indicates the end of tests. There are atmost 300 testcases.

Output

For each test case output the number of ways as described, in a separate line.

Example

Input:

```
2 2
2 4
3 5
0 0
```

Output:

```
0
4
24
```

Added by: u.swarnaprakash
Date: 2009-01-16
Time limit: 3s
Source limit:50000B
Languages: All
Resource: Kurukshetra 09 OPC

SPOJ Problem Set (main)

3724. Rainbow Ride

Problem code: RAINBOW

Mr.Raju and his extended family are on vacation in Chennai. They visit MGM and see the Rainbow ride. They want to enjoy the ride. However, there are some problems.

Each person in the family likes some other people in the family. Each person insists that he or she will go on the ride only if all the people whom that person likes and all the people who like that person also go on the ride. If someone doesn't like anyone and no one likes him, he may come for the ride.

You have been roped in to solve this dilemma. Given the weight of the each person in the family, and the list of people they like, and the maximum weight that the Rainbow can bear safely, calculate the maximum number of people in the family who can go on the rainbow.

Input

There will be multiple test cases in the input. For our convenience the family members are numbered from 1 to n. Each test case begins with a line containing two integers **n** (1 <= n <= 1000) and **C** (0 <= C <= 1000), where n is the number of people in the family and C the maximum capacity of the ride in kilograms.

The next line contains n space separated integers with the **i**th integer giving the weight of the **i** th family member. These weights are positive and less than or equal to 200 kilograms. Then n lines follow. Each line contains a sequence of integers separated by spaces. The first integer $\mathbf{k_i}$ gives the number of people in the family person **i** likes, followed by the persons **i** likes. An empty line separates each test case. The end of input is indicated by n=0 and C=0 and it should not be processed.There are atmost 50 test cases.

Output

For each test case output on a separate line the maximum number of persons that can take the ride under the given conditions.

Example

Input:

```
5 200
50 50 50 50 50
1 2
1 3
0
1 5
1 4

3 200
100 100 100
1 2
1 3
```

```
1 1

0 0
```

Output:

```
3
0
```

Added by: u.swarnaprakash
Date: 2009-01-16
Time limit: 5s
Source limit:50000B
Languages: All
Resource: Kurukshetra 09 OPC

SPOJ Problem Set (main)

3725. Taming a T-REX

Problem code: TREX

Although evolution theory suggests that mammals started to dominate this world only after the mass extinction of the dinosaurs, there are some people who say man and dinosaurs may have co-existed for some time. Some argue that man even tamed and used some of these animals like the Flintstones. Shankar is such a believer and Sunil is a skeptic.

One day Sunil asked Shankar, "If your argument is right how would you tame a T-REX and what would you use it for?". Shankar replied, "We can use it for cow transportation from one village to another and we can keep it calm by feeding it at constant intervals". Sunil argued that the T-REX would have a maximum capacity **C** to carry. Let us say the distance is **d** km. If it is long, the T-REX would eat all the cows before it reaches the other village. Shankar argued that he knew a method that takes the maximum possible number of cows **M** to the destination. Sunil replied, "Oh really? I will give a few conditions. The T-REX will eat 1 cow every km until it reaches the destination. It may do so at any time during a 1 km stretch. So,there can not be a situation where the TREX has no cow to eat. If you drop cows in the middle, you can do so only at distances which are a multiple of 1 km. And, finally all the cows at the destination need to be alive i.e you can not cut a cow (fractions are not allowed)".

Shankar was stunned. He needs your help. Given **I** (the number of cows at the starting village) , **d** and **C** find **M**, the maximum number of cows that can be taken to the destination subject to the mentioned constraints.

Input

There will be multiple test cases in the input. The input begins with a line containing a single integer **n**,n <= 300, which gives the number of test cases. Then n lines follow each containing the three integers **I**, 1 <= I <= 10^6, **d**, 1 <= d <= 10^5, and **C**, 1 <= I <= 10^6, in that order separated by spaces. **d** is in kilometers.

Output

For each test case print on a separate line the maximum number of cows that can be transported to the destination village under the given conditions.

Example

Input:

```
2
3000 1000 1000
30 10 10
```

Output:

```
533
5
```

Note:

A few test cases have been added and this problem has been rejudged on January 25th, 2009.

Added by:	**u.swarnaprakash**
Date:	**2009-01-17**
Time limit:	**5s**
Source limit:	**50000B**
Languages:	**All**
Resource:	**Kurukshetra 09 OPC**

SPOJ Problem Set (classical)

3749. Subset Sums

Problem code: SUBSUMS

Given a sequence of N (1 <= N <= 34) numbers S_1, ..., S_N (-20,000,000 <= S_i <= 20,000,000), determine how many subsets of S (including the empty one) have a sum between A and B (-500,000,000 <= A <= B <= 500,000,000), inclusive.

Input

The first line of standard input contains the three integers N, A, and B. The following N lines contain S_1 through S_N, in order.

Output

Print a single integer to standard output representing the number of subsets satisfying the above property. Note that the answer may overflow a 32-bit integer.

Example

Input:
```
3 -1 2
1
-2
3
```

Output:
```
5
```

The following 5 subsets have a sum between -1 and 2:

- 0 = 0 (the empty subset)
- 1 = 1
- 1 + (-2) = -1
- -2 + 3 = 1
- 1 + (-2) + 3 = 2

Added by: Neal Wu
Date: 2009-01-19
Time limit: 1s-2s
Source limit: 50000B
Languages: All

SPOJ Problem Set (classical)

3763. George

Problem code: GEORGE

Last week Mister George visited Croatia. Since Mister George is a very important person, while he was in a street, the police **disallowed entry** to that street, but vehicles that entered the street before Mister George could continue driving.

While Mister George was visiting, Luka drove his truck around town. But because of some of the streets being closed off, he couldn't make his delivery in time and almost lost his job. Although it is late now, he is wondering how he could have planned his delivery better i.e. what would have been the least time needed to make his delivery while Mister George was visiting. He knows the route mister George took.

The city is modeled with intersections and two-way streets connecting them. For each street, Luka knows how much time he needs to traverse it (mister George needs he same amount of time).

For example, if Mister George starts traversing a street during minute 10 and needs 5 minutes to exit it, this street will be blocked during minutes 10, 11, 12, 13 and 14. Luka can enter the street during minutes 9 and earlier, or 15 and later.

Write a program that calculates the least amount of time Luka needs to make his delivery, if he starts driving K minutes after the arrival of Mister George.

Input

The first line contains two integers N and M (2 <= N <= 1000, 2 <= M <= 10 000), the number of intersections and the number of streets. The intersections are numbered 1 to N. The second line contains four integers A, B, K and G (1 <= A, B <= N, 0 <= K <= 1000, 0 <= G <= 1000). These are, in order:

- The intersection where Luka starts;
- The intersection Luka must get to;
- The difference in starting times between mister George and Luka (Luka starts at intersection A exactly K minutes after mister George starts his route);
- The number of intersections on Mister George's route.

The third line contains G integers, the labels of intersections mister George will visit. Every pair of adjacent integers denotes a street he will traverse. That street will exist and Mister George will traverse every street at most once. Each of the following M lines contains three integers A, B and L, meaning that there is a street between intersection A and B, and it takes L minutes to traverse. L will be between 1 and 1000.

Output

Output the least amount of time (in minutes) Luka needs to make his delivery.

Example

Input:
```
6 5
1 6 20 4
5 3 2 4
1 2 2
2 3 8
2 4 3
3 6 10
3 5 15
```

Output:
```
21
```

Input:
```
8 9
1 5 5 5
1 2 3 4 5
1 2 8
2 7 4
2 3 10
6 7 40
3 6 5
6 8 3
4 8 4
4 5 5
3 4 23
```

Output:
```
40
```

Croatian Open Competition in Informatics (COCI) - 2007/2008 Contest #6

Added by: Andrés Mejía-Posada
Date: 2009-01-25
Time limit: 1s
Source limit: 50000B
Languages: All
Resource: Croatian Open Competition in Informatics (COCI) - 2007/2008 Contest #6

SPOJ Problem Set (classical)

3791. Street

Problem code: STREET

There are **n** lots on one side of a street (where **n** <= 500). We would like to erect at most **k** apartment buildings on these lots. Each building must occupy an interval of at most **t** consecutive lots. Moreover, each lot **i** has a height restriction **r[i]** (where **r[i]** <= 100). A building cannot exceed any of the height restriction of any lot on which it is built (that is, the maximal height of the building that can be erected on lot **i** to **j** is:

$$H = \min\{r[i], r[i + 1], ..., r[j]\})$$

Hence, the maximum usable facade space of the building is: **H × (j - i + 1)**. We would like to have a program to select at most **k** non-overlapping intervals to erect the buildings such that the total usable facade space is maximized.

Example 1

Consider a street of length 10. The height restriction of each lot is as follows:

```
7, 3, 12, 11, 13, 4, 8, 6, 6, 20
```

Suppose we would like to erect at most **k** = 2 buildings and each building occu- pies at most **t** = 4 lots. Then, to maximize the total usable facade space, we choose two intervals **r**[3..5] = (12, 11, 13) and **r**[7..10] = (8, 6, 6, 20) (see "Example 1" in the figure below). The maximum usable facade space is 3 * min{12, 11, 13} + 4 * min{8, 6, 6, 20} = 57.

Example

Example 2

Suppose we would like to erect at most **k** = 3 buildings on the same street with the same height restrictions as in Example 1, and each building occupies at most **t** = 4 lots. Then, to maximize the total usable facade space, we choose three intervals **r**[3..5] = (12, 11, 13), **r**[7..9] = (8, 6, 6) and **r**[10..10] = (20) (see "Example 2" in the figure above). The maximum usable facade space is 3 * min{12, 11, 13} + 3 * min{8, 6, 6} + 1 * 20 = 71.

Input

The input file is as follows: The first line contains three integers **n**, **k**, and **t** separated by a space character, where 1 <= **n** <= 500, 1 <= **k** <= n, and 1 <= **t** <= n. The rest of the **n**lines contain **n** positive integers representing the height restriction for the **n** lots. For Example 1, the input file looks like:

```
10 2 4
7
3
12
11
13
4
8
6
6
20
```

The input should be read from the standard input, and your program will be run several times, each one with one of the test cases.

Output

The output file contains an integer which is the maximum usable facade space. For the above example, the output file looks like:

```
57
```

National Olympiad in Informatics (NOI) - 2007

Added by: Andrés Mejía-Posada
Date: 2009-01-31
Time limit: 3s
Source limit: 50000B
Languages: All
Resource: National Olympiad in Informatics (NOI) - 2007

SPOJ Problem Set (classical)

3831. Lubenica

Problem code: LUBEN

Children in school are having fun instead of listening to the teacher. With their iPhone devices the children throw watermelons at each other on the Facebook social site.

The game started when Goran threw one watermelon at each of his friends during the first class that day. During subsequent classes, all children (including Goran) behaved like this:

- If they had been hit by an odd number of watermelons during the previous class, they threw exactly one watermelon at each of their friends;
- If they had been hit by an even number of watermelons (including zero), then they hit each of their friends with two watermelons.

The children are numbered 1 through N, Goran obviously being number 1. The friend relationships between the children are also known.

Write a program that will calculate the total number of watermelons thrown after H classes.

Input

The first line contains two integers N and H (1 <= N <= 20, 1 <= H <= 1 000 000 000), the number of kids and classes.

Each of the following N lines contains a string of N characters '0' or '1'. The character (A, B) in this matrix is '1' if children A and B are friends.

No child will be their own friend. The input matrix will be symmetric

Output

Output the number of watermelons after H classes.

Example

Input:
```
5 3
01000
10110
01000
01001
00010
```

Output:
```
26
```

Added by:	Race with time
Date:	2009-02-08
Time limit:	1s
Source limit:	50000B
Languages:	All
Resource:	COCI 2008/2009 - Contest #5, 7th February 2009

SPOJ Problem Set (classical)

3832. Kruska

Problem code: KRUS

Aladdin has become bored of life at the palace. He has a steady job, his wife Jasmine, kids are on the way and life is becoming monotonous. Prompted by all this, he has decided to have one more adventure before settling down.

He has decided to find the Golden Pear, an extremely valuable legendary artifact that no one has been able to find.

The desert Aladdin is searching is can be modeled as an N×N grid of cells. Rows and columns are numbered 1 through N top to bottom and left to right. Some of the cells in the desert contain wizards that help Aladdin's quest in an unusual way.

Aladdin starts his quest in the top left corner of the desert on a Monday facing right. His movement involves repeating these steps:

1. If the current cell contains a wizard that is awake, then Aladdin turns 90 degrees left or right, depending on what the wizard says.

2. If moving forward would take Aladdin out of desert, he turns 180 degrees.

3. Aladdin moves forward one cell and it takes him exactly one day.

For each wizard we know his location and his activity schedule for each day of the week. The schedule is a string of exactly seven letters 'L', 'R' or 'S', each character telling us what the wizard does on one day of the week (starting with Monday). The letter 'L' means that Aladdin will be told to turn left, 'R' that Aladdin will be told to turn right, while 'S' means the wizard sleeps that day.

An old prophecy says that after K changes in direction (in steps 1 and/or 2) Aladdin will find the Pear.

Write a program that calculates how many days the search will last, according to the ancient prophecy.

Input

The first line contains two integers N and K (2 <= N <= 200, 1 <= K <= 1 000 000 000), the size of the desert and the number of direction changes in the prophecy.

The second line contains an integer M (0 <= M <= 10 000), the number of wizards.

Each of the following M lines contains two integers R and C (1 <= R, C <= N), and a string of seven letters 'L', 'R' or 'S'. The numbers represent the row and column where the wizard is located, while the string is his schedule.

No two wizards will share the same cell, nor will there be a wizard in cell (1, 1).

Output

Output the length of the search in days.

Example

Input:
```
5 2
2
1 3 RRSRRRR
1 5 RRRRLRR
```

Output:
```
4
```

Added by:	Race with time
Date:	2009-02-08
Time limit:	1s
Source limit:	50000B
Languages:	All
Resource:	COCI 2008/2009 - Contest #5, 7th February 2009

SPOJ Problem Set (classical)

3833. Tresnja

Problem code: TRES

Lana lives in a small but merry village. There is a row of cherry trees next to the main street. Lana numbered the trees with consecutive integers starting with 1.

After much studying, Lana noticed that the number of the tree uniquely determines the amount of cherries the tree gives.

For one tree, consider consecutive groups of digits in the tree's number. For each group of digits, multiply the digit by the square of the length of the group. Adding these numbers for all groups gives the total number of cherries the tree gives.

For example, in tree number 77744007, the groups are 777, 44, 00 and 7. The amount of cherries will be 7.3^2 + 4.2^2 + 0.2^2 + 7.1^2 = 86 units.

The time has come to pick the cherry trees and the villagers have agreed to pick all trees numbered A through B (inclusive). Write a program that will calculate the total amount of cherries picked.

Input

Input consists of two integers A and B (1 <= A <= B <= 10^15), the first and last trees to be picked.

Output

Output a single integer, how many units of cherries will be picked.

Example

Input:
```
100 111
```

Output:
```
68
```

Added by: Race with time
Date: 2009-02-08
Time limit: 1s
Source limit:50000B
Languages: All
Resource: COCI 2008/2009 - Contest #5, 7th February 2009

SPOJ Problem Set (classical)

3863. Area of circles

Problem code: VCIRCLES

Given N circles. Calculate the total area that these N circles cover.

Input

First line: N (1 <= n <= 50)

- In the i-th line of the next n lines contains 3 integers Xi, Yi and Ri, separated by spaces. These are the coordinates of the center and the radius of the i-th circle
(-10000<=Yi-Ri,Yi+Ri,Xi-Ri,Xi+Ri<=10000)

Output

The total area that these N circles cover with 5 digits after decimal point

Example

Input:
```
2
5 6 3
5 5 5
```

Output:
```
78.53982
```

Added by: Phenomenal
Date: 2009-02-14
Time limit: 3s
Source limit:50000B
Languages: All
Resource: IOICAMP

SPOJ Problem Set (classical)

3865. Reljef

Problem code: RELJEF

Two groups of cavemen got into a land dispute and decided to settle it the old fashion-way, by throwing sticks at each other. The fight was organized in a cave, high enough that the ceiling is of no concern, but mineral deposits on the ground get in the way of flying sticks.

The cave can be divided into R rows and C columns, so that the entire cave consists of R×C cells. Each cell in the cave is either empty or contains a chunk of mineral. Two chunks of minerals are part of the same cluster if they are adjacent in one of the four main directions (up, down, left, right).

One group of cavemen is on the left side of the cave, the other on the right side. The groups alternate throwing sticks at the other group; first a group chooses the height at which the stick will fly and then (climbing on each others' shoulders as necessary) they throw it and the stick flies horizontally through the cave at the chosen height.

If the stick hits a chunk of mineral on the way, it destroys the chunk, the cell becomes empty and the stick stops its journey.

When a chunk is destroyed, it is possible that a cluster falls apart. If a newly created cluster would float in the air, then it falls down because of gravity. While falling, the cluster does not change shape i.e. all chunks in it fall together. As soon as some chunk in the falling cluster lands on a chunk from a different cluster or the ground, the entire cluster stops falling. Of ourse, if a cluster lands on another, they merge and become one.

Your program will be given the layout of minerals in the cave and the heights at which sticks were thrown. Determine the layout of minerals after the sticks are exchanged.

Input

The first line contains two integers R and C (1 <= R, C <= 100), the dimensions of the cave.

Each of the following R lines will contain C characters. The character '.' represents an empty cell, while the letter 'x' represents a chunk of mineral.

The next line contains an integer N (1 <= N <= 100), the number of sticks thrown.

The last line contains N integers separated by spaces, the heights at which sticks were thrown. All heights will be between 1 and R (inclusive), with height 1 being the bottom of the matrix and height R the top. The first tick is thrown left to right, the second right to left and so on.

No cluster will initially float in the air. Also, the input data will be such that at no point will two or more clusters fall simultaneously, so that there will be no ambiguous situations.

Output

The output should consist of R lines, each containing C characters, the final layout of the cave, in the same format as in the input.

Example

Input:

```
8 8
........
........
...x.xx.
...xxx..
..xxx...
..x.xxx.
..x...x.
.xxx..x.
5
6 6 4 3 1
```

Output:

```
........
........
........
........
.....x..
..xxxx..
..xxx.x.
..xxxxx.
```

Added by: Race with time
Date: 2009-02-15
Time limit: 1s
Source limit:50000B
Languages: All
Resource: COCI 2008/2009 - Croatian Regional

SPOJ Problem Set (classical)

3866. Finding Palindromes

Problem code: VPALIN

A word is called a palindrome if we read from right to left is as same as we read from left to right. For example, "dad", "eye" and "racecar" are all palindromes, but "odd", "see" and "orange" are not palindromes.

Given n palindromes, you can generate n × n pairs of them and concatenate the pairs into single words. The task is to count how many of the so generated words are palindromes.

Input

The first line of input file contains the number of strings n. The following n lines describe each string: The i+1-th line contains the length of the i-th string li, then a single space and a string of li small letters of English alphabet. You can assume that the total length of all strings will not exceed 2,000,000. Two strings in different line may be the same.

Output

Print out only one integer, the number of palindromes.

Example

Input:

```
3
1 a
2 bb
2 aa
```

Output:

```
5
The 5 palindromes are:
aa aaa aaa bbb aaaa
```

Added by: Phenomenal
Date: 2009-02-15
Time limit: 5s-7s
Source limit:50000B
Languages: All
Resource: POI 2006

SPOJ Problem Set (classical)

3867. Who is The Boss

Problem code: VBOSS

Several surveys indicate that the taller you are, the higher you can climb the corporate ladder. At TALL Enterprises Inc. this "de facto standard" has been properly formalized: your boss is always at least as tall as you are. Furthermore, you can safely assume that your boss earns a bit more than you do. In fact, you can be absolutely sure that your immediate boss is the person who earns the least among all the employees that earn more than you and are at least as tall as you are. Furthermore, if you are the immediate boss of someone, that person is your subordinate, and all his subordinates are your subordinates as well. If you are nobody's boss, then you have no subordinates. As simple as these rules are, many people working for TALL are unsure of to whom they should be turning in their weekly progress report and how many subordinates they have. Write a program that will help in determining for any employee who the immediate boss of that employee is and how many subordinates they have. Quality Assurance at TALL have devised a series of tests to ensure that your program is correct. These test are described below.

Input

On the first line of the input is a single positive integer n, telling the number of test scenarios to follow. Each scenario begins with a line containing two positive integers m and q, where m (at most 30000) is the number of employees and q (at most 200) is the number of queries. The following m lines each list an employee by three integers on the same line: employee ID number (six decimal digits, the first one of which is not zero), yearly salary in Euros and finally height in mm (1 mm = 10-6 meters - accuracy is important at TALL). The chairperson is the employee that earns more than anyone else and is also the tallest person in the company. Then there are q lines listing queries. Each query is a single legal employee ID.

The salary is a positive integer which is at most 10 000 000. No two employees have the same ID,and no two employees have the same salary. The height of an employee is at least 1 000 000 mm and at most 2 500 000 mm.

Output

For each employee ID x in a query output a single line with two integers y k, separated by one space character, where y is the ID of x's boss, and k is the number of subordinates of x. If the query is the ID of the chairperson, then you should output 0 as the ID of his or her boss (since the chairperson has no immediate boss except, possibly, God).

Example

Input:

```
2
3 3
123456 14323 1700000
123458 41412 1900000
123457 15221 1800000
```

```
123456
123458
123457
4 4
200002 12234 1832001
200003 15002 1745201
200004 18745 1883410
200001 24834 1921313
200004
200002
200003
200001
```

Output:

```
123457 0
0 2
123458 1
200001 2
200004 0
200004 0
0 3
```

Added by: Phenomenal
Date: 2009-02-15
Time limit: 1s
Source limit: 50000B
Languages: All
Resource: NWERC 2003

SPOJ Problem Set (classical)

3870. Military Story

Problem code: VMILI

Military headquaters plan to develop a better protection for a spaceport. They suppose that the spaceport would be best protected if it is surrounded with as many fences as possible and each fence is patroled by armed guards. The corresponding order was issued and military engineers started to develop a project.

Wishing to be promoted, sergeant Stupid sent soldiers to dig in fence poles before the project was actually ready. Without much thinking, the soldiers put a lot of poles at random. Help the sergeant to decide how to make barbwire fences using the poles so that the number of fences is maximal.

Input

The first line contains an integer 3 <= N <= 4000, which is the number of the poles. Each of the following N lines contains two integers 0 <= x, y <= 10000, which are the coordinates of a corresponding pole. No two poles have the same position.

Output

The output should contain a single integer number, which is the maximal possible number of nested fences that can be constructed. Each fence is a closed polygonal line without self-crossing whose vertices are poles. Different fences should not have common points.

Example

Input:

```
4
100 100
200 100
100 200
300 300
```

Output:

```
1
```

Added by: Phenomenal
Date: 2009-02-16
Time limit: 1s
Source limit:50000B
Languages: All
Resource: acm.timus.ru

SPOJ Problem Set (classical)

3871. GCD Extreme

Problem code: GCDEX

Given the value of N, you will have to find the value of G. The meaning of G is given in the following code

```
G=0;

for(k=i;k< N;k++)

for(j=i+1;j<=N;j++)

{

G+=gcd(k,j);

}
```

/*Here gcd() is a function that finds the greatest common divisor of the two input numbers*/

Input

The input file contains at most 20000 lines of inputs. Each line contains an integer N (1<N<1000001). The meaning of N is given in the problem statement. Input is terminated by a line containing a single zero.

Output

For each line of input produce one line of output. This line contains the value of G for the corresponding N. The value of G will fit in a 64-bit signed integer.

Example

Input:

```
10
100
200000
0
```

Output:

```
67
13015
143295493160
```

Time limit has been changed. Some AC solutions get TLE

Added by: Phenomenal
Date: 2009-02-16
Time limit: 3s
Source limit:50000B
Languages: All
Resource: ACM World Final Warm up 1 - 2008

SPOJ Problem Set (classical)

3872. Party At School

Problem code: VPARTY

Today there is a party at school. N girls and M boys attend this party. The principal wants to give some presents to the girl and he decides to make the boy do that. Each boy has told the principal the name of the two girls that he wants to give the present to so the principal wants the form teacher to choose some boys to do that. However, he is short in cash now so he wants the number of boys selected is minimum but he also wants all the girls to have at least one present. If you select a boy, then he will give the presents to both of the girls he has chosen.

Input

The first line contains two integers N and M (2 <= N <= 1000, 1 <= M <= 1000) which is the number of the girls and boys.

The i-th line of the following M lines contains two integers ai and bi (1 <= ai, bi <= N) which are the two girls that the i-th boy wants to give present to

Output

One single integer number, which is the minimum boys the form teacher should choose.

Example

Input:
```
3 3
1 2
2 3
1 3
```

Output:
```
2
```

Added by: Phenomenal
Date: 2009-02-16
Time limit: 1s
Source limit:50000B
Languages: All
Resource: IOICAMP

SPOJ Problem Set (classical)

3884. When (You Believe)

Problem code: WHEN

It's said: there can be miracles, when you believe. The following programming language shows the power of "when". It has a very simple (case insensitive) grammar, shown below:

```
PROGRAM := WHEN | PROGRAM WHEN
WHEN := 'when ' EXPRESSION <ENTER> STATEMENTS 'end when' <ENTER>
STATEMENTS := STATEMENT | STATEMENTS STATEMENT
STATEMENT := PRINT | SET
PRINT := 'print '  EXPRESSION_LIST <ENTER>
SET := 'set ' ASSIGNMENT_LIST <ENTER>
EXPRESSION_LIST := EXPRESSION | EXPRESSION_LIST ',' EXPRESSION
ASSIGNMENT_LIST := ASSIGNMENT | ASSIGNMENT_LIST ',' ASSIGNMENT
ASSIGNMENT := VARIABLE '=' EXPRESSION
EXPRESSION := '(' EXPRESSION OP EXPRESSION ')' | VARIABLE | NUMBER
OP :=  '<' | '+' | '-' | 'and' | 'or' | 'xor'
VARIABLE := '$' NOT_DOLLAR_STRING '$'
NUMBER := DIGIT | NUMBER DIGIT
DIGIT := '0' | .. | '9'
NOT_DOLLAR_STRING := any sequence of printable characters (including blanks)
                     that does not contain a $ symbol.
```

In the above, any string enclosed in single quotes are to be treated literally. <ENTER> is the end of line.

In words, Spaces are allowed before or after any literal except inside a number. Spaces are allowed in variable names, and each non-empty sequence of spaces is treated as a single underscore, so the following refer to the same variable:

```
$Remote Switch#1$
$Remote_Switch#1$
$Remote   switch#1$
```

All numbers appearing in the program will be integers between 0 and 1000000000, inclusive. All variable and literal values are integers between -1000000000 and 1000000000, inclusive. All variables are global and initially zero. The programs you will be tested on will never have an `EXPRESSION` that evaluates to a value outside of this range. The logical operators evaluate to 0 for false and 1 for true, and treat any nonzero value as true.

Running the program amounts to executing all the active when clauses until none are active. More specifically, the active list of when clauses is initially empty, then the following steps are repeated:

- In the order they appear in the program, the conditions of all when clauses that are not currently active are evaluated. If true, the clause is added to the end of the active list, with its first statement marked as "ready". Each active when clause has one "ready" statement.
- If the active list is empty after this step, the program terminates.
- The "ready" statement from the "current" when clause (initially the first clause in the active list) is executed.

- The statement marked as "ready" is advanced, removing the when clause from the active list if this is the last statement in the "current" when clause.
- The when clause marked as "current" is advanced, cycling to the beginning of the active list if the end is reached.

In other words, inactive when conditions are evaluated to determine if these clauses are added to the active list. Then one statement (set or print) is executed from the current active when clause. If this is the last statement in that clause, it is removed from the active list. One the next iteration, one statement is executed from the next active when clause, etc.

A set statement executes all the assignments concurrently, so that

```
set $x$=$y$,$y$=$x$
```

swaps the values of x and y. The same variable cannot appear twice on the left hand part of the same set statement (so set x=1,x=2 is illegal).

A print statement evaluates and prints the given expressions in the output, separated by commas and followed by a new line. So

```
print 1,(2+3)
```

results in the line

```
1,5
```

in the output.

Input

The input consists of a single syntactically correct program. You may assume that the program will not execute more than 100000 set statements and 100000 print statements.

Output

Print the output produced by executing the given program. Both the input and output file will not exceed 100KB.

Example

Input:

```
When ($Mr. Bill$<5)
   Set $mr._bill$=($mr.  bill$+1),$Y$=($Y$+10)
End When
When ($mr. Bill$<10)
  Set $MR. BILL$=($mr. bill$+1)
  Print $mr. bill$,$Y$
End When
```

Output:

```
3,20
6,40
```

```
7,40
8,40
9,40
10,40
```

Added by: Blue Mary
Date: 2009-02-17
Time limit: 2s
Source limit: 6666B
Languages: All except: C99 strict
Resource: Whitney Houston: When You Believe

SPOJ Problem Set (classical)

3894. Bouncing Balls

Problem code: BOBALLS

Consider a grid having NxM squares. The top left square is (0,0) and the bottom right is (N-1,M-1). Each square in the grid is either occupied by a platform or has a number written on it. Two balls are released from the top of the grid (from locations (0,Y1) and (0,Y2), 0 <= Y1,Y2 < M). Each ball falls down vertically, unless either it falls down the bottom row, or encounters a platform beneath. When the ball encounters a platform beneath, it rolls either to the left or to the right, each with an equal probability. The score obtained by a ball is the sum of the numbers on the squares that it passes (including the starting square). However, if both the balls pass over the same square, points corresponding to that square are obtained only once, and not twice. Your goal is to choose Y1 and Y2 such that the expected score obtained by the two balls is maximized. For example, consider the grid below : (P reprsents a platform)

```
N = 6, M = 6
112214
211243
30PPP2
423378
1P9753
220102
```

Here, dropping a ball from position (0,3) could result in one of the following three scores :

```
1) 2 + 2 + 1 + 1 + 0 + 2 + 4 + 1 + 2 = 15
2) 2 + 2 + 1 + 1 + 0 + 2 + 3 + 9 + 0 = 20
3) 2 + 2 + 4 + 3 + 2 + 8 + 3 + 2 = 26
```

The expected score is (considering only 1 ball) :

```
1/2*(1/2*(15) + 1/2*(20)) + 1/2*(26)
```

Input

The first line contains the number of test cases.

The first line for each test case consists of N and M.

Lines 2..N+1 for each test case consist of M characters each. Each character is either a digit from 0 to 9, or the letter 'P'.

Output

The maximum expected score accurate upto 4 decimal places.

Example

Input:

```
4
5 5
53214
53214
53214
54214
53214
5 5
00000
0P0P0
00000
01P20
00000
5 5
09090
0P0P0
00000
01P20
00000
6 6
112214
211243
30PPP2
423378
1P9753
220102
```

Output:

```
45.0000
2.5000
19.3125
35.5000
```

Constraints

Dataset 1: 1 <= number of test cases <= 100

3 <= N,M <= 100

All possible paths from the top will eventually lead to the ball falling from the bottom. There will be no "rebounds" possible. If there is a 'P' on square (x,y), there will not be a 'P' on squares (x-1,y-1) or (x-1,y+1) or (x+1,y-1) or (x+1,y+1). Also, platforms will not occur on the boundaries of the grid. Thus, the X coodinate of a platform will never be 0 or N-1, and the Y coordinate will never be 0 or M-1. The test case was generated to guarantee that any answer with absolute error in 1e-9 will got accepted. Time limit: 7s

---

Added by: Race with time
Date: 2009-02-19
Time limit: 7s
Source limit:50000B
Languages: All
Resource: Code Craft 09

SPOJ Problem Set (classical)

3920. Lucius Dungeon

Problem code: BYTESE1

LUCIUS' DUNGEON (5 points) There are a set of rooms in a dungeon, arranged in an M × N rectangular grid. In one of the rooms, evil Lucius Malfoy has imprisoned Hermione, owing to his hatred towards the mudbloods. The noble Harry potter is on his way to rescue Hermione. Harry potter starts in the room at the top left corner of the grid, which is labeled (1,1). Each room contains some guards. It takes some time for Harry potter to kill all the guards in the room he is in. The time taken to kill the guards varies from room to room. Once he has killed all the guards in a room, he can move on to any one of its neighbors by going left, right, up or down, provided, of course, that there is a neighboring room in the corresponding direction. He cannot move diagonally.
>Lucius Malfoy, knowing that Harry Potter is on his way, has set a time bomb that will kill Hermione after T seconds. You will be given the position of Hermione, the time left for the bomb to go off and the time it takes for Harry to kill the guards in each of the rooms in the dungeon. Your task is to determine if it is possible for Harry to reach Hermione and save her by defusing the bomb before the T seconds expire. For example, suppose the dungeon is described by the following grid of numbers where the numbers start from (1,1):
> 2 3 2
> 2 5 1
> 5 3 1
> 3 1 1
> The number at position (i,j) indicates the time taken for Harry Potter to overpower the guards in room (i,j). Suppose Hermione is in the room at position (4,2). If T = 10. There is no way Harry Potter can reach Hermione in time. However, if M = 15, Harry Potter can reach Hermione with 4 seconds to spare, as follows. Starting from (1,1), he moves right to (1,2) and then (1,3), comes down all the way to (4,3) and then moves to (4,2). This takes 11 seconds (note that he must also overpower the guard in the room where Hermione is incarcerated). You can check that he cannot reach Hermione with more than 4 seconds to spare by any route.
>Note: If Harry reaches Hermione at exactly 'T' seconds from the start then the answer is "YES" with 0 seconds to spare.

Input<h3>

The first line consists of the number of test cases K (1<=K<=20). In each test case, the first line contains two integers M and N indicating the number of rows and columns in the rectangular dungeon(1 <= M,N<= 100). Next M lines contain N integers (single digits only). The jth integer on ith line is the time taken to overpower the guards at room (i,j). The last line in each test case, contains three integers a, b and T, where (a,b) is the position of the cell where Hermione is held and T is the amount of time before the bomb goes off.

Output

For each of the test case, if it is not possible for Harry Potter to save Hermione then print NO. Otherwise, print two lines. The first line should say YES. The second line should contain a single integer indicating the maximum possible time to spare when Harry Potter rescues the Hermione.

Example

Input:

```
2
4 3
2 3 2
2 5 1
5 3 1
3 1 1
4 2 15
2 2
1 2
1 1
2 2 2
```

Output:

```
YES
4
NO
```

Added by:	**Paritosh Aggarwal**
Date:	**2009-02-21**
Time limit:	**1s-8s**
Source limit:	**50000B**
Languages:	**All**

SPOJ Problem Set (classical)

3921. The Great Ball

Problem code: BYTESE2

The Great Ball (5 points) Hogwarts has organized The Great Ball to welcome the schools participating in the Triwizard Tournament. The ball is being held in the Great Hall and The Weird Sisters have been called to play the band. The students drift in to dance and then go out when they get tired. Hagrid is stationed at the gate and is noting down the time at which people enter and leave the hall. At the end of the day, he wonders what the maximum number of dancers was during the course of the ball. For convenience, he writes down for each person entering, the number of minutes from the start of the ball at which the person entered and left. The door of the hall is narrow, so at any time, either one person can enter or one person can exit, but not both. For example, suppose the observations noted down by Hagrid are the following:

Serial No	Enters at	Leaves at
1	1	7
2	2	4
3	6	9
4	3	8
5	5	10

Each line denotes the entry time and exit time of one person. (The identity of the person is not important - the same person may enter and leave many times. For instance, in the example, it might well be that the entries and exits recorded at serial no. 2 & 5 refer to the same person). In this example, the maximum size of the dancers during the ball was 4. This was achieved between time 6 & 7. Hagrid is not good at Math so he requires your help. Your task is to read the list of entry and exit times and compute the maximum number of dancers during the ball.

Input

The first line is a single integer, T (1<=T<=100), which is the number of test cases. For each of the test case, the first line contains a single integer N, (1<=N<=100), the number of entries and exits recorded. This is followed by N lines. Each of these lines consists of two integers, separated by a space, describing the entry and exit time of that person. The entry and exit times are guaranteed to be distinct, and the entry time will be less than the exit time. The constraint on entry and exit times is 10000000.

Output

A total of T lines each of them containing a single integer, denoting the maximum number of dancers during the ball.

Example

Input:

```
1
5
1 7
2 4
6 9
3 8
5 10
```

Output:

```
4
```

Added by: Paritosh Aggarwal
Date: 2009-02-21
Time limit: 1s
Source limit: 50000B
Languages: All

SPOJ Problem Set (classical)

3922. Mystical River

Problem code: BYTESM1

Harry Potter was punished for his impudence by Snape and was sent to the forbidden forest for detention. He has now lost his way amidst the forest. He has come across 3 mystical rivers flowing in his way to the school but he is not sure which river does he have to cross. Alongside each mystical river are stones after every meter with a number written on them. The numbers start with 1, 3 and 9 respectively for the three rivers. The number on a stone is equal to the no. on the previous stone plus the sum of its digits. For example, if one no. is 403 then the next no. is 403+4+0+3=410, the next no. is 410+4+1+0=415 and so on. Harry calls this a 'stone river'. Harry knows that he has to cross at the stone number which is part of the stone river of the number he remembers. He has to now find the point of intersection of the stone river generated by the no. 'N' that he remembers, with the rivers of 1 or 3 or 9. Your task is to find which mystical river intersects the stone river generated by Harry's number. Also find the point of intersection at which Harry can cross the mystical river. For example, if Harry remembers that he has to cross at the stone number which is part of the stone river of 29 then he has to cross at the stone number 107 which is also in the river of number 1. Stone river of 29: 29,40,44,52,59,73,83,94,107,.... River of 1: 1,2,4,8,16,23,28,38,49,62,70,77,91,101,103,107,.... Both the rivers meet at 107 and hence Harry has to cross at stone number 107. It is possible that the stone river might intersect with more than one of the mystical rivers. In that case, output the least intersection number.

Input

The first line consists of a single integer 'T' (1<=T<=50) which is the total number of test cases. Each of the next T lines consist of a single integer 'N' (1<=N<=99999999) which is the number that Harry remembers.

Output

A total of T lines, where each line consists of two integers separated by a space. The first integer represents the river out of 1, 3 and 9 which intersects with the stone river of 'N'. The second integer represents the stone number at which Harry can cross.

Example

Input:

```
2
29
42
```

Output:

```
1 107
3 111
```

Added by: Paritosh Aggarwal
Date: 2009-02-21
Time limit: 6s
Source limit:50000B
Languages: All
Resource: British Informatics Olympiad, 1999

SPOJ Problem Set (classical)

3923. Philosophers Stone

Problem code: BYTESM2

One of the secret chambers in Hogwarts is full of philosopher's stones. The floor of the chamber is covered by h × w square tiles, where there are h rows of tiles from front (first row) to back (last row) and w columns of tiles from left to right. Each tile has 1 to 100 stones on it. Harry has to grab as many philosopher's stones as possible, subject to the following restrictions:

- He starts by choosing any tile in the first row, and collects the philosopher's stones on that tile. Then, he moves to a tile in the next row, collects the philosopher's stones on the tile, and so on until he reaches the last row.
- When he moves from one tile to a tile in the next row, he can only move to the tile just below it or diagonally to the left or right.

Given the values of h and w, and the number of philosopher's stones on each tile, write a program to compute the maximum possible number of philosopher's stones Harry can grab in one single trip from the first row to the last row.

Input

The first line consists of a single integer T, the number of test cases. In each of the test cases, the first line has two integers. The first integer h (1<=h<=100) is the number of rows of tiles on the floor. The second integer w (1<=w<=100) is the number of columns of tiles on the floor. Next, there are h lines of inputs. The ith line of these, specifies the number of philosopher's stones in each tile of the ith row from the front. Each line has w integers, where each integer m (0<=m<=100) is the number of philosopher's stones on that tile. The integers are separated by a space character.

Output

The output should consist of T lines, (1<=T<=100), one for each test case. Each line consists of a single integer, which is the maximum possible number of philosopher's stones Harry can grab, in one single trip from the first row to the last row for the corresponding test case.

Example

Input:

```
1
6 5
3 1 7 4 2
2 1 3 1 1
1 2 2 1 8
2 2 1 5 3
2 1 4 4 4
5 2 7 5 1
```

Output:

```
32

//7+1+8+5+4+7=32
```

Added by: Paritosh Aggarwal
Date: 2009-02-21
Time limit: 1s-3s
Source limit:50000B
Languages: All

SPOJ Problem Set (classical)

3924. Filchs Dilemna

Problem code: BYTESH1

Filch’s Dilemma (15 points) Argus Filch, the caretaker of Hogwarts, has been given the task to carpet the way to Hogwarts through the grounds. The way is 2 units wide and ‘N’ units long. He has only two types of carpets available, one is 1 unit wide and 2 units long and the other one is L shaped, having 3 square units. Here are their pictures:
> [IMAGE] /www.dcetech.com/events/troika/bytes/images/tiles1.jpg" align="center"/>
> Filch can rotate the carpets when he lays them and has an infinite supply of both types of carpets. As Filch is a squib he cannot magically arrange the carpets and has to resort to logic to find out all possible ways of carpeting the way. He wishes to calculate the number of different ways of carpeting the way.
> For instance, a 2x3 way can be carpeted in 5 different ways as follows:
>
> [IMAGE]
> Notice that both types of carpets can be used simultaneously. Consider, for instance the following way of carpeting a 2x4 way:
>
> [IMAGE] /www.dcetech.com/events/troika/bytes/images/tiles3.jpg" align="center"/>
> Given N, you have to help Filch determine the number of ways to carpet the way of size 2XN. Since this number may be very large, it is sufficient to report the last four digits of the answer. For instance the number of ways to carpet a 2x13 way is 13465. Your program should just print 3465. If the answer is in 4 digits or less it should print the entire answer. For example, if N=3 you should print 5.
>

Input

The first line of the input consists of a single integer T(1<=T<=100). Each of the next T lines consists of a single integer N (1<=N<=1000000), indicating the size of the way.

Output

For each test case, output the last four digits of the number of ways of carpeting the 2xN way. If the answer involves fewer than 4 digits, print the entire number.
> Important Update - If the output of last four digits has leading zeros, print the output without the leading zeros

Example

Input:
```
2
3
13
```

Output:
```
5
3465
```

Added by:	**Paritosh Aggarwal**
Date:	**2009-02-21**
Time limit:	**2s**
Source limit:	**50000B**
Languages:	**All**

SPOJ Problem Set (classical)

3999. FROGGER

Problem code: FROGGER

"Frogger" was one of the first really popular arcade games after it was introduced by SEGA in 1981. The game consists of helping a frog cross a multi-lane motorway without getting run over by a car. You are given a view of an n-lane motorway where each lane consists of m different spaces that can either be empty or be occupied by a car. On each side of the motorway is a curb on which the frog can move freely. In the traffic lanes the frog can only move on the spaces not occupied by cars. The motorway is constructed in such a way that the direction in which the cars travel alternates between the lanes, with cars in the first lane (the one closest to the starting point of the frog) moving to the right. The cars never switch lanes and only move one step forward in each turn. To ensure a steady supply of traffic, a car that reaches the boundary of its lane is reentered at the opposite end of its lane. In one turn of the game all the cars move one step in their assigned direction while the frog can either move one step to the right or to the left, or one step up or down (between lanes or between the curb and the adjoining lane), or it can stand still. Contrary to the cars the frog cannot "wrap-around" i.e. move in one step between the first and last position of a lane or a curb. The frog and the cars move simultaneously. Thus the frog can move to a space given that there will be no car on it in the next round. If the frog is on the same space at the same time as a car it is run over and dies. Note that the frog can jump over an adjacent approaching car in the same lane as itself. Your job is to write a computer program that will calculate the minimum number of turns needed for the frog to get from its starting position on the curb to its final position on the curb on the other side of the road or to determine that this is not possible within a given number of rounds.

Input

First there will be a line containing the number of scenarios you are asked to help the frog in. For each scenario there will first be a line containing a positive integer x <= 10^5 giving the maximum number of rounds that can be used. The next line contains the number of lanes n, 1 <= n <= 20, and the length of each lane m, 1 <= m <= 50. Each of the next n + 2 lines will contain a string of m characters. The character X indicates a car, the character O (letter O) indicates a free space, the character F gives the starting position of the frog, and the character G gives the final destination of the frog. The first line indicates the destination curb, consisting of O's and exactly one G while the last line gives the starting curb consisting of O's and exactly one F, while the intermediate lines each represent one lane of the motorway.

Output

The output will be one line per scenario, either giving the minimum number of turns needed before the frog can get from its starting position to the final position without getting run over by a car or a statement indicating that this was not possible within the maximum number of allowed turns.

Example

Input:

```
2
10
4 4
OOGO
XXOO
XOOX
XXOO
XXOO
OOFO
2
2 2
OG
XX
OO
FO
```

Output:

```
The minimum number of turns is 9.
The problem has no solution.
```

Added by: Fabio Avellaneda
Date: 2009-03-01
Time limit: 60s
Source limit: 50000B
Languages: All
Resource: I maratón interuniversitaria del circuito Redis - Acis. Sedes: Politécnico - Javeriana

SPOJ Problem Set (classical)

4000. GALLUP

Problem code: GALLUP

Often, we see results of gallups, like this: Prefer red: 3.5% Prefer green: 4.5% Prefer yellow: 22.0% Prefer blue: 70.0% and you begin to wonder: how many people did they really ask? If the numbers are simple, like 20%, 40%, and 40%, you know that they asked 5 people (or 10, or 15, or more, but we are interested in the minimum number of people). Your task is to write a program that reads sets of percentages and calculates the smallest number of people that could produce the given percentages. We know that this number is always less than 10 000.

Input

The input is a set of percentages. Each set is on a line of its own. Every line starts with an integer n (0 <= n <= 20) giving the number of percentages in the set. If n > 0, the percentages follow as n numbers; these numbers may have 0-5 decimals, and all percentages in a set have the same number of decimals. (If there are no decimals, there is no decimal point.) The percentages always add up to about 100% as there may be small rounding errors. Numbers are rounded when digits are removed; they are rounded upwards if the first removed digit is 5 or more. Thus, 4.472 is rounded to 4.47, 4.5, or 4, depending on how many digits you want.

Output

For each set of data, print a line starting with "Case i :", where "i" is the data set's number. Then follows a space and an integer giving the computed number of people. If no legal answer in the range 1-9999 exists, print "error" instead of the number.

Example

Input:

```
3 20 40 40
3 33.3 33.3 33.3
2 33 67
1 100.0000
4 3.75 4.25 22.00 70.00
2 49 51
2 50 51
2 49 50
0
```

Output:

```
Case 1: 5
Case 2: 3
Case 3: 3
Case 4: 1
Case 5: 400
Case 6: 35
Case 7: 200
Case 8: error
```

Added by: Fabio Avellaneda
Date: 2009-03-01
Time limit: 60s
Source limit: 50000B
Languages: All
Resource: I maratón interuniversitaria del circuito Redis - Acis. Sedes: Politécnico - Javeriana

SPOJ Problem Set (classical)

4003. Subway planning

Problem code: SUBWAYPL

The government in a foreign country is looking into the possibility of establishing a subway system in its capital. Because of practical reasons, they would like each subway line to start at the central station and then go in a straight line in some angle as far as necessary. You have been hired to investigate whether such an approach is feasible. Given the coordinates of important places in the city as well as the maximum distance these places can be from a subway station (possibly the central station, which is already built), your job is to calculate the minimum number of subway lines needed. You may assume that any number of subway stations can be built along a subway line.
Figure 1: The figure above corresponds to the first data set in the example input.
subway example

Input

The first line in the input file contains an integer N, the number of data sets to follow. Each set starts with two integers, n and d (1 <= n <= 500, 0 <= d < 150). n is the number of important places in the city that must have a subway station nearby, and d is the maximum distance allowed between an important place and a subway station. Then comes n lines, each line containing two integers x and y (-100 <= x, y <= 100), the coordinates of an important place in the capital. The central station will always have coordinates 0, 0. All pairs of coordinates within a data set will be distinct (and none will be 0, 0).

Output

For each data set, output a single integer on a line by itself: the minimum number of subway lines needed to make sure all important places in the city is at a distance of at most d from a subway station.

Example

Input:

```
2
7 1
-1 -4
-3 1
-3 -1
2 3
2 4
2 -2
6 -2
4 0
0 4
-12 18
0 27
-34 51
```

Output:

```
4
2
```

Added by: Fabio Avellaneda
Date: 2009-03-01
Time limit: 60s
Source limit:50000B
Languages: All
Resource: I maratón interuniversitaria del circuito Redis - Acis. Sedes: Politécnico - Javeriana

SPOJ Problem Set (classical)

4004. Exploding CPU

Problem code: CPU

The well known hardware manufacturing company Processors for Professors is about to release a highly specialized CPU with exceptional functionality in, amongst other areas, number theory. It has, for example, an instruction PFACT that takes one parameter and returns all prime factors of that parameter, with an outstanding execution speed. It has, however, one considerable problem. The scientists at the testing lab has just found out that the PFACT instruction for some special input values freaks out and makes the entire processor explode. Even though this could be an amusing effect, it is not the way it was intended to work. The skilled mathematicians have, by trial and error, found that the explosive numbers all share the same interesting number theoretic properties, which might be of help when troubleshooting. An explosive number is a number x = p0p1p2 . . . pn where all pis are distinct prime numbers such that pi = Api-1 +B for i = 1, 2, . . . , n. n >= 3, p0 = 1. A and B are always integers, and might be different for different explosive numbers. For example, the processor will explode when factorizing the number 4505, because 4505 = 1 . 5 . 17 . 53 and 5 = 3 . 1 + 2, 17 = 3 . 5 + 2 and 53 = 3 . 17 + 2 and the numbers 5, 17 and 53 are all prime numbers. In this case A = 3 and B = 2. You are kindly asked to write a computer program that will aid this company in estimating the impact of the errors, by calculating the amount of explosive numbers that exists within a given range of integers.

Input

The input starts with a row containing the number 0 <= N <= 100 of test cases that will follow. For each test case, there will be one row containing two integers, xL and xH separated by a single space. These numbers are such that 0 <= xL <= xH <= 2,000,000,000.

Output

For each test case, output the number of explosive numbers that exist in the range xL <= x <= xH.

Example

Input:

```
2
4505 4505
0 5000
```

Output:

```
1
5
```

Added by: Fabio Avellaneda
Date: 2009-03-01
Time limit: 60s
Source limit: 50000B
Languages: All
Resource: I maratón interuniversitaria del circuito Redis - Acis. Sedes: Politécnico - Javeriana

SPOJ Problem Set (classical)

4033. Phone List

Problem code: PHONELST

Phone List Given a list of phone numbers, determine if it is consistent in the sense that no number is the prefix of another. Let's say the phone catalogue listed these numbers:

* Emergency 911

* Alice 97 625 999

* Bob 91 12 54 26

In this case, it's not possible to call Bob, because the central would direct your call to the emergency line as soon as you had dialled the first three digits of Bob's phone number. So this list would not be consistent.

Input

The first line of input gives a single integer, 1 <= *t* <= 40, the number of test cases. Each test case starts with *n*, the number of phone numbers, on a separate line, 1 <= *n* <= 10000. Then follows *n* lines with one unique phone number on each line. A phone number is a sequence of at most ten digits.

Output

For each test case, output "YES" if the list is consistent, or "NO" otherwise.

Example

Input:

```
2
3
911
97625999
91125426
5
113
12340
123440
12345
98346
```

Output:

```
NO
YES
```

Added by: Andres Galvis
Date: 2009-03-08
Time limit: 1s-3s
Source limit: 50000B
Languages: All
Resource: Nordic Collegiate Programming Contest 2007

SPOJ Problem Set (classical)

4036. Cuckoo Hashing

Problem code: CUCKOO

One of the most fundamental data structure problems is the dictionary problem: given a set *D* of words you want to be able to quickly determine if any given query string *q* is present in the dictionary *D* or not. Hashing is a well-known solution for the problem. The idea is to create a function *h* : S * -> [0..*n* - 1] from all strings to the integer range 0,1,..,*n* - 1, i.e. you describe a fast deterministic program which takes a string as input and outputs an integer between 0 and *n*-1. Next you allocate an empty hash table *T* of size *n* and for each word *w* in *D*, you set *T*[*h*(*w*)] = *w*. Thus, given a query string *q*, you only need to calculate *h*(*q*) and see if *T*[*h*(*q*)] equals *q*, to determine if *q* is in the dictionary. Seems simple enough, but aren't we forgetting something? Of course, what if two words in *D* map to the same location in the table? This phenomenon, called collision, happens fairly often (remember the Birthday paradox: in a class of 24 pupils there is more than 50% chance that two of them share birthday). On average you will only be able to put roughly SQRT *n*-sized dictionaries into the table without getting collisions, quite poor space usage!

A stronger variant is Cuckoo Hashing. The idea is to use two hash functions h_1 and h_2. Thus each string maps to two positions in the table. A query string *q* is now handled as follows: you compute both $h_1(q)$ and $h_2(q)$, and if $T[h_1(q)] = q$, or $T[h_2(q)] = q$, you conclude that *q* is in *D*. The name "Cuckoo Hashing" stems from the process of creating the table. Initially you have an empty table. You iterate over the words *d* in *D*, and insert them one by one. If $T[h_1(d)]$ is free, you set $T[h_1(d)] = d$. Otherwise if $T[h_2(d)]$ is free, you set $T[h_2(d)] = d$. If both are occupied however, just like the cuckoo with other birds' eggs, you evict the word *r* in $T[h_1(d)]$ and set $T[h_1(d)] = d$. Next you put *r* back into the table in its alternative place (and if that entry was already occupied you evict that word and move it to its alternative place, and so on). Of course, we may end up in an infinite loop here, in which case we need to rebuild the table with other choices of hash functions. The good news is that this will not happen with great probability even if *D* contains up to *n*/2 words

Input

On the first line of input is a single positive integer 1 <= *t* <= 50 specifying the number of test cases to follow. Each test case begins with two positive integers 1 <= *m* <= *n* <= 10000 on a line of itself, *m* telling the number of words in the dictionary and *n* the size of the hash table in the test case. Next follow *m* lines of which the *i*:th describes the *i*:th word d_i in the dictionary *D* by two non-negative integers $h_1(d_i)$ and $h_2(d_i)$ less than *n* giving the two hash function values of the word d_i. The two values may be identical.

Output

For each test case there should be exactly one line of output either containing the string "successful hashing" if it is possible to insert all words in the given order into the table, or the string "rehash necessary" if it is impossible.

Example

Input:

```
2
3 3
0 1
1 2
2 0
5 6
2 3
3 1
1 2
5 1
2 5
```

Output:

```
successful hashing
rehash necessary
```

Added by: Andres Galvis
Date: 2009-03-09
Time limit: 1s-3s
Source limit: 50000B
Languages: All
Resource: Nordic Collegiate Programming Contest 2007

SPOJ Problem Set (classical)

4060. A game with probability

Problem code: KPGAME

Alice and Bob play the following game. First, they collect N small stones and put them together in one pile. After that, they throw a coin one by one. Alice starts first. If a player throws heads then he takes exactly one stone from the pile. In case of tails he don't do anything. The one who takes the last stone wins. For each player, his skill of throwing a coin is known (to everyone, including himself and his opponent). More precisely, if Alice wants to throw some specific side of the coin, she always succeeds with probability P. The same probability for Bob is Q. You are to find probability that Alice will win the game if both guys play optimally.

Input

Input starts with a line containing one integer T - a number of test cases (1 <= T <= 50). Then T test cases follow. Each of them is one line with three numbers N, P, and Q separated with a space (1 <= N <= 99999999, 0.5 <= P, Q <= 0.99999999). P and Q have not more than 8 digits after decimal point.

Output

For each test case output one line with a probability that Alice will win the game. Your answer must be precise up to 10^-6.

Example

Input:
```
1
1 0.5 0.5
```

Output:
```
0.666666667
```

Added by: Pavel Kuznetsov
Date: 2009-03-16
Time limit: 1s
Source limit:50000B
Languages: All
Resource: Vologda 2009

SPOJ Problem Set (classical)

4069. Morphing is Fun

Problem code: MORPH

Morphic is a tree that grows very rapidly, bringing happiness to its owner. It has a single trunk consisting of a number of cells stacked one on top of another. Each cell has one of n possible colors which determine the way it mutates during the night, while nobody can see it. Florists denote these colors by the first **n** small letters of the English alphabet and know exactly into how many cells, and of what colors, a cell of each color divides. In fact, they have wrote their knowledge down simply with **n** nonempty words, each word representing the resulting sequence of colors.

A seed of a Morphic has a single cell of color a and is rooted firmly in the ground. As long as the Morphic is still alive, each night all its cells simultaniously morph according to the aforementioned rules, possibly causing an exponential growth because each new cell is of the same size as the original one. For example, if rules say that a becomes ab, and b becomes ca, then after two nights a seed will evolve to a trunk consisting of 4 cells: abca.

Therefore the top of a Morphic is usually hidden in clouds. The only way to tell if it is still alive is to check if visible part of the trunk is changing colors. In order to do so, one can build enormously high (yet still of constant height) tower, and watch from its top a fixed fragment of the trunk.

As you can easily see, it is either sufficient to observe first **k** cells from the bottom for some fixed **k**, or no matter how high the tower is, you will not be able to tell for sure if a Morphic died. The latter happens when for every **k**, rules cause the **k**-th cell to eventually stop changing colors, even though the tree is still alive and mutating.

To prevent waste of money on building such enormous towers, you are to write a program that determines if it is possible to monitor health of a Morphic.

Input

The input contains several Morphics descriptions. The first line contains the number of descriptions **t** (**t** <= 10000) that follow. Each of them begins with the number of colors **n** (1 <= **n** <= 26). Next **n** lines contain the rules by which the Morphic grows. The i-th one describes the sequence of colors in bottom-up order obtained from a single cell of i-th color. Each line contains at most 100 lowercase English letters.

Output

For each test case output one line containing YES if building of a tower is pointless (as in: YES, we can save money!). Otherwise output NO.

Example

Input:

```
4
2
ab
a
3
ba
c
c
3
ba
c
b
3
bbbbbbbbbbbbbbb
ccccccccccccccc
c
```

Output:

```
YES
YES
NO
YES
```

Warning: enormous input/output data, be careful with certain languages

---

Added by: Blue Mary
Date: 2009-03-17
Time limit: 13s
Source limit: 50000B
Languages: All except: C99 strict
Resource: ACM Central European Programming Contest, Wrocław 2008

SPOJ Problem Set (classical)

4070. Two Professors

Problem code: TWOPROF

There are two professors at the great Academy of X that really do not get along with each other. In order not to reveal their names, we will call them 1 and 2. The Academy employs exactly n professors, each of them has to give exactly one lecture. As their schedules are rather tight (they are professors, remember?), the starting and the ending time of each lecture is already fixed. However, it is not yet fixed where each lecture will take place. Obviously, it is impossible to schedule two lectures in the same room if their durations overlap; on the other hand, it is possible if one of them starts exactly at the same time that the other one ends. Your tasks is to find the minimal number of rooms allowing to arrange all the lectures. But know that professors 1 and 2 hate each other so much that they will never give their lectures in the same room.

Input

The input contains several test cases. The first line contains the number of test cases t (t <= 250). Each test begins with a line containing the number of professors n (2 <= n <= 100000). Next n lines follow, i-th of which contains two integers $start_i$ and end_i (0 <= $start_i$ < end_i <= 1000000000), the starting and the ending time of the lecture that the i-th professor gives, respectively.

Output

For each test case output the minimal number of rooms necessary to schedule all the lectures.

Example

Input:

```
4
2
0 10
10 20
3
0 10
10 20
10 20
5
4 14
3 13
2 12
1 11
0 10
4
0 10
10 20
20 30
30 40
```

Output:

```
2
2
5
2
```

Warning: enormous input/output data, be careful with certain languages

Note: The input is too large, so we have 4 input files and the total time limit is 17 seconds.

Added by: Blue Mary
Date: 2009-03-17
Time limit: 17s
Source limit: 50000B
Languages: All except: C99 strict
Resource: ACM Central European Programming Contest, Wrocław 2008

SPOJ Problem Set (classical)

4103. Extend to Palindrome

Problem code: EPALIN

Your task is, given an integer N, to make a palindrome (word that reads the same when you reverse it) of length at least N (1 <= N <= 100,000). Any palindrome will do.

Easy, isn't it? That's what you thought before you passed it on to your inexperienced team-mate. When the contest is almost over, you find out that that problem still isn't solved. The problem with the code is that the strings generated are often not palindromic. There's not enough time to start again from scratch or to debug his messy code.

Seeing that the situation is desperate, you decide to simply write some additional code that takes the output and adds just enough extra characters to it to make it a palindrome and hope for the best. Your solution should take as its input a string and produce the smallest palindrome that can be formed by adding zero or more characters at its end. The input string will consist of only upper and lower case letters.

Example

Input:

```
aaaa
abba
amanaplanacanal
xyz
```

Output:

```
aaaa
abba
amanaplanacanalpanama
xyzyx
```

Note: 1. All palindromes are considered case-sensitive (i.e. 'Aa' is not a palindrome).2. Large I/O. Be careful in certain languages.

Added by: Muntasir Azam Khan
Date: 2009-03-22
Time limit: 3s
Source limit:50000B
Languages: All
Resource: Own problem, used in Next Generation Contest 5

SPOJ Problem Set (classical)

4110. Fast Maximum Flow

Problem code: FASTFLOW

Given a graph with N (2 <= N <= 5,000) vertices numbered 1 to N and M (1 <= M <= 30,000) undirected, weighted edges, compute the maximum flow / minimum cut from vertex 1 to vertex N.

Input

The first line contains the two integers N and M. The next M lines each contain three integers A, B, and C, denoting that there is an edge of capacity C ($1 <= C <= 10^9$) between nodes A and B (1 <= A, B <= N). Note that it is possible for there to be duplicate edges, as well as an edge from a node to itself.

Output

Print a single integer (which may not fit into a 32-bit integer) denoting the maximum flow / minimum cut between 1 and N.

Example

Input:

```
4 6
1 2 3
2 3 4
3 1 2
2 2 5
3 4 3
4 3 3
```

Output:

```
5
```

Viewing the problem as max-flow, we may send 3 units of flow through the path 1 - 2 - 3 - 4 and 2 units of flow through the path 1 - 3 - 4. Viewing the problem as min-cut, we may cut the first and third edges. Either way the total is 5.

Note: see also http://www.spoj.pl/problems/MATCHING/.

Added by: Neal Wu
Date: 2009-03-25
Time limit: 5s
Source limit:50000B
Languages: All

SPOJ Problem Set (classical)

4142. Ellipse

Problem code: ELLIPSE

Given 5 points on a ellipse, calculate the area of the ellipse. We accept solutions with absolute error less than 10^{-6}.

Input

Many test cases. Each contains a line with 10 integers with absolute value less than 1000 - the X and Y coordinates of the 5 points, respectively.

Input terminates by EOF. Note that there can be extra spaces in a single line.

Output

Each line contains a single float-point number - the area of the corresponding ellipse, or "IMPOSSIBLE" if the ellipse doesn't exist or can't be unique determined.

Example

Input:
```
6 1 3 2 -2 -3 -3 -2 1 6
7 -3 2 7 6 3 5 5 -2 -9
```

Output:
```
IMPOSSIBLE
157.079633
```

Note: You can click on "Wrong Answer" to get further information.

Added by: Blue Mary
Date: 2009-03-27
Time limit: 10s
Source limit:50000B
Languages: All except: C99 strict
Resource: Classical Problem, description by Blue Mary

SPOJ Problem Set (classical)

4157. Domino

Problem code: DOMINO2

You have an **n**x**m** rectangle, some cells have some obstacles in. A domino piece is a 1x2 or 2x1 rectangle. You're going to place some domino pieces in this rectangle so that there's no empty cell is covered more than once and no cell with obstacles is covered. For some unknown reason, you have to ensure there's at least one piece covering some cell in row i and some cell in row i+1 at the same time for all i in 1..**n**-1. Similarly there's at least one piece covering some cell in column i and column i+1 for all i in 1..**m**-1. Your task is to count the number of different valid domino covering.

Input

The first line of the input contains two integer numbers **n**, **m** (1<=**n**,**m**<=15).

The following **n** lines describe the rectangle. Each line contains **m** characters. The j-th character of line i+1 may be either a 'x'(ASCII code 120), representing obstacles in cell (i, j), or a '.'(ASCII code 46), representing an empty cell.

Output

One number, representing the number of different valid domino placing.

Since the number could be quite large, output the answer modulo 19901013.

Example

Input:
```
3 3
...
...
...
```

Output:
```
2
```

Note

The 2 different placings are

```
112     411
4.2     4.2
433     332
```

Added by: Jin Bin
Date: 2009-03-30
Time limit: 15s
Source limit: 10000B
Languages: All except: C99 strict
Resource: Zhejiang TSC for Chinese National OI 2009

SPOJ Problem Set (first)

4164. A conjecture of Paul Erdős

Problem code: HS08PAUL

In number theory there is a very deep unsolved conjecture of the Hungarian Paul Erdős (1913-1996), that there exist infinitely many primes of the form x^2+1, where x is an integer. However, a weaker form of this conjecture has been proved: there are infinitely many primes of the form x^2+y^4. You don't need to prove this, it is only your task to find the number of (positive) primes not larger than n which are of the form x^2+y^4 (where x and y are integers).

Input

An integer T, denoting the number of testcases (T<=10000). Each of the T following lines contains a positive integer n, where n<10000000.

Output

Output the answer for each n.

Example

Input:
```
4
1
2
10
9999999
```

Output:
```
0
1
2
13175
```

Added by: Robert Gerbicz
Date: 2009-04-05
Time limit: 5s
Source limit: 4096B
Languages: All
Resource: High School Programming League 2008/09

SPOJ Problem Set (classical)

4166. Four colors

Problem code: HS08FOUR

Let there be given n points: $P_1, P_2 ..., P_n$ arranged in this order on a line. We would like to color them using four colors: white, black, red, and blue, in such a way that for every three consecutive points it is true that either
1. the colors of these three points are pairwise distinct, or
2. the color of some point is white.

Input

An integer T, denoting the number of testcases (T<100000). In each line you are given one positive integer (n<1000000000). There are 5 input sets.

Output

Find the number of possible colorings of the n points. Since the answer can be very big, output only the answer modulo 1000000007.

Example

Input:41231000**Output:**41643283570349

Warning: large input/output data, be careful with certain languages

Warning: A naive algorithm will probably solve only the first input set.

Added by: Robert Gerbicz
Date: 2009-04-05
Time limit: 1s
Source limit:4096B
Languages: All
Resource: High School Programming League 2008/09

SPOJ Problem Set (classical)

4168. Square-free integers

Problem code: SQFREE

In number theory we call an integer square-free if it is not divisible by a perfect square, except 1. You have to count them!

Input

First line contains an integer *T*, the number of test cases (*T*<=100). The following T lines each contains one positive integer: *n*, where n <= 10^{14}

Output

T lines, on each line output the number of (positive) square-free integers not larger than n.

Example

Input:
```
3
1
1000
100000000000000
```

Output:
```
1
608
60792710185947
```

Warning: A naive algorithm probably not works.

Added by: Robert Gerbicz
Date: 2009-04-06
Time limit: 20s
Source limit: 2009B
Languages: All
Resource: classic, own input

SPOJ Problem Set (classical)

4172. Multiplicative digital root

Problem code: DROOT

For an integer find the multiplicative digital root of it! Multiple all nonzero digits of that number and repeat this process until it is only a single digit. We call that digit the multiplicative digital root of the number. For example the multiplicative digital root of n=2009 is 8, because the first iteration is: 2*9=18, the second is 1*8=8, and we stop here.

Input

The first line of the input file contains one integer T, the number of test cases. The following T lines each contains a big positive integer: n, where $n<10^{10000}$

Output

Output the mulplicative digital root for each n.

Example

Input:

```
4
6
2009
555555555
847938630482747410708417738635300464477112059683336648877683
```

Output:

```
6
8
5
2
```

Warning: large input data, be careful with certain languages
Warning: not every languages are available for this task

Added by: Robert Gerbicz
Date: 2009-04-06
Time limit: 2s
Source limit: 2048B
Languages: C C++ PAS gpc PAS fpc
Resource: classic, own input

SPOJ Problem Set (classical)

4176. A Knightly Pursuit

Problem code: KPURSUIT

In chess, game pieces move about an chessboard in a fashion defined by their type. The object of the game is to capture opposing pieces by landing on their squares, and eventually trapping the king piece. In our version of the game, we shall use a variable sized board with only 2 pieces on it: A white pawn which moves relentlessly towards the top row of the chessboard one square at a time per move; and a black knight which can move from its current location in any of up to eight ways: two squares up or down and one square left or right, or one square up or down and two squares left or right. The knight must remain on the board at all times; any move that would take it off the board is therefore disallowed. In the diagram below, the knight's position is labelled K and its possible moves are labelled 1 to 8.

. . . . . .
. . 8 . 1 . .
. 7 . . . 2 .
. . . K . . .
. 6 . . . 3 .
. . 5 . 4 . .
. . . . . .

The pawn moves first; then the knight and pawn alternate moves. The knight tries to land either on the square occupied by the pawn (a win) or on the square immediately above the pawn (a stalemate). If the pawn reaches the top row of the board the game ends immediately and the knight loses (a loss).

Input

The first line of input contains a positive integer, n, the number of games to analyze. For each game there are six lines on input:
r, the number of rows in the chessboard.
c, the number of columns in the chessboard.
pr, the row of the starting position of the pawn.
pc, the column of the starting position of the pawn.
kr, the row of the starting position of the knight.
kc, the column of the starting position of the knight.
All numbers in the input don't exceed 100. (Thanks to Blue Mary for pointing that out).
The pawn and the knight will have different starting positions. Row 1 is at the bottom of the board and Row r is at the top of the board. Column 1 is at the left and column c is at the right.

Output

If the knight can win and, output the minimum number of moves it must make to do so. If the knight cannot win, your program should determine if it can cause a stalemate and, if it can, the minimum number of moves it must make to do so. Finally if the knight cannot win or cause a stalemate, your program should compute the number of moves the knight makes before the pawn wins.

Example

Input:

```
3
99
99
33
33
33
35
3
3
1
1
2
3
99
99
96
23
99
1
```

Output:

```
Win in 1 knight move(s).
Stalemate in 1 knight move(s).
Loss in 2 knight move(s).
```

Added by: Analysis Mode (Elspeth, Knight-Errant)
Date: 2009-04-07
Time limit: 1s
Source limit: 50000B
Languages: All except: C99 strict
Resource: Canadian Computing Competition 1999 Senior Question 4

SPOJ Problem Set (classical)

4177. Herding

Problem code: HERDING

Oh no! A number of stray cats have been let loose in the city, and as the City Cat Catcher, you have been assigned the vital task of retrieving all of the cats. This is an ideal opportunity to test your latest invention, a cat trap which is guaranteed to retrieve every cat which walks into a square-shaped subsection of the city.
Fortunately, you have the assistance of one of the world's foremost cat psychologists, who has the amazing ability of predicting, given a square subsection of the city, exactly which of the four cardinal directions (north, east, south or west) the cat will head. While this information is handy, you still don't know where all the cats currently are.
In order to prove the cost-effectiveness of your method to the City it would, of course, be important to minimize the number of traps used.

Input

The input will begin with a line consisting of two numbers n and m, separated by a space (1 <= n, m <= 1000). The city will be an n x m grid of square subsections. The next n lines will each consist of a string of length m, consisting of the letters 'N', 'E', 'S', or 'W', representing north, east, south and west, respectively. (The first character of the first line will be the northwesternmost point.) The direction in the square is the direction which cats will head if they are in that square. The cat psychologist assures you that cats have no interest in leaving the city.

Output

Output the minimum number of traps needed.

Example

Input:
```
3 4
SWWW
SEWN
EEEN
```

Output:
```
2
```

Added by:	Analysis Mode (Elspeth, Knight-Errant)
Date:	2009-04-07
Time limit:	1s-3s
Source limit:	50000B
Languages:	All except: C99 strict
Resource:	Canadian Computing Competition 2008 Stage 2 Day 2 Problem D

SPOJ Problem Set (classical)

4178. Distance on a square lattice

Problem code: LATTICE

Let L to be an *nXn* square lattice, you can consider its points as *(x,y)*, where *x* and *y* are integers from the [1,*n*] interval. And let f(*n*) to be the expected distance between two not neccesserily distinct points on the lattice. For example f(1)=0 and f(2)=(2 + SQRT 2)/4.

Input

There is no input.

Output

5000 lines, on the n-th line give the value of f(*n*) by 2 digits after the decimal point.

Example

Input:
```
No input.
```

Output:
```
0.00
0.85
1.45
2.01
2.55
.
.
.
2607.03
```

Added by: Robert Gerbicz
Date: 2009-04-07
Time limit: 1s
Source limit:50000B
Languages: TEXT
Resource: own resource

SPOJ Problem Set (classical)

4179. Temptation Island

Problem code: TEMPTISL

On Monday, the number of frosh were reduced in half. To further reduce the number of engineers to a manageable number, the following challenge was devised for the second day. Each of the students would have to take this challenge individually.
Each student would be placed at a vertex of perimeter fence of Waterloo (oh yeah, some background: to keep UofT's engineering Lady Godiva band out of Waterloo, a fence was erected surrounding the university. The fence just happens to be an N-gon). At some other vertex along the fence would be located a temptation so seductive that no Waterloo student could resist - an extra-credit assignment. The challenge of each student is to go from his starting vertex to the vertex with the prize. There are however 3 rules:
a) The student can only travel from vertex to vertex (backwards or forwards) along the polygonal fence.
b) The student has to make contact with exactly K vertices (the vertex he starts at doesn't count unless he returns to it). The K vertices need not be unique. The final vertex has to be the one with the prize.
c) If the student cannot reach the prize and make contact with exactly K vertices, he fails the test and is kicked out of the university.
Of course, no Waterloo student is satisfied with only 1 solution to any problem. Therefore, inevitably, each student determines all ways that he/she can win. Note that there may be no solution to the problem (the astute student has figured out that this will result in a class size of 0 - this is entirely allowable as the variable used to quantify enrollment was incorrectly defined as a whole number instead of a natural number).

Input

N K (N, K <= 50)
A B (A = the starting vertex number, B = destination vertex number)
-1 -1 terminates input

Output

The total number of ways of reaching the destination from the starting point by following the above rules. The total number of ways will be less than 2^{63} - 1. Output 0 if there are no solution.

Example

Input:
```
8 5
1 4
-1 -1
```

Output:
```
6
```

Added by:	Analysis Mode (Elspeth, Knight-Errant)
Date:	2009-04-08
Time limit:	0.5s-1s
Source limit:	50000B
Languages:	All except: C99 strict
Resource:	Woburn Challenge 2001

SPOJ Problem Set (classical)

4182. Candy (Again)

Problem code: FCANDY

You and a friend have a big bag of candy. You want to keep slim and trim, and so you would like to equalize the candy which you are sharing with your friend in terms of calorie count. That is, your task is to divide the candies into two groups such that the number of calories in each group is as close together as possible.

Input

The first line of input contains the number of different kinds of candy you have in your bag of candy N (1 <= N <= 100). On the following N lines, there are pairs of numbers describing each type of candy. The candy description is of the form k_i c_i where k_i is the number of that particular type of candy contained in the bag and ci is the calorie count for each piece of that type of candy. You may assume that 1 <= k_i <= 500 and 1 <= c_i <= 200.

Output

Your output is one integer which is the minimum difference of calories between friends

Example

Input:

```
4
3 5
3 3
1 2
3 100
```

Output:

```
74
```

Added by: Analysis Mode (Elspeth, Knight-Errant)
Date: 2009-04-08
Time limit: 0.5s-9s
Source limit:50000B
Languages: All except: C99 strict
Resource: Canadian Computing Competition 2008 Stage 2 Question E

SPOJ Problem Set (classical)

4185. Cube

Problem code: CCCCUBE

Imagine a cube formed from solid interlocking pieces of various shapes. If the pieces are sufficiently intertwined, the only way to separate them would be to cut some of them. We can ask the question, "is the cube stable?" That is, is it physically impossible to separate the cube into 2 or more fragments without deforming and cutting any individual piece?

Your program must answer this question for a variety of such cubes.
The pieces that make up a cube will be specified as follows: divide the cube into a grid of *n*n*n* miniature cubes, each labelled by a capital letter. Two adjacent (face-sharing) are joined together if and only if they are labelled by the same letter. For instance, the first example cube given consists of 3 solid pieces.

Input

Your program will be given the specification of up to 10 different cubes. The first two lines of each specification will consist of the size of that cube, *n* (1 <= n <= 10), and a blank line. There will be no spaces in the input. The input will be terminated by a number 0 on a line by itself.

Output

For each cube given, in the order specified, print "Yes" if that cube is stable, and "No" if it is not.

Example

Input:

```
2

AB
AB

BB
BA

3

AAA
BBB
AAA

AAA
ABA
AAA

ABA
ABA
ABA

0
```

Output:

```
No
Yes
```

Added by:	Brian
Date:	2009-04-08
Time limit:	3s
Source limit:	50000B
Languages:	All
Resource:	2003 Canadian Computing Competition Stage 2 - Day 1, Question 3

SPOJ Problem Set (classical)

4186. Break a New RSA system

Problem code: HS08CODE

Today, Gerrob's RSA company has featured a New RSA cryptosystem: its public key is n, the secret keys are three distinct primes p, q and r, where $n=p*q*r$. Note that the ordinary RSA uses only 2 primes! Unfortunately some hackers have stolen a DVD from the company. It does not store the secret keys, only some information about the system, namely, the values of:
ph (n) - Euler's totient function and
s (n) - the sum of the divisors.
Obviously you know also n, because that's public.

Now, Gerrob's RSA employees are trying to determine if hackers will be able to break the system. Could you help them to answer this question?

Input

The first line contains a single integer T, the number of test cases, where $T<=$ 20000. The following T lines each contains three numbers n, ph (n) and s (n) in this order. There are 5 input sets.

Output

Output T lines, the values of p, q and r in increasing order. It is guaranteed that p, q, $r<10^6$.

Example

Input:
```
4
30 8 72
61321 54912 68040
451464315257 451286179344 451642497600
91896729624994213 91896040105364880 91897419147616160
```
Output:
```
2 3 5
13 53 89
6397 8039 8779
231859 574261 690187
```

Warning: large input/output data, be careful with certain languages

Warning: A naive algorithm will probably solve only the first input set.

Added by:	Robert Gerbicz
Date:	2009-04-08
Time limit:	0.400s-1s
Source limit:	50000B
Languages:	All
Resource:	High School Programming League 2008/09

SPOJ Problem Set (classical)

4188. Amazing equality

Problem code: HS08EQ

The definition of a perfect number is about 2300 years old. A perfect number is defined as a positive integer which is the sum of its proper positive divisors, that is, the sum of the positive divisors excluding the number itself. What can we get if, in the sum, we replace each divisor by its square? You can prove that there is no such number. But there are many numbers for which the sum of some divisors' squares is equal to *n*, so $n=d_1^2+d_2^2+...+d_k^2$, where $d_1, d_2, ...,d_k$ are distinct (positive) divisors of *n*. You have to count how many times this happens. For example: the divisors of *n*=120 are 1, 2, 3, 4, 5, 6, 8, 10, 12, 15, 20, 24, 30, 40, 60, 120. And there are exactly two amazing equalities:
$120=2^2+4^2+10^2$
$120=2^2+4^2+6^2+8^2$

Input

The first number is *T*, denoting the number of test cases ($T<1000$). *T* lines follow, each of which contains one positive integer ($n<10^{10}$).

Output

Output *T* lines, the answer for each *n*.

Example

Input:
```
6
120
720
1000
1200
92070
123618780
```

Output:
```
2
13
0
10
6448
292
```

Added by: Robert Gerbicz
Date: 2009-04-09
Time limit: 12s
Source limit: 4096B
Languages: All
Resource: High School Programming League 2008/09

SPOJ Problem Set (classical)

4189. Landing

Problem code: LANDING

Keep watching the skies! Alien spacecraft are due to land any day now to share all of their advanced programming secrets with us.
In preparation for this day, you've been asked with preparing a landing pad for our visitors in a given field. Unfortunately, due to enviromental considerations, you will not be permitted to remove any of the trees which currently exist on the field. These trees are of immense scientific research, since they have zero radius and only grow at points with integer co-ordinates. However, this could be a blessing in disguise. For security reasons, the landing pad must be in contact with atleast three trees. Security cameras will be placed at the tops of these trees.
Alien spacecraft are perfectly circular craft of various sizes, so the landing pad will also be circular. Since it would be polite to warn potential visitors ahead of time if their spacecraft is too large for our landing pad, you must now determine the size of largest circular region that we can place on the field which contacts at least three trees, but does not contain any trees within.

Input

The first line of input consists of the number n of trees (3 <= n <= 100 000). the next n lines will each consist of a pair of integers x and y (-10000 <= x, y <= 10000), separated by a space, giving the co-ordinates of a tree. You may assume that no two trees are at the same co-ordinates.

Output

Output the radius of the largest possible landing pad. If the correct answer is R, you should output number a such that

[IMAGE]

The above calculation is used to define an acceptable range or tolerance for the answer you find. You may also assime that $r < 10^9$. You may assume there exists at least one landing pad.

Example

Input:
```
4
1 1
1 -1
-1 -1
-1 1
```

Output:
```
1.41421356
```

[IMAGE]

Added by:	Analysis Mode (Elspeth, Knight-Errant)
Date:	2009-04-09
Time limit:	5s-15s
Source limit:	50000B
Languages:	All except: C99 strict
Resource:	Canadian Computing Competition 2008 Stage 2 Day 2 Problem E

SPOJ Problem Set (classical)

4197. Dominoes

Problem code: DOMINOES

Johnny is playing with some dominoes one afternoon. His dominoes come in a variety of heights and colors.
Just like any other child, he likes to put them in a row and knock them over.
He wants to know something: how many pushes does it take to knock down all the dominoes?
Johnny is lazy, so he wants to minimize the number of pushes he takes.
A domino, once knocked over, will knock over any domino that it touches on the way down.
For the sake of simplicity, imagine the floor as a one-dimensional line, where 1 is the leftmost point.
Dominoes will not slip along the floor once toppled. Also, dominoes do have some width: a domino of length 1 at position 1 can knock over a domino at position 2.
For the mathematically minded:
A domino at position x with height h, once knocked over to the right, will knock all dominoes at positions x+1, x+2, ..., x+h rightward as well.
Similarly, the same domino knocked over to the left will knock all dominoes at positions x-1, x-2, ..., x-h leftward.

Input

The input starts with a single integer N (N <= 100000), the number of dominoes, followed by N pairs of integers.
Each pair of integers represents the location and height of a domino, in that order (0 <= location <= 10^9, 0 <= height <= 10^9).
No two dominoes will have the same location.

Output

A single integer on a single line: the minimum number of pushes Johnny must make in order to ensure that all dominoes are knocked over.

Example

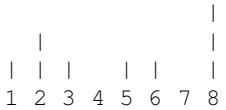
Input:

```
6
1 1
2 2
3 1
5 1
6 1
8 3
```

Output:

```
2
```

Explanation



Pushing 1 causes 2 and 3 to fall, while pushing 8 causes 6 to fall and gently makes 5 tip over as well.

Added by: Brian
Date: 2009-04-10
Time limit: 1s-2s
Source limit: 50000B
Languages: All
Resource: Hanson Wang

SPOJ Problem Set (classical)

4198. Lego

Problem code: LEGO

It's Christmas morning, and you've got what you wanted: a box of Lego(TM)! (Okay, maybe not, but better than nothing)
Lego is pretty fun to tinker with, and you've decided to build some sort of shape.
(For the sake of this problem, let's say your shape is basically 2-dimensional - it'll be a slab)
But once you pick it up, you discover that you didn't plan it properly, and your wonderful shape just falls apart.
Now, you're planning to build something big, and so you're going to use the computer to help you.
Write a program, that given the layout of a Lego design, outputs the number of pieces it would break into if picked up.
(Assume that the bricks bind together perfectly)
The Legos will be built on a x-y coordinate plane, with (0,0) being the bottom left corner.
The blocks are flat on your carpet, so a block will never 'fall down'.
(If you haven't seen a Lego brick before: A Lego brick has grooves on its top that match with notches on the bottom.
If a groove and a notch bind, the bricks will stay together. See the diagram.
A brick will bind with another brick securely even if just a single notch touches another groove.

Input:

The first line contains N (the number of Lego pieces), $1 <= N <= 100000$.
N lines follow, each with 4 integers x_1, y_1, x_2, y_2 ($0 <= x_1 < x_2 <= 2\times10^9, 0 <= y_1 < y_2 <= 2\times10^9$)
This means that there is a brick with bottom left corner (x_1,y_1) and top right corner (x_2,y_2). x denotes the horizontal coordinate and y the vertical coordinate. Two bricks will bind if one's bottom y-coordinate coincides with the other's top y-coordinate and the union of the two intervals (the bottom of one and the top of the other) has nonzero length.
No bricks will overlap.

Output:

A single line containing the number of separate pieces that these blocks form.

Sample Input:

```
4
0 0 2 2
1 2 3 4
2 0 4 2
4 0 6 2
```

Sample Output:

```
2
```

Explanation

Blocks #1,2,3 are joined securely.
However, #4 is just hanging around.
[IMAGE]

Added by:	Brian
Date:	2009-04-10
Time limit:	1s-3s
Source limit:	50000B
Languages:	All
Resource:	Hanson Wang

SPOJ Problem Set (classical)

4200. Hamster flight

Problem code: HAMSTER1

There is a competition of flying hamsters in Hamsterburg. Each competing hamster is thrown from a sling. The judges rate the flight according to its length and height. Let X meters be the distance of the flight, and Y meters - maximum height to which the hamster rose during the flight. The hamster will receive K1*X + K2*Y points for such a flight. The initial speed of the hamsters is V0 m/s. Free fall acceleration is g = 10 m/s^2. There is no air friction. The size of the hamster and the sling are negligible. When the hamster is thrown from the sling its height is 0 meters. You should determine the angle at which the hamster must be thrown so that he receives maximum points.

Input

The first line of input contains number t - the amount of tests. Then t tests follow one per line. The description of each test consists of three integers separated by single spaces. The first integer is V0, the second - K1, the third - K2.

Constraints

1 <= t <= 10000
1 <= V0 <= 100
0 <= K1, K2 <= 1000
0 < K1 + K2

Output

For each test output the angle in radians at which the hamster must be thrown, and the amount of points it will receive. The numbers should be separated with spaces. Print the numbers with exactly three digits in the fractional part.

Example

Input:
```
3
10 10 0
10 0 10
10 10 10
```

Output:
```
0.785 100.000
1.571 50.000
0.908 128.078
```

Added by:	Spooky
Date:	2009-04-10
Time limit:	2s
Source limit:	50000B
Languages:	All
Resource:	Advancement Spring 2009, http://sevolymp.uuuq.com/

SPOJ Problem Set (classical)

4201. Coder Ratings

Problem code: RATING

Some of the more elite (and not-so-elite) coders around take part in a certain unnamed programming contest. In said contest, there are multiple types of competitions. Here, we consider the Open and High School competition types. For each type, each competitor receives a *rating*, an integer between 1 and 100000, inclusive. A coder's rating is based upon his or her level of performance in matches and is calculated using a complicated formula which, thankfully, you will not be asked to implement.
Although the Open and High School ratings for a coder who has participated in both competition types lately are usually close, this is not always the case. In particular, High School matches are more about speed, since many coders are able to solve all the problems, whereas Open matches require more thinking and there is a steeper curve in terms of problem difficulty.

Problem Statement

You are given N coders ($1 <= N <= 300000$), conveniently numbered from 1 to N. Each of these coders participates in both High School and Open matches. For each coder, you are also given an Open rating A_i and a High School rating H_i. Coder i is said to be *better* than coder j if and only if both of coder i's ratings are greater than or equal to coder j's corresponding ratings, with at least one being greater. For each coder i, determine how many coders coder i is better than.

Input Format

On the first line of input is a single integer N, as described above.
N lines then follow. Line i+1 contains two space-separated integers, A_i and H_i.

Output Format

Line i should contain the number of coders that coder i is better than.

Sample Input

```
8
1798 1832
862 700
1075 1089
1568 1557
2575 1984
1033 950
1656 1649
1014 1473
```

Sample Output

```
6
0
2
4
7
1
5
1
```

Added by:	Brian
Date:	2009-04-10
Time limit:	1s-10s
Source limit:	50000B
Languages:	All
Resource:	own problem

SPOJ Problem Set (classical)

4202. Brackets Parade

Problem code: BRPAR

Count the number of different correct bracket sequences consisting of k1 pairs of brackets of the 1st type, k2 pairs of brackets of the 2nd type, ..., km pairs of brackets of the m-th type. The bracket sequence is considered correct in the following cases:

- empty sequence is correct;
- if A is correct and B is correct then AB is correct;
- if A is correct then $(_i A)_i$ is correct where $(_i$ and $)_i$ are opening and closing brackets of the same type.

Input

The first line of input is the number 0 < n <= 1000 of test cases. Each of the following n lines describe a test case. Each line starts with number 0 < m <= 100 the amount of different bracket types. Then m positive numbers k1, k2, ..., km follow each separated with a space. Number ki is the amount of pairs of brackets of i-th type. The total amount of pairs of brackets is not greater than 1000.

Output

For each test case output a line containing single integer - the answer to the problem modulo 1000000007.

Example

Input:
```
3
1 4
2 2 2
3 1 2 3
```

Output:
```
14
84
7920
```

Added by: Spooky
Date: 2009-04-11
Time limit: 1s
Source limit:50000B
Languages: All
Resource: Open All-Ukrainian Collegiate Contest Semi-Final, 2009

SPOJ Problem Set (classical)

4206. Fast Maximum Matching

Problem code: MATCHING

FJ has N (1 <= N <= 50,000) cows and M (1 <= M <= 50,000) bulls. Given a list of P (1 <= P <= 150,000) potential matches between a cow and a bull, compute the greatest number of pairs that can be matched. Of course, a cow can be matched to at most one bull, and vice versa.

Input

The first line contains three integers, N, M, and P. Each of the next P lines contains two integers A (1 <= A <= N) and B (1 <= B <= M), denoting that cow A can be matched with bull B.

Output

Print a single integer that is the maximum number of pairs that can be obtained.

Example

Input:
```
5 4 6
5 2
1 2
4 3
3 1
2 2
4 4
```

Output:
```
3
```

Cow 1 can be matched to bull 2, cow 3 to bull 1, and cow 4 to bull 3.

Note: see also http://www.spoj.pl/problems/FASTFLOW/.

Added by: Neal Wu
Date: 2009-04-12
Time limit: 3s
Source limit:50000B
Languages: All

SPOJ Problem Set (classical)

4235. Wandering Queen

Problem code: QUEEN

There is a checkmates board with **n** rows and **m** columns. Some of the cells of the board are occupied. There is a queen standing on a certain cell. It wants to move to another cell of this board. Help it do this making the least possible moves. The queen can go any number of cells in any of eight directions in a single move, but it can't pass through or stand on the occupied cells and leave the board.

Input

The first line of the input contains number **t** - the amount of tests. Then **t** test descriptions follow. The first line of each test consists of two numbers **n** and **m** separated with a space. Then **n** lines follow each containing **m** characters describing the board. Character '.' means a free cell, character 'X' - an occupied cell, character 'S' - the starting cell of the queen, character 'F' - the cell where the queen wants to go. It is guaranteed that there will be exactly one character 'S' and one character 'F' on each board.

Constraints

1 <= **t** <= 30
2 <= **n**, **m** <= 1000

Output

For each test case print the minimum number of moves the queen has to do to reach the desired cell. Print '-1' if the queen can't reach the cell.

Example

Input:
```
3
3 3
S..
...
..F
3 3
S..
XX.
F..
3 3
S..
XXX
..F
```

Output:
```
1
3
-1
```

Added by:	Spooky
Date:	2009-04-16
Time limit:	5s
Source limit:	50000B
Languages:	All
Resource:	Advancement Spring 2009, http://sevolymp.uuuq.com/

SPOJ Problem Set (classical)

4273. Train TimeTable

Problem code: TTTABLE

A train line has two stations on it, A and B. Trains can take trips from A to B or from B to A multiple times during a day. When a train arrives at B from A (or arrives at A from B), it needs a certain amount of time before it is ready to take the return journey - this is the turnaround time. For example, if a train arrives at 12:00 and the turnaround time is 0 minutes, it can leave immediately, at 12:00.
A train timetable specifies departure and arrival time of all trips between A and B. The train company needs to know how many trains have to start the day at A and B in order to make the timetable work: whenever a train is supposed to leave A or B, there must actually be one there ready to go. There are passing sections on the track, so trains don't necessarily arrive in the same order that they leave. Trains may not travel on trips that do not appear on the schedule.

Input

The first line of input gives the number of cases, **N(1<=N<=100)**. **N** test cases follow.
Each case contains a number of lines. The first line is the turnaround time, **T(0<=T<=60)**, in minutes. The next line has two numbers on it, **NA** and **NB**. **NA** is the number of trips from A to B, and **NB** is the number of trips from B to A**(0<=NA,NB<=100)**. Then there are **NA** lines giving the details of the trips from A to B.
Each line contains two fields, giving the HH:MM departure and arrival time for that trip. The departure time for each trip will be earlier than the arrival time. All arrivals and departures occur on the same day. The trips may appear in any order - they are not necessarily sorted by time. The hour and minute values are both two digits, zero-padded, and are on a 24-hour clock (00:00 through 23:59).
After these **NA** lines, there are **NB** lines giving the departure and arrival times for the trips from B to A.

Output

For each test case, output one line containing "Case #x: " followed by the number of trains that must start at A and the number of trains that must start at B.

Example

Input:

```
2
5
3 2
09:00 12:00
10:00 13:00
11:00 12:30
12:02 15:00
09:00 10:30
2
2 0
09:00 09:01
12:00 12:02
```

Output:

```
Case #1: 2 2
Case #2: 2 0
```

Added by: abhijith reddy d
Date: 2009-04-27
Time limit: 1s
Source limit:50000B
Languages: All
Resource: Google Codejam 2008

SPOJ Problem Set (classical)

4305. Drilling

Problem code: AE3A

<pByteman is the person in charge of a team that is looking for crude oil reservoirs. He has made two boreholes: he found crude oil in point A and found out that there is no crude oil in point B. It is known that the oil reservoir occupies a connected fragment of segment AB with one end at point A. Now Byteman has to check, how far, along the segment connecting points A and B, does the oil reservoir reach. It is not that simple, however, because in some locations one can drill faster than in other locations. Moreover, Byteman's team is rather small, so they can drill in at most one location at a time. Byteman's boss would like him to predetermine when he will be able to identify the boundary of the oil reservoir.

Byteman has asked you for help. He has divided the segment connecting points A and B into n+1 segments of equal length. If we assume that point A has coordinate 0, and point B coordinate n + 1, then there are n points with coordinates 1, 2, . . . , n between them. It is enough to find the farthest from A of these points in which some crude oil occurs. Byteman has informed you about the amounts of time necessary for making boreholes in these points -- they are equal to $t_1, t_2, \ldots, t_n$ respectively. You should create such a plan of drilling, that the time necessary to identify the oil reservoir's boundary is shortest possible, assuming the worst-case scenario.

Input

The first line of the standard input contains a single positive integer n (1 <= n <= 2000). The second line contains n positive integers t1, t2, . . . , tn separated by single spaces ($1 <= t_i <= 10^6$).

Output

Your program should write a single integer to the standard output--the smallest amount of time that Byteman has to spend (assuming the worst-case scenario) drilling in search of oil, to be sure that he will identify the reservoir's boundary.

Example

For the input data:

```
4
8 24 12 6
```

the correct result is:

```
42
```

Explanation of the example.

Assume that Byteman makes the first borehole at point 1, what takes him time 8. It can then turn out that he finds crude oil there and he will have to check, how far to the right does the reservoir reach. He will need two more boreholes, making which requires 36 units of time in the worst case. Therefore, in this case Byteman will spend in total 44 units of time drilling.

It turns out that it is better to start at point 2. If there is no crude oil there, it is sufficient to check point 1. However, in the worst case Byteman will have to make two more boreholes in points 3 and 4 and end his work in total time equal to 42.

Added by: Race with time
Date: 2009-05-03
Time limit: 1s
Source limit:50000B
Languages: All
Resource: Algorithmic Engagements 2009

SPOJ Problem Set (classical)

4323. Voting Districts

Problem code: VOTE

The land of Yu consists of N cities, labeled 1,2,...N connected by N-1 roads in such a way that there exists a path between any two cities.

For the first time in history, Yu is holding free elections. But they need to divide their new republic into voting districts.

The division of Yu into voting districts must satisfy the following:

- Each city belong to exactly one district.
- Districts must have an equal number of cities.
- Each district must be entirely self-connected (i.e. for any two cities in the same district, there exists a path between them passing only including cities of that district).

For what numbers of voting districts can this work?

Input

The first line of input is the number of test cases (less than 100) to follow.

Each test case is preceded by a blank line and begins with N (less than 10^5), the number of cities. N lines follow.

The first integer of the ith line (indexing starting at 1) is K_i, the number of cities directly connected to City i. The next K_i integers are the cities directly connected to City i.

Output

For each test case print a line of all numbers of districts into which Yu can be divided, from least to greatest, separated by spaces.

Example

Input:
```
3

4
1 2
2 1 3 4
1 2
1 2

11
2 2 3
3 1 4 5
1 2
1 2
2 6 7
1 5
1 5
2 9 10
1 8
1 8
1 10
```
Output:
```
1 2 4
1 11
```

Note: Notice that Yu can always be divided into 1 district (of N cities), or N districts (of 1 city each).

Added by: Paul Draper
Date: 2009-05-05
Time limit: 0.5s-8s
Source limit: 50000B
Languages: All

SPOJ Problem Set (classical)

4324. The fate of the pineapple

Problem code: EVERLAST

In order to help terraform Mars, astronauts have brought (among other things) N (0<=N<5) young, healthy pineapple plants.

This particular type of pineapple reproduces asexually in the following way:

1. A single pineapple plant produces K (0<=K<15) new pineapple in one growing season.
2. At the end of the growing season, the new pineapples are adults, and the old ones are dead.
3. Increased levels of radiation have a P (0<=P<=1) chance of sterilizing any new pineapple that develops on Mars. This probability is independent for each pineapple.

What is the probability that the pineapple population will never die out?

Input

The first line is the number of test cases (no more than 10^5). Each of the following lines describes a test case. The integers N and K and the decimal number P are separated by single spaces.

Output

There will be one line for each test case. Each line will have the probability of eventual survival in percent, to two decimals, followed by the percent sign.

Example

Input:

```
5
1 3 0.6666666666666666
1 3 0.65
1 1 0
1 0 1
3 4 0.7101634622811129
```

Output:

```
0.00%
13.83%
100.00%
0.00%
70.94%
```

Added by: Paul Draper
Date: 2009-05-05
Time limit: 1s-4.5s
Source limit:50000B
Languages: All

SPOJ Problem Set (classical)

4343. Empty Boxes

Problem code: EBOXES

N large empty boxes (assume they are of type:1) are initially placed on a table. An unknown number of boxes (type:1) are selected and in each of them **K** smaller boxes (type:2) are placed. Again an unknown number of type:2 boxes are selected and **K** boxes of type:3 are placed inside. This process is repeated **T** times. Now a box is assumed to be empty when it has no smaller boxes inside it. Finally after all the processes are complete let there be **F** empty boxes in total.

LIMITS

1< N,K,T,F <1000000

Input

First line of the input file contains the number of test cases. Then each line contains 4 integers N,K,T,F as described above.

Output

Each line should contain the total number of boxes on the table.

Example

Input:
```
1
11 8 2 102
```

Output:
```
115
```

Added by: abhijith reddy d
Date: 2009-05-07
Time limit: 5s
Source limit:50000B
Languages: All

SPOJ Problem Set (classical)

4357. Enter the Matrix

Problem code: MATRIX1

Neo has to enter again in the Matrix

Neo has to enter again in the Matrix. The Matrix is represented as a two dimensional plane of coordinates x-y. When the Matrix was born, agent Smith showed up and imposed control. Unfortunately he is not alone, other agents are around him to do the same work. An agent is at some point in the Matrix given by a pair (Xi, Yi). Neo is now very furious because he cut his hair and wants to attack two agents. Neo can travel any distance to attack the first agent, but later he will be very tired, so he wants to walk the smallest distance possible to the next agent.

Neo is very busy conquering Trinity, who is a very difficult girl in terms of taking her out on a date, and he has no time to think about silly things like this, all he can think about is fun and war... Please, help him find the smallest distance possible from the first agent to the second.

Input

In the first line of the input appears one integer: T. This representes that your program must solve exactly T data sets for Neo. Each data set is not related to the previous one. In the first line of each data set appears one integer: N, representing that there are N agents in that layout of agents. Following this line, N pairs of integers will appear, each pair in a new line, representing the position of each agent.

Output

For each data set your program must output only one line represents the smallest distance between two agents. Please for avoid decimal roundings write the result whit zero decimal places after the comma.

Example

Input:

2

4

1 1

-1 3

4 4

2 2

3

1 1

2 4

5 1

Output:

1

3

Constraints

- 1 <= T <= 20

- 2 <= N <= 100000

- -1000000 <= Xi, Yi <= 1000000

Added by: Reinier César Mujica Hdez
Date: 2009-05-13
Time limit: 0.5s
Source limit:50000B
Languages: All
Resource: My own problem

SPOJ Problem Set (main)

4407. Counting Arborescence

Problem code: DAGCNT

"In graph theory, an arborescence is a directed graph in which, for a vertex v called the root and any other vertex u, there is exactly one directed path from v to u. In other words, an arborescence is a directed, rooted tree in which all edges point away from the root. Every arborescence is a directed acyclic graph."

-- from Wikipedia, the free encyclopedia

You are given a directed graph with *N* vertices, and your task is to count the number of different arborescences of size *N* that can be found in the given graph.

Two arborescences are considered different when they consist of different edges.

Input

Input consists of multiple test cases.

For each test case, the first line contains one integer *N* described as above.

N lines follows, each consists of *N* characters, either '0' or '1', representing the adjacency matrix of the graph.

The directed graph contains edge (*i*,*j*) if and only if the *j*th character of the *i*th line of the matrix is '1'.

The graph consists of no more than 8 vertices.

End of input is indicated by a line consisting of a single 0.

Output

For each test case, output one line consisting of one single integer, the number of arborescences.

Example

Input:

```
2
00
00
2
01
10
0
```

Output:

```
0
2
```

Added by: Blue Mary
Date: 2009-05-23
Time limit: 11s
Source limit:50000B
Languages: All except: C99 strict
Resource: Fudan University Local Contest #1

SPOJ Problem Set (classical)

4408. Build a Fence

Problem code: FENCE1

There is a wall in your backyard. It is so long that you can't see its endpoints. You want to build a fence of length L such that the area enclosed between the wall and the fence is maximized. The fence can be of arbitrary shape, but only its two endpoints may touch the wall.

Input

The input consists of several test cases.

For every test case, there is only one integer L ($1 <= L <= 100$), indicating the length of the fence.

The input ends with $L=0$.

Output

For each test case, output one line containing the largest area. Your answer should be rounded to 2 digits after the decimal point.

Example

Input:
```
1
0
```

Output:
```
0.16
```

Added by: Blue Mary
Date: 2009-05-23
Time limit: 1s
Source limit: 50000B
Languages: All except: C99 strict
Resource: Fudan University Local Contest #1, practise session

SPOJ Problem Set (classical)

4409. Circle vs Triangle

Problem code: AREA1

You are given a triangle and a circle in a plane. You can arbitrarily rotate or move them. What's the maximum possible area of their overlapping region?

Input

Input consists of one or more lines. For each line, there are four integers describing one test case: the lengths of three sides of a triangle a, b, c; and the radius of a circle r; where $1 <= a <= b <= c <= 100$, $1 <= r <= 100$, $a+b>c$.

End of input is indicated by a line consisting four zeros.

Output

For each test case, output a single line showing the largest overlapping area of the circle and the triangle. We accept solutions with absolute error less than 10^{-2}.

Example

Input:

```
3 4 5 1
5 5 8 4
0 0 0 0
```

Output:

```
3.14
12.00
```

Added by: Blue Mary
Date: 2009-05-23
Time limit: 11s
Source limit: 50000B
Languages: All except: C99 strict
Resource: Fudan University Local Contest #1

SPOJ Problem Set (main)

4410. Repair the Door

Problem code: REPAIR1

Once upon a time, there was a famous university called Famous University. As thousands of students studied and lived in FU, a gigantic residential building was built, which is called 'B37'. All students lived in B37 happily.

After decades, FU is still as famous as it had been in the past; however, the students living in it are now unhappy, because B37 is too old. Although the door of the building looks fine, it can be easily broken when being opened by some careless student too forcefully.

So, Blue Mary, the accommodation officer of B37, is facing an extremely serious problem.

With some mysterious methods, Blue Mary has predicted that exactly *N* students will enter or exit B37 during the next term. Unfortunately she doesn't know who the careless ones are, so she assumes that every student opening the door has a probability of *P* percent to be a careless one. When the door is broken by some careless guy, Blue Mary may repair it immediately or after some time, with a cost of *A* yuan. Unfortunately when a student goes through the door and finds it already broken and not repaired, she will report it to the headmaster, and Blue Mary will be subject to a fine of *B* yuan. The door is in good condition before the term begins, and will be repaired by the university after the term ends, so Blue Mary can leave the door unrepaired at the end of the term.

Being good at mathematics, Blue Mary has made a strategy, to decide when to and when not to repair the door, in order to minimize her expense.

Please write a program to calculate the expectation of her expense.

Input

The input consists of multiple test cases, the number of them is about 200000.

For each test case, there is one line containing four non-negative integers *N*, *P*, *A*, *B* described as above, with 0<=*N*<=100000, 0<=*P*<=100, 0<=*A*<=100, 0<=*B*<=100.

End of input is indicated by a line consisting of four zeros.

Output

For each test case, output one line containing the expectation of Blue Mary's minimal expense. We accept solutions with absolute error less than 10^{-4}.

Example

Input:

```
10 100 0 1
10 100 1 0
2 50 2 1
0 0 0 0
```

Output:

```
0.0000
0.0000
0.5000
```

Hint

In the first sample, the door will be broken every time it is opened, but repairing is free, so just repair it every time.

In the second sample, nothing will be fined, so just leave the door unrepaired.

In the last sample, if the door is broken by the first student, Blue Mary will be fined 1 yuan, otherwise she doesn't need to pay anything.

Added by:	Blue Mary
Date:	2009-05-23
Time limit:	11s
Source limit:	50000B
Languages:	All except: C99 strict
Resource:	Fudan University Local Contest #1

SPOJ Problem Set (main)

4411. Counting Expressions

Problem code: EXPR3

Count the number of distinct expressions involving *n* different operands **a**, **b**, **c**, etc. Only operators +, -, *, / and parentheses are permitted.

Two expression are distinct if for some valid input values (i.e. You won't divide some number by zero) **a**, **b**, **c**, ... , the two expressions leads to different results. For example, a/b/c and a/(b*c) are the same expressions, but a/b+c and a/(b+c) are not.

Input

Multiply test cases. For each test case:

A single line - *n*.(1<= *n* <=50).

Input terminates by a single zero.

Output

For each test case:

The number of different expressions, modulo 499999999999993.

Example

Input:
```
3
0
```

Output:
```
68
```

Added by: Blue Mary
Date: 2009-05-23
Time limit: 30s
Source limit:50000B
Languages: All except: C99 strict
Resource: Fudan University Local Contest #1

SPOJ Problem Set (classical)

4412. Factorization, Factorization, Factorization

Problem code: FACTOR1

Factorize $x^n - 1$ into several irreducible polynomials over the integers.

Input

Multiply test cases. For each test case:

A single line - n.($2 <= n <= 1200$).

Input terminates by a single zero.

Output

For each test case, output the factorization of the given polynomial.

There are multiple ways to express the factorization of a polynomial. To make it unique, we sort the irreducible polynomials according to the following rules:

Lower order polynomials are always lexicographically smaller than higher order polynomials. Same order polynomials should be sorted by their coefficients. We compare the coefficients from high degree terms to low degree terms, including the omitted terms, which the coefficients are regard as 0. Coefficients are being compared first by absolute value then by sign. Smaller absolute values are lexicographically smaller. For the same absolute value, negative coefficients are lexicographically smaller than positive coefficients.

See example for more output format details.

Example

Input:
```
12
0
```

Output:
```
(x-1)(x+1)(x^2+1)(x^2-x+1)(x^2+x+1)(x^4-x^2+1)
```

Hint

There should **not** be a "*" between digit and x, i.e. -2x^2 should be printed as -2x^2.

Added by: Blue Mary
Date: 2009-05-23
Time limit: 11s
Source limit:50000B
Languages: All except: C99 strict
Resource: Fudan University Local Contest #1

SPOJ Problem Set (main)

4413. Gem

Problem code: GEM

You are given a board with 8*8 squares. In each square, there can be either a colored gem or no gem at all. Gems with different colors are represented by different integers. It is guaranteed that there are no more than two consecutive gems with the same color either in a row or in a column.

```
........
........
........
........
........
..43366.
..121556
44212335
```

For two neighboring (up, down, left or right, we don't consider diagonal neighbors) squares, you can exchange the gems.

```
........
........
........
........
........
..43366.
..111556
44222335
```

You can also exchange a gem with a space. After that, if there are more than two consecutive gems with the same color in a row or in a column after exchange, these gems will be taken away simultaneously. Note that a gem could be counted both in its row and in its column; refer to the sample test cases for details.

```
........
........
........
........
........
..43366.
.....556
44...335
```

If there is no gem under a gem, the gem will fall to the square below.

```
........
........
........
........
........
.....66.
.....556
44433335
```

After all the squares falling down to the floor or another gem square, repeat the procedure until there's no gem can be taken away: if there are more than two gems with the same color in a row or in a column, these gems will be taken away simultaneously. Then some gems will fall to the squares below, if there are no gems under those gems.

```
........
........
........
........
........
.....66.
.....556
.......5

........
........
........
........
........
........
.....666
.....555

........
........
........
........
........
........
........
........
```

Given a board with 8*8 squares. This board is stable and you can't take away any gems in the original board. Your task is to determine whether all gems can be taken away by a single exchange or not.

Input

The input consists of several test cases. Each test case will be eight lines, and each line contains eight characters. If in a square there is no gem, '.' is used to identify it, otherwise an integer k is used to identify the gem's color, $1 <= k <= 9$.

There is a blank line between two consecutive test cases.

End of input is indicated by a line consisting of 0.

Output

For each test case, output a single line. If all gems can be taken away by a single exchange, output **Yes**; otherwise output **No**.

Example

Input:

```
........
........
........
........
```

```
........
..43366.
..121556
44212335

........
........
........
.2......
.2.22...
.1.11...
.2.22...
.2.22...

12121212
21212121
12121212
21212121
12121212
21212121
12121212
21212121

........
........
........
........
........
...96...
...96...
.996966.

0
```

Output:

```
Yes
Yes
No
Yes
```

Added by: Blue Mary
Date: 2009-05-23
Time limit: 1s
Source limit: 50000B
Languages: All except: C99 strict
Resource: Fudan University Local Contest #1

SPOJ Problem Set (main)

4414. Highway

Problem code: HIGHWAY1

As we all know, every day, there are hundreds of trucks passing through the highway. As some of the trucks might carry several tons of goods, the highway will be damaged unless it is frequently repaired. The administration of highway is worried about this, so it invented repairing cars to ensure that the cars can pass through the highway.

The highway has an initial durability. If a truck with x tons' goods pass the highway, its durability will be decreased by x. Once the highway's durability is less or equal to zero, it will be broken and can never be repaired. The trucks can't pass through the broken ones.

There are two kinds of repairing cars: T1 can increase the highway's durability by r, T2 can increase the highway's durability to p, if the highway's durability is less than p. Although the repairing cars can pass through the broken parts, the broken parts can't be repaired.

Input

The input consists of several test cases.

For every test case, there are three integers N (1<=N<=100000), M (1<=M<=100000), I (1<=I<=1000) in the first line, indicating the highway's length, the numbers of cars and the initial durability of the highway.

Each of the next M lines described the information of cars in the following format:
1 s t d-- There is a truck with d tons' goods wanted to pass the interval [s, t]. You should check whether the truck can pass it. Notice that if the truck can't pass the whole interval, it will give up the whole passing; otherwise it can pass the highway freely, even if the highway will be broken after the truck's passing.
2 s t r-- A T1 car will pass the interval [s, t] and increase its durability by r.
3 s t p-- A T2 car will pass the interval [s, t] and increase its durability to p.

You can assume that 1<=s<=t<=N, 1<=d, p, r<=1000

The input ends with N=M=I=0.

Output

For each case, you should return how many trucks can successfully pass the interval.

Example

Input:

```
5 5 5
1 1 3 3
2 2 3 10
1 1 3 3
```

```
1 1 3 1
1 2 3 1
5 3 10
1 1 2 5
1 2 3 5
1 1 3 5
0 0 0
```

Output:

```
3
2
```

Hint

In the second test case, the third truck can't pass the road, because although the durability of interval [1, 2) and (2, 3] is larger than 0, in position 2, the durability is 0.

Added by: Blue Mary
Date: 2009-05-23
Time limit: 11s
Source limit: 50000B
Languages: All except: C99 strict
Resource: Fudan University Local Contest #1

SPOJ Problem Set (main)

4415. Power of Integer

Problem code: INTEGER1

For a given positive integer y($y > 1$), if we can find a largest integer k and a smallest positive integer x, such that $x^k=y$, then the power of y is regarded as k.

Calculate the sum of the power of the integers from a to b. ($2 <= a <= b <= 10^{18}$)

Input

The input consists of multiple test cases.

For each test case, there is one line containing two integers a and b.

End of input is indicated by a line containing two zeros.

Output

For each test case, output the sum of the power of the integers from a to b.

Example

Input:
```
2 10
248832 248832
0 0
```

Output:
```
13
5
```

Added by: Blue Mary
Date: 2009-05-23
Time limit: 11s
Source limit: 50000B
Languages: All except: C99 strict
Resource: Fudan University Local Contest #1

SPOJ Problem Set (main)

4416. Jumping Hands

Problem code: JUMP1

In Shanghai, there are some famous clock towers built more than 100 years ago, such as the office building of Shanghai Customs, Xujiahui Church, etc. Every clock tower's clock-face consists of three hands: an hour hand, a minute hand, and a second hand. The hands are not move smoothly as time passing. When a second passes, the hands jump to the next place clockwise: the second hand moves 1/60 ring, the minute hand moves 1/3600 ring and the hour hand moves 1/43200 ring.

We assume that every hand's mass is evenly distributed, and its thickness can be ignored. In other words, every hand's centre of gravity is at the middle position of the respective hand. Suppose p_1, p_2 and p_3 are, respectively, the position of the centre of gravity of the hour hand, that of the minute hand, and that of the second hand. The position of all three hands' centre of gravity, *P*, is the average of p_1, p_2 and p_3 weighted by the hands' mass.

Suppose P_1 and P_2 are the positions of the three hands' centre of gravity at the start time and at the end time, respectively. Your task is to calculate the length of the path from P_1 to P_2. That is, if P_3 is the position of the three hands' centre of gravity on the clock-face at a point of time between start time and end time, *X* is the length of the path from P_1 to P_3, and *Y* is the length of the path from P_3 to P_2, the length of the path from P_1 to P_2 is *X*+*Y*.

Input

Input consists of one or more lines, each line describing one data set. Each line begins with 6 integers: L_1, L_2, L_3, M_1, M_2, M_3, followed by start time and end time. L_1, L_2 and L_3 indicate the lengths of hour hand, minute hand and second hand respectively, where 1<= L_1<L_2<L_3 <=100. M_1, M_2 and M_3 indicate the weights of hour hand, minute hand and second hand respectively, where 1<= M_1, M_2, M_3 <=100. The format of start time and end time is *hh:mm:ss*, where 0<= *hh* <=23, 0<= *mm* <=59, and 0<= *ss* <=59. Start time and end time should be in the same day.

End of input is indicated by a line consisting of -1.

Output

For each data set, output a single line. Each line should give the length of the path for the positions of three hands' centre of gravity at the start time and at the end time. We accept solutions with absolute error less than 10^{-2}.

Example

Input:

```
1 2 3 1 2 3 00:00:00 00:00:01
3 4 5 1 1 1 09:00:00 18:00:00
-1
```

Output:

```
0.08
2826.27
```

Added by:	Blue Mary
Date:	2009-05-23
Time limit:	11s
Source limit:	50000B
Languages:	All except: C99 strict
Resource:	Fudan University Local Contest #1

SPOJ Problem Set (classical)

4420. Counting Graphs

Problem code: KPGRAPHS

In this problem your task is to count the amount of graphs of different types. We only consider undirected graphs without self-loops. Every pair of vectices can be connected with at most one edge. Graphs are labeled, i.e. if a graph has N vertices, then each of them has a unique label from 1 to N.

We will be interested in three types of graphs - connected, eulerian and bipartite. A graph is connected, if and only if there is at least one path between any pair of vertices. A graph is eulerian, if and only if it's connected and there is a cycle that goes through every edge exactly once. A graph is bipartite, if and only if we can split all of its vertices into two subsets A and B, such that every edge has one endpoint in A and another in B.

Input

The first line of the input contains one integer number T (1 <= T <= 1000) - the number of test cases.

Next T lines contain different test cases. Each test case contains one integer number N (1 <= N <= 1000) - the number of vertices in a graph.

Output

For each test case, output the number of connected graphs, the number of euleran graphs and the number of bipartite graphs - all modulo 1000000007. See examples for the required format. Output one additional empty line after each test case.

Example

Input:212**Output:**Connected: 1Eulerian: 1Bipartite: 1Connected: 1Eulerian: 0Bipartite: 2

Added by: Pavel Kuznetsov
Date: 2009-05-25
Time limit: 5s
Source limit:7000B
Languages: All

SPOJ Problem Set (classical)

4421. Irreducible polynomials over GF2

Problem code: GF2

Find the number of degree **n** irreducible polynomials over GF(2). For example: for n=1 there are two such polynoms: x and x+1. For **n**=2 there is only one: x^2+x+1. Note that in R[x] the polynom x^2+1 is irreducible, but not over GF(2), because $x^2+1=(x+1)*(x+1)$

Input

A single positive integer **n**, where **n**<500000

Output

Output the answer for **n.**

Example

```
Input:201Output:1598943327620885846310410042130510052260825081399500494621 8Input:1Output:2Input:2Output:1Input:3Output:2
```

Added by:	Robert Gerbicz
Date:	2009-05-25
Time limit:	0.5s-6s
Source limit:	4096B
Languages:	All
Resource:	classic problem, own input

SPOJ Problem Set (classical)

4429. Spelling Lists

Problem code: MIB

J, of the Men In Black, has been learning an alien language and has has a spelling test tomorrow. J, however, is bored of studying the nonsensical (and often unpronouncable) words.

Instead, he is seeing how many ways he can reorder his spelling list. After making all possible permutations of word on his list, he sorts the rearranged lists lexiographically (by the first word, then the second...). After the sort, in what position, with the lexiographically first list being in position 1, is his original spelling list?

Input

The first line is the number of spelling lists (no more than 10).

For each spelling list, a line with the number of words (no more than 1000) is given, followed by the original list on the next line.

All words within a spelling list are unique. Each word is composed of the letters a-z, is fewer than 100 characters, and is followed by a single space.

Output

On separate lines, give the positions of the original lists.

Example

Input:`4
a b c d
4
d c b a
1
mrsmith
6
a aaaaa aaaaa aaaa b bb` **Output:**`1
24
1
55`

Added by: Paul Draper
Date: 2009-05-28
Time limit: 1s-2s
Source limit: 50000B
Languages: All

SPOJ Problem Set (classical)

4452. Simple Arithmetics II

Problem code: ARITH2

While browsing aimlessly, Peter stumbled upon an old riddle he used to solve on his calculator when he was still young. It was the kind of a riddle where you punch in a bunch of numbers and operators into a simple pocket calculator and then turn it upside down to get the answer:

*These come in many different sizes but they are always exactly one foot long. Answer: 103 * 103 * 5.*

*What are made of ice to keep people warm? Answer: 50 * 40 * 250 + 791.*

After a few minutes he found a large amount of such riddles and full of excitement he went to solve them. He turned his computer screen upside down...

... only to find out that he does not have a reasonable calculator program installed on his computer.

Problem specification

You are given multiple sequences of button presses of a simple pocket calculator that consist of digits and arithmetic operators. For each such sequence find the number it would produce on a pocket calculator's display.

Note that the pocket calculator evaluates the operators in the order in which they are given. (i.e., there is no operator precedence.) Assume that the display of the calculator is large enough to show the result, and that its memory is sufficient to store all intermediate results.

Input specification

The first line of the input file contains an integer T specifying the number of test cases. Each test case is preceded by a blank line.

Each test case represents one sequence of button presses for a pocket calculator. The sequence consists of non-negative integers and arithmetic operators and ends with an equal sign. It may also contain spaces to improve readability.

The operator / represents integer division, rounded down. You may assume that no test case contains division by zero and that in all test cases all intermediate results are non-negative.

Tip: **long long int** in C/C++, **long** in Java or **int64** in Pascal is enough for this problem.

Output specification

For each sequence from the input file output the number that would be displayed on the calculator.

Example

Input:

```
4

1 + 1 * 2 =

29 / 5 =

103 * 103 * 5 =

50 * 40 * 250 + 791 =
```

Output:

```
4
5
53045
500791
```

Hint

The first test case shows that there is no operator precedence.

The second one shows that integer division always rounds down.

The last two outputs are the answers to the two riddles in the problem statement: "shoes" (53045 upside down), and "igloos"(500791 upside down).

Added by: Blue Mary
Date: 2009-05-31
Time limit: 3s
Source limit:50000B
Languages: All except: C99 strict
Resource: IPSC 2009

SPOJ Problem Set (classical)

4453. Bouncing Balls II

Problem code: BOBALLS2

"Behold, my queen", said the jester, "the great Bouncing Ball Bowl!" The queen boredly waved her hand and sarcastically replied: "Let the fun begin!". And the fun begun! The jester spoke a magic word and all the colorful balls in his bowl started to roll and bounce, creating interesting pictures.

The queen watched vividly for a few minutes, but then she started to be bored again. "Just wait a moment, Your Majesty, in a minute they'll..." started the jester, but the queen interrupted: "I'm a queen! I don't want to wait! Can't you just fast forward it or something?"

Problem specification

The jester's box is an $X * Y$ ($1 <= X, Y <= 5000$)rectangle. The rectangle contains N($N <= 3001$) small balls. At any moment, each ball is travelling at the same speed in one of the four diagonal directions.

The movement of the balls is continuous and for the purpose of this problem we may consider them to be points. When two or more balls meet, they bounce in a way described below.

Your task is to determine the state of the box at given moments in time.

Bouncing specification

Bouncing does not change the speed of the balls. Following images show how the balls bounce off each other, and also off walls. Each image can be rotated arbitrarily. For example, the first image shows that whenever two balls meet at a right angle, they bounce and depart at a right angle again. One particularly tricky case is shown in the third image.

Input specification

The input starts with a line containing the dimensions X and Y of the box. We will use a coordinate system with axes parallel to the sides of the box, (0,0) at one of the corners and (X,Y) at the opposite corner.

The second line contains the number of balls N.

Each of the next N lines contains four integers x,y,vx,vy, where (x,y) are the coordinates of one ball at time 0 and (vx,vy) is its current velocity vector. (Each ball will be strictly inside the box and for each ball both the absolute values of vx and vy will be equal to 1. No two balls will start at the same place.)

The following line contains the number of queen's requests M.($1 <= M <= 20$)

On the last line there are M numbers $t_1,...,t_M$ - the points in time the queen wants to see. These numbers will be less than 10^{12}.

Output specification

As a solution to this problem, we expect a file with M blocks, with the i-th block describing the situation at time t_i.

Each block must contain N lines, and each line must contain the x and y coordinates of one ball. The balls in each block must be sorted - primarily by to their first, secondarily by their second coordinate at that point in time.

You may output an empty line between the blocks.

Example

Input:

```
6 4
4
1 2 1 1
5 2 1 1
2 1 1 -1
3 1 -1 -1
1
4
```

Output:

```
1 3
3 2
5 2
6 3
```

Note that the balls that start at (2,1) and (3,1) bounce off each other at a non-integer point in time.

Added by: Blue Mary
Date: 2009-05-31
Time limit: 3s
Source limit: 50000B
Languages: All except: C99 strict
Resource: IPSC 2009

SPOJ Problem Set (classical)

4454. Brackets II

Problem code: BRCKTS2

Peter is preparing slides for his lecture on parsing arithmetic expressions. In the first part of the lecture he wants to focus just on parsing brackets. He invented an interesting geometric representation of a correct bracket sequence for his students, because one image is better than a thousand words:

Formally, the definition of the geometric representation looks as follows. The simplest correct bracket sequence () is represented by a 1 * 1 square. If A is a correct bracket sequence and g(A) its represenation, then the representation for (A) is g(A) surrounded by a rectangle two units wider than g(A) and one unit taller than the highest point of g(A). If A and B are two correct bracket sequences and g(A) and g(B) are their representations, then we get g(AB) by placing g(B) one unit to the right of g(A).

After he finished his slides, Peter started to play with the images he prepared. He painted the bounded areas of the images alternately black and white, in such a way that the outer-most areas are all painted black. For the example above this coloring looks as follows:

Problem specification

You are given a correct bracket sequence. Calculate the area that is colored black.

Input specification

The first line of the input file contains an integer T specifying the number of test cases. Each test case is preceded by a blank line.

Each test case consists of one line with a correct bracket sequence with length less than 350000. Every line will only contain characters (and).

Output specification

For each test case output one line with one integer - the area of the black part of the corresponding geometric representation.

Example

Input:
```
2

((()))

(())(()(()))
```

Output:
```
10
20
```

Added by: Blue Mary
Date: 2009-05-31
Time limit: 1s-7s
Source limit: 50000B
Languages: All except: C99 strict
Resource: IPSC 2009

SPOJ Problem Set (classical)

4455. Going to the Movies

Problem code: MOVIE

N(N <= 120) friends decided to go to the local cinema together. They all bought tickets to the same row. As there was still some time left, each of them took her ticket and went shopping until the movie starts.

They all arrived back late, the movie already started. The usher standing at the door agreed to let them in one by one. Each of the girls was supposed to find her place and sit down.

However, the machine that printed their tickets was broken. Instead of consecutive numbers, each girl received a random seat number between 1 and K, where K is the number of seats in their row. The seat numbers they received were not necessarily distinct.

When a girl tries to sit down, she enters the row at the end where seat number 1 is, and walks until she reaches the number on her ticket. If her desired seat is free, she just sits down. If it is already taken, she continues to walk in the same direction until she finds the first free seat, and sits there.

Of course, it is possible that some unfortunate girl will reach the end of the row without finding a place to sit. In that case, the usher comes and throws her out.

Problem specification

You are given the numbers N and K.(1<= K <=200)

Assume that each girl's ticket had a number between 1 and K, inclusive. Each number was drawn uniformly at random, and draws were independent.

Also assume that the entire row was empty when the first girl started to look for her seat.

Compute the probability that at least one girl suffered the sad fate of being thrown out by the usher.

Input specification

The first line of the input file contains an integer T specifying the number of test cases. Each test case is preceded by a blank line.

Each test case consists of a single line containing two integers N and K.

Output specification

For each test case output a single line with the probability as a simplest fraction. (Do not output any spaces before or after the / sign.)

Example

Input:

```
3

1 10

2 3

3 3
```

Output:

```
0/1
1/9
11/27
```

In the third case there are $3^3 = 27$ possibilities. Out of these, in 11 some girl is thrown out. These 11 sequences are: 133, 222, 223, 232, 233, 313, 322, 323, 331, 332, and 333.

For example, if the sequence of numbers were 322, the first girl sits at seat #3, the second one at #2, and then the third one tries to sit at #2, but finds both seat #2 and seat #3 occupied, and she's thrown out.

Added by:	Blue Mary
Date:	2009-05-31
Time limit:	1s-7s
Source limit:	50000B
Languages:	All except: C99 strict
Resource:	IPSC 2009

SPOJ Problem Set (classical)

4456. Jumbo Airlines

Problem code: AIRLINES

The new catchphrase of Jumbo Airlines is "No annoying neighbors, each flight a unique experience!"

And as in most cases, the advertisement was produced by the marketing department, without ever consulting the engineers. They only learned about it after the boss asked them to "handle it ASAP".

There are M seats in each row, and there are N rows of seats in the airplane. Hence the seats form an M * N grid. (For the purpose of this problem we will ignore the presence of aisles.) The airline sells exactly K tickets for each flight.

To make sure that the "no annoying neighbors" part of the motto is satisfied, the seating must obey the following rule: Whenever a seat is occupied, the seats immediately in front of it and behind it, as well as the seats immediately to the left and to the right must remain free.

An allowed arrangement is a set of K occupied seats that obeys the rule above.

The "unique experience" part of the motto is then satisfied by using a different arrangement of occupied seats for each flight. (Two seating arrangements are different if there is at least one seat which is occupied in one arrangement and free in the other.)

Problem specification

You are given the numbers M, N and K. Find the number of different allowed seating arrangements.

As this number can be very large, we're only interested in its value modulo 420047.

Input specification

Multiple test cases, seperated by blank lines.

Each test case consists of a single line containing three integers M, N and K. (M <= N)

There are two kinds of input:

The input of the first kind satisfies that 1<= M <=15, 1<= N <=50, 3<= K <=50.

The input of the second kind satisfies that 1<= M<= 4, 1<= N <= 10^9, 3<= K <=5.

Input terminates by EOF.

Output specification

For each test case output a single line with the number of allowed arrangements modulo 420047.

Example

Input:
```
2 4 4
```

Output:
```
2
```

Hint

The input file can be downloaded here. You may submit a TEXT file - the corresponding output file.

Added by:	Blue Mary
Date:	2009-05-31
Time limit:	1s
Source limit:	50000B
Languages:	TEXT
Resource:	IPSC 2009

SPOJ Problem Set (classical)

4457. Shopping II

Problem code: SHOP2

Karl is going to spend his holiday in Nothingland. Since there is nothing there, he has to buy all supplies now. At the moment, he is waiting at the checkout counter with a shopping cart full of stuff.

Of course, he has a sufficient amount of money in his wallet. However, he prefers to use alternate means of payment if possible: luncheon vouchers, gift certificates, different types of coupons, etc. What makes the matter complicated is that the use of these items is often limited: e.g., luncheon vouchers can only be used to buy food, and gift certificates are often limited to a certain type of gifts.

Problem specification

You are given the number N(1<= N <=2000) of items in Karl's shopping cart and their prices. You are also given the number M(1<= M <=2000) of vouchers in his wallet, together with the information on their allowed use.

When paying for his shopping, Karl may use vouchers for a larger sum than the cost of the things he is buying. It is also possible to split an item's cost between multiple vouchers and use a voucher to pay for more than one item.

Compute the minimum amount of additional cash money Karl needs to pay for his shopping.

Input specification

The first line of input file contains an integer T specifying the number of test cases. T blocks follows, each block describes one test case. Each block is preceded by a blank line.

Each block starts with line containing two positive integers N (the number of items) and M (the number of vouchers). The second line contains N numbers(each no more than 10000), the i-th of them being the price of the i-th item in Karl's shopping cart. The third line contains M numbers, the i-th of them being the cash value of the i-th voucher Karl has in his wallet. M lines follow. Each line consists of a number K_i (the count of items such that you can pay for them using the i-th voucher, no more than 100) followed by K_i numbers (the numbers of those items; items are numbered from 1 to N).

Output specification

For each test case output a single number specifying how much cash money Karl needs to pay for his shopping.

Example

Input:

```
1

3 2
15 20 10
20 30
3 1 2 3
1 3
```

Output:

```
15
```

Added by: Blue Mary
Date: 2009-05-31
Time limit: 7s
Source limit: 50000B
Languages: All except: C99 strict
Resource: IPSC 2009

SPOJ Problem Set (classical)

4460. Tic Tac Toe - Best Move

Problem code: TITATO

Tic-tac-toe is played on a 3x3 grid between two players. All squares begin empty, and play is as follows:

1. The first player places an X on any empty square.
2. The second player then places an O on any emtpy square.
3. The first player then places another X onany empty square
4. ...

The players continue in this matter until one of the following occurs:

- Three X's form a straight line (horizontal, vertical, or diagonal), in which case the first player wins.
- Three O's form a straight line, in which case the second player wins.
- The grid is filled with X's and O's (with no three X's or three O's in a line), in which case the players tie.

Le the tic-tac-toe grid be numbered as follows:

1	2	3
4	5	6
7	8	9

Write a computer program that will decide the best move for the current player based on the previous moves of the two players. In case of multiple best moves, choose the one with the lowest number in the grid above.

The "best" move is defined as the move that guarantees the highest outcome (win>tie>loss). An outcome is "guaranteed" if the player can always cause it to happen, regardless of the other's player's moves.

All tic-tac-toe games given will still be in progress, i.e. there is an empty space and no player has won yet.

Input

The first line is the number of test cases (fewer than 2000) to follow.

Each test case consists of two lines. The first line is the number of moves that have already happened. The second line lists the grid numbers of the previous moves, in order, each followed by space.

Output

For each test case, output the best move with the lowest number for the current player.

Example

Input:421 2 21 5 0 81 2 3 4 5 8 6 9 **Output:**4217

Added by: Paul Draper
Date: 2009-06-01
Time limit: 1s
Source limit:50000B
Languages: All

SPOJ Problem Set (classical)

4465. The Ant

Problem code: ANTTT

There are **n** sticks lying on the ground. The Ant can move only along the sticks. It can go from one stick to another only if the sticks intersect or touch each other. Help the Ant find out if it can reach the stick **y** from the stick **x**.

Input

The first line of the input contains number **t** - the amount of tests. Then **t** test descriptions follow.The first line of each test contains two integers **n** and **m** - the number of stick and the number of queries. Next **n** lines contain four integers **Ax**, **Ay**, **Bx**, **By** - the coordinates of the endpoints of a stick. You may consider stick to be straight segment on a plane. The next **m** lines contain two integers each **x** and **y** which are the queries.

Constraints

1 <= **t** <= 100
1 <= **n**, **m** <= 1000
-10000 <= **Ax**, **Ay**, **Bx**, **By** <= 10000
1 <= **x**, **y** <= **n**

Output

For each query print "YES" if the Ant can reach the stick number **y** from the stick number **x**, otherwise print "NO".

Example

```
Input:23 31 3 4 33 4 3 13 1 5 11 21 32 23 31 1 3 12 1 3 13 2 4 11 21 32 3

Output:YESYESYESYESNONO
```

Added by:	Spooky
Date:	2009-06-02
Time limit:	7s
Source limit:	50000B
Languages:	All
Resource:	Advancement Spring 2009, http://sevolymp.uuuq.com/ author: elmariachi1414

SPOJ Problem Set (classical)

4476. Playfair Cracker

Problem code: PLAYFAIR

In this problem, you will crack Playfair cyphers or decide a multiple solutions exist, or decide no solution exist.

Rules:

1) Convert all letters in the text to uppercase and omit all non-alphabetic characters.

2) Replace all letters ’J’ by ’I’.

3) Form digraphs, but avoid having twice the same letter in a digraph. Insert an extra ’X’ between the identical letters if necessary. If the repeated letter is an ’X’, insert a ’Q’ instead.

4) If the last digraph would be incomplete, append an extra ’X’ to the text (or a ’Q’ if the last letter in the text is an ’X’).

Consider the following message: "Programming in C and Pascal is easy; I will learn Java next year."
The digraph representation would be:
PR OG RA MX MI NG IN CA ND PA SC AL IS EA SY IW IL LX LE AR NI AV AN EX TY EA RX
Note the extra ’X’ between the two ’M’s of ’programming’. There is no extra ’X’ between the two ’L’s of ’will’, because they are in different digraphs, but there is one between ’will’ and ’learn’. There is also an extra ’X’ at the end of the message. The ’J’ in ’Java’ is replaced by an ’I’.
To illustrate the exceptions for the letter ’X’ in the original text, consider the message "I am an ex-xenophobe, attempting to relax!". This becomes:
IA MA NE XQ XE NO PH OB EA TX TE MP TI NG TO RE LA XQ
The next stage is the replacement of each digraph by an other digraph according to the following rules:

The uppercase letters, ’J’ excluded, are placed in a 5X5 square in some predetermined order. This is the key for the encryption.

If the two letters of the digraph are in the same row of the square, replace them by the letters occuring at the immediate right of each one. If one of the letters is in the rightmost column, replace it by the letter in the first column of the same row (wrap around).

If the two letters of the digraph are in the same column of the square, replace them by the letters occuring immediately below each one. Wrap around to the same column in the top row if one of them is in the bottom row.
If the letters are neither in the same row nor the same column, replace the first letter by the letter in the same row as the first letter and the same column as the second letter. Replace the second letter by the letter in the same row as the second letter and the same column as the first letter.
The resulting digraphs form the encrypted code.

Input

The input contains several cases, the number of which is on the first line. Every case has three parts. The first part is the plaintext and consists of one or more lines of ordinary text. The second part is the code that is the result of encrypting the first part. The third part is code for the text you are to decrypt. The parts are terminated by a hashmark ('#') on a line by itself. Code parts are printed as uppercase digraphs, 20 digraphs on a line, separated by one space. The last line of a code part can contain fewer than 100 digraphs. No code part will contain more than 5000 digraphs.

Output

For each case, first output a line "Case x:" where x is the case number (starting from 1).
Then output the decrypted code represented as digraphs in the same format as the code parts in the input. If more than one solution exist, output the following on a single line.

MULTIPLE SOLUTIONS

If no solutions exist output in one line:

NO SOLUTIONS

Separate the cases by an empty line.

Example

Added by: Chen Xiaohong
Date: 2009-06-05
Time limit: 1s
Source limit:50000B
Languages: All
Resource: Enhanced Version of Regional Warmup Contest 2005

SPOJ Problem Set (classical)

4478. Counting Expressions II

Problem code: EXPR4

Count the number of distinct expressions involving *n* different operands **a**, **b**, **c**, etc. Only operators +, -, *, / and parentheses are permitted.

Two expression are distinct if for some valid input values (i.e. You won't divide some number by zero) **a**, **b**, **c**, ... , the two expressions leads to different results. For example, a/b/c and a/(b*c) are the same expressions, but a/b+c and a/(b+c) are not.

Input

Multiply test cases. For each test case:

A single line - *n*.(1<= *n* <=2000).

Input terminates by a single zero.

Output

For each test case:

The number of different expressions, modulo 1000000007.

Example

Input:
```
3
0
```

Output:
```
68
```

Added by: Blue Mary
Date: 2009-06-06
Time limit: 30s
Source limit:50000B
Languages: All except: C99 strict
Resource: Tomek Czajka

SPOJ Problem Set (classical)

4487. Can you answer these queries VI

Problem code: GSS6

Given a sequence A of N *(N <= 100000)* integers, you have to apply Q *(Q <= 100000)* operations: Insert, delete, replace an element, find the maximum contiguous(non empty) sum in a given interval.

Input

The first line of the input contains an integer N.
The following line contains N integers, representing the starting sequence A1..AN, *(|Ai| <= 10000)*.
The third line contains an integer Q. The next Q lines contains the operations in following form:
I x y: insert element y at position x *(between x - 1 and x)*.
D x : delete the element at position x.
R x y: replace element at position x with y.
Q x y: print max{Ai + Ai+1 + .. + Aj | x <= i <= j <= y}.
All given positions are valid, and given values are between -10000 and +10000.
The sequence will never be empty.

Output

For each **"Q"** operation, print an integer(one per line) as described above.

Example

Input:
```
53 -4 3 -1 6
10I 6 2Q 3 5R 5 -4Q 3 5D 2Q 1 5I 2 -10Q 1 6R 2 -1Q 1 6
```
Output:
```
8
3
6
3
5
```

Added by: Alfonso2 Peterssen
Date: 2009-06-08
Time limit: 2s-3s
Source limit: 50000B
Languages: All
Resource: my own

SPOJ Problem Set (classical)

4491. Primes in GCD Table

Problem code: PGCD

Johnny has created a table which encodes the results of some operation -- a function of two arguments. But instead of a boring multiplication table of the sort you learn by heart at prep-school, he has created a GCD (greatest common divisor) table! So he now has a table (of height a and width b), indexed from (1,1) to (a,b), and with the value of field (i,j) equal to gcd(i,j). He wants to know how many times he has used prime numbers when writing the table.

Input

First, $t <= 10$, the number of test cases. Each test case consists of two integers, $1 <= a,b < 10^7$.

Output

For each test case write one number - the number of prime numbers Johnny wrote in that test case.

Example

Input:
```
210 10100 100
```

Output:

```
302791
```

Added by: Yash
Date: 2009-06-12
Time limit: 10s
Source limit: 11111B
Languages: All
Resource: Codechef

SPOJ Problem Set (classical)

4492. Magic1

Problem code: MAGIC1

A Magic Square is an N by N matrix such that the sum of all the elements in each row, each column and the two diagonals is the same. Also, all the elements of a Magic Square are different and are in the range from 1 to N^2.

Your are given a matrix with some element missing. Your task is to complete the matrix so that it is a Magic Square (if it is possible).

Input

The first line of the standard input will contain one integer **N**, (1 <= **N** <= 5). Each of the next **N** lines will contain **N** integers representing the elements in the matrix. The number 0 represents an empty element.

Output

If it is possible to complete the matrix so that it is a Magic Square, then output the matrix to the standard output - in **N** lines output **N** integers separated by a single space that represent the matrix. If it is impossible to fill out the empty elements so that the marix is a Magic Square, then output -1 to the standard output.

Example

Input:

2
1 0
2 3

Output:-1

Added by: Tomas. Bob
Date: 2009-06-12
Time limit: 0.400s-1s
Source limit:50000B
Languages: All
Resource: www.z-trening.com

SPOJ Problem Set (classical)

4493. Equation

Problem code: EQUAD1

You are given an equation, which when you solve, you get a huge prize.

The equation consists out of integers between 1 and 1000, signs **+**, **-** (plus and minus) and parenthesis **(** and **)**, and finally one **?** which represents the unknown variable. The question mark can be on either side of the equation.

Input

From the first line of the standard input read the equation, the equation will contain at most 1000000 characters.

Output

You shoud write the wanted number.

Example

```
Input:5 + ? = 32 - 2Output:
```

25

Added by: Tomas. Bob
Date: 2009-06-12
Time limit: 0.009s
Source limit:50000B
Languages: All
Resource: www.z-trening.com

SPOJ Problem Set (classical)

4523. Binomial Coefficients

Problem code: UCI2009B

We all got too excited when we learned (A + B)^2 = A^2 + 2AB + B^2. After solving this problem, maybe you could get even more excited because you will have to calculate (A + B)^N, where (0 <= N <= 1000).

Follow the rules below when giving the answer:

1. Consecutive terms must be separated by a '+' character.
2. At the i-th term, A must be raised to N - i and B must be raised to i (0<=i<=N).
3. Binomial coefficients must not be printed, print their prime factorization instead.
4. Use '^' for exponentiation and 'x' for multiplication in step 3.
5. Avoid the use of number 1 when possible.

See sample output for clarification.

Input

Input starts with an integer T, representing the number of test cases (1<=T<=15). T lines follow, each one consisting of an integer N, (0<=N<=1000).

Output

For each test case, print (A + B)^N, on a single line.

Example

Input:6012345**Output:**1A+BA^2+2xAB+B^2A^3+3xA^2B+3xAB^2+B^3A^4+2^2xA^3B+2x3xA^2B^2+2^2xAB^3+B^4A^5+5xA^4B+2x5xA^3B^2+2x5xA^2B^3+5xAB^4+B^5**Warning: Large output. Be careful with certain languages.**

Added by: yandry pérez clemente
Date: 2009-06-23
Time limit: 3s
Source limit:50000B
Languages: All

SPOJ Problem Set (classical)

4525. Digger Octaves

Problem code: UCI2009D

After many years spent playing Digger, little Ivan realized he was not taking advantage of the octaves. Oops, sorry! Most of you were not born when Digger came to light!

Digger is a Canadian computer game, originally designed for the IBM personal computer, back in 1983. The aim of the game is to collect precious gold and emeralds buried deep in subterranean levels of and old abandoned mine.

We Digger gurus call a set of eight consecutive emeralds an octave. Notice that, by consecutive we mean that we can collect them one after another. Your Digger Mobile is able to move in the four directions: North, South, West and East.

In a simplified Digger version, consisting only of emeralds and empty spaces, you will have to count how many octaves are present for a given map.

Input

Input starts with an integer T, representing the number of test cases (1<=T<=20). Each test case consists of a map, described as follows:

An integer N (1<=N<=8), representing the side length of the square-shaped map. N lines follow, N characters each. A 'X' character represents an emerald, and a '.' represents an empty space.

Output

For each test case print the number of octaves on a single line.

Example

Input:
```
23
XXXX.
XXXX
3
XXXXXXXXX
```
Output:
```
1
5
```

Added by: yandry pérez clemente
Date: 2009-06-23
Time limit: 1s
Source limit: 50000B
Languages: All

SPOJ Problem Set (classical)

4528. Frog Wrestling

Problem code: FROGS

Billy Jean loves collecting frogs. Recently, she developed the sport of frog wrestling. Now she wants to rank her frogs by their wrestling prowess.

Billy Jean has made a algorithm for sorting her frogs.

1. She arranges N cages, numbered 1,2,...N, each with one frog.
2. For each pair of cages in a specified, pre-determined list of K pairs of cages,
    1. she removes the frogs from the two cages,
    2. has the frogs wrestle,
    3. puts the winner in the higher-numbered cage, and
    4. puts the loser in the lower-numbered cage.

When she is finished, she hopes to have all her frogs sorted from worst to best in the cages 1 to N. Will her algorithm work regardless of the initial order of the frogs?

Note:

- Assume that a strict ordering by wrestling ability is possible.
- Billy Jean isn't the sharpest tool in the shed. Sometimes she has written the same two numbers for a pair. In this case, that frog is simply taken out and then put back.

Constraints

1<=N<=20

1<=K<=1000

Input

The first line is the number of test cases. Each test cases is preceded by a blank line.

The first line of each test case is N. The next line is K. The next K lines are the pairs, separated by a single space.

Output

On separate lines, output whether Billy Jean's algorithm is correct. Output "YES" (without quotes) if it is or "NO" (without quotes) if it is not.

Example

Input:
```
4212 1211 1111 1451 23 41 32 42 3
```
Output:
```
YESNOYESYES
```

Added by: Paul Draper
Date: 2009-06-24
Time limit: 1s-7s
Source limit: 50000B
Languages: All

SPOJ Problem Set (classical)

4533. Determinant of Banded Matrices

Problem code: BANDMATR

Computing the determinant of a matrix using Gaussian elimination takes O(n^3). On the other hand, computing the determinant of tridiagonal matrix is O(n) using a recurrence. In this problem you will compute the determinant of banded matrices. A band matrix is a sparse matrix, whose non-zero entries are confined to a diagonal band, comprising the main diagonal and zero or more diagonals on either side. In this problem, given a banded **NxN** square integer matrix with **M** bands on each side of the diagonal, we ask you to compute the determinant of this matrix. For example a tridiagonal matrix has exactly 1 band on each side, and the 8x8 Matrix in the sample input has 2 bands on each side. For a good discussion of banded matrices, see Thorson's paper at:

http://sepwww.stanford.edu/oldreports/sep20/20_11_abs.html

Input

A total of <10 inputs. For each input,

First line has dimension, **N (1<N<501)**, of the matrix, followed by **N** lines with **N** integers, each less than **10001**, and greater than **-10001**. It is guarantteed that the number of bands on each side of the diagonal, **M < 51.** That is there are at most **101** bands in total including the diagonal. Use scanf IO, and avoid stl IO.

Output

For each input matrix, output its determinant **modulo 10^9+7.**

Hint: Use Montgomery multiplication for fast computation, i.e., see:
http://everything2.com/title/Montgomery%2520multiplication

Example

```
Input:22 00 221 00 181 0 -1 0 0 0 0 0-1 1 0 -1 0 0 0 0-1 0 -1 1 -1 0 0 00 -1 0 -1 0 -1 0 00 0 -1 0 1 0 -1 00 0 0 -1 -1 1 0 -10 0 0 0 -1 0 -1 10 0 0 0 0 -1 0 -1Output:4136
```

Added by: Chen Xiaohong
Date: 2009-06-26
Time limit: 0.5s
Source limit:20000B
Languages: All
Resource: classical numerical analysis

SPOJ Problem Set (classical)

4546. Tobo or not Tobo

Problem code: ANARC08A

The game of Tobo is played on a plastic board designed into a 3X3 grid with cells numbered from 1 to 9 as shown in figure (a). The grid has four dials (labeled 'A' to 'D' in the figure.) Each dial can be rotated in 90 degrees increment in either direction. Rotating a dial causes the four cells currently adjacent to it to rotate along. For example, figure (b) shows the Tobo after rotating dial 'A' once in a clockwise direction. Figure (c) shows the Tobo in figure (b) after rotating dial 'D' once in a counterclockwise direction.

Kids love to challenge each other playing the Tobo. Starting with the arrangement shown in figure (a), (which we'll call the standard arrangement,) one kid would randomly rotate the dials, X number of times, in order to shuffle the board. Another kid then tries to bring the board back to its standard arrangement, taking no more than X rotations to do so. The less rotations are needed to restore it, the better. This is where you see a business opportunity. You would like to sell these kids a program to advise them on the minimum number of steps needed to bring a Tobo back to its standard arrangement.

Input

Your program will be tested on one or more test cases. Each test case is specified on a line by itself. Each line is made of 10 decimal digits. Let's call the first digit Y . The remaining 9 digits are non-zeros and describe the current arrangement of the Tobo in a row-major top-down, left-to-right ordering. The first sample case corresponds to figure (c).
The last line of the input file is a sequence of 10 zeros.

Output

For each test case, print the result using the following format:
k. R
where k is the test case number (starting at 1,) is a single space, and R is the minimum number of rotations needed to bring the Tobo back to its standard arrangement. If this can't be done in Y dials or less, then R = -1.

Example

Input:
```
3413569728116543278900000000000
```
Output:
```
1. 22. -1
```

Added by: Ahmed Aly
Date: 2009-07-02
Time limit: 5s
Source limit:50000B
Languages: All
Resource: ANARC 2008

SPOJ Problem Set (classical)

4549. Adding Sevens

Problem code: ANARC08B

A seven segment display, similar to the one shown on the right, is composed of seven light-emitting elements. Individually on or off, they can be combined to produce 127 different combinations, including the ten Arabic numerals. The figure below illustrates how the ten numerals are displayed. 7-seg displays (as they're often abbreviated) are widely used in digital clocks, electronic meters, and calculators.

A 7-seg has seven connectors, one for each element, (plus few more connectors for other electrical purposes.) Each element can be turned on by sending an electric current through its pin. Each of the seven pins is viewed by programmers as a single bit in a 7-bit number, as they are more comfortable dealing with bits rather than electrical signals. The figure below shows the bit assignment for a typical 7-seg, bit 0 being the right-most bit.

For example, in order to display the digit 1, the programmer knows that only bits 1 and 3 need to be on, i.e. the 7-bit binary number to display digit 1 is "0001010", or 10 in decimal. Let's call the decimal number for displaying a digit, its display code, or just code for short. Since a 7-seg displays 127 different configurations, display codes are normally written using 3 decimal places with leading zeros if necessary, i.e. the display code for digit 1 is written as 010.
In a 9-digit calculator, 9 7-seg displays are stacked next to each other, and are all controlled by a single controller. The controller is sent a sequence of 3n digits, representing n display codes, where $0 < n < 10$. If $n < 9$, the number is right justified and leading zeros are automatically displayed. For example, the display code for 13 is 010079 while for 144 it is 010106106
Write a program that reads the display codes of two numbers, and prints the display code of their sum.

Input

Your program will be tested on one or more test cases. Each test case is specified on a single line in the form of A+B= where both A and B are display codes for decimal numbers a and b respectively where $0 < a$, $b < a + b < 1,000,000,000$. The last line of the input file is the word "BYE" (without the double quotes.)

Output

For each test case, print A+B=C where C is the display code for $a + b$.

Example

Input:
```
010079010+010079=
106010+010=
BYE
```

Output:
```
010079010+010079=010106106
106010+010=106093
```

Added by: Ahmed Aly
Date: 2009-07-03
Time limit: 1s
Source limit:50000B
Languages: All
Resource: ANARC 2008

SPOJ Problem Set (classical)

4551. Match Maker

Problem code: ANARC08C

In Computer Science, pattern matching is the act of checking if a certain sequence conforms (matches) a given pattern. Patterns are usually specified using a language based on regular expression. In this problem, we'll use a simple regular expression to express patterns on sequences of decimal digits. A pattern is a sequence of one or more decimal digits '0' ...'9', asterisks '*', and hash signs '#'. A '*' denotes a sequence of an even number of digits, whereas a '#' denotes a sequence of an odd number of digits. For example, the pattern "129" only matches the sequence 129. The pattern "1*3" matches all sequences beginning with 1, ending with 3, and having an even number of decimal digits between the first and last digits. As another example, the pattern "#55" matches the sequences 155, 12355, 1234555, but none of the sequences 55, 1255, 123455. Your task is to write a program to find if a given sequence matches a given pattern.

Input

Your program will be tested on one or more data sets. Each data set contains a single pattern and one or more sequences to match. The first line of each data set specifies the pattern, and the remaining lines specify the sequences to match against that pattern. The end of a data set (except the last) is identified by the word "END" (without the double quotes.) The end of the last data set is identified by the word "QUIT". All lines are 100,000 characters long or shorter.

Output

k.s. result
Where k is the test case number (starting at one,) and s is the sequence number (starting at one within each test case,) and result is either the word "match" if the given string matches the pattern, or the word "not" if it doesn't.

Example

Input:

```
129
1299
129
1129
END
1*3
123
1223
END
#55
155
12355
55
1255
QUIT
```

Output:

```
1.1. not
1.2. match
1.3. not
2.1. not
2.2. match
3.1. match
3.2. match
3.3. not
3.4. not
```

Added by: Ahmed Aly
Date: 2009-07-03
Time limit: 5s
Source limit: 50000B
Languages: All
Resource: ANARC 2008

SPOJ Problem Set (classical)

4552. Adding up Triangles

Problem code: ANARC08D

Take a look at the triangle on the left of the figure below. It is made of 9 (unit) triangles arranged in three rows (N = 3). Needless to say, a unit triangle is a triangle with N = 1 .

If you study the figure for few seconds, you'll realize that you can find 13 different triangles (which we'll call sub-triangles.) Of these 13 sub-triangles we have: Nine unit triangle; three with N = 2 , and one with N = 3 . The following table lists the number of sub-triangles in arrangements with N < 5 .

# of Rows:	N = 1	N = 2	N = 3	N = 4
# of Sub-triangles:	1	5	13	27

Let's define the value of a unit triangle to be the integer value written in that triangle. In general, the value of a triangle is the sum of values in all its unit triangles. The triangle on the right is the same as the other one but with the sub-triangle having the largest value being highlighted. Write a program to determine the sub-triangle with the largest value.

Input

Your program will be tested on one or more test cases. Each test case is specified in a single line made of integers (separated by spaces.) The first integer is the number of rows in the test case, and the remaining integers are the values of the unit triangles specified in a top-down, left-to-right order. (the first test case in the example below is the same as the one in the figure.) The last line of the input file contains the number 0 (which is not part of the test cases.)
The maximum number of rows is 400. The absolute value of a unit triangle is less than 1000.

Output

For each test case, print the result using the following format:
k. V
where k is the test case number (starting at 1,) is a single space, and V is the maximum value of a sub-triangle in that test case.

Example

```
Input:3 6 -24 0 12 -10 12 40 -4 6
4 1 1 -1 1 1 -1 1 -1 1 1 -1 1 -1 1 -1 1
0
Output:1. 54
2. 4
```

Added by:	Ahmed Aly
Date:	2009-07-03
Time limit:	5s
Source limit:	50000B
Languages:	All
Resource:	ANARC 2008

SPOJ Problem Set (classical)

4554. Relax! It is just a game

Problem code: ANARC08E

You: What's the score? Did I miss much?
Me: It's 2-1 for elAhli and the second half just started. The first half was quite boring.
You: Who scored first? elAhli or ezZamalek?
Me: What difference does it make?
You: Big difference! I can predict the outcome of the match if I knew the order of which goals were scored in the first half.
Me: What do you mean?
You: It's 2-1 for elAhli, right? One of three things could have happened: elAhli scored two goals then ezZamalek scored; Or, elAhli scored its first goal, then ezZamalek, then elAhli again; Or, ezZamalek scored first, then elAhli scored its two goals.
Me: So?!! I still don't understand what difference does that make? It's still 2-1 for elAhli! Why don't you just relax and let us continue watching the game in peace.
You: You don't understand!! I believe the probability of who'll win depends on the order of how goals were scored. Now I have to predict the outcome for 3 possibilities.
Me: And what if the score was 3-2? What would you have done then?
You: I would have to work for 5 different possibilities. No?
Me: Of course not! The number of possibilities isn't always equal to the sum.
You: Can you tell me when will it be equal to the sum?
Me: You're a programmer, why don't you write a program that counts the number of possibilities and compare it to the sum?
You: I don't have the time, I want to watch the match. Besides, I have nine other problems to worry about.
Me: I'll give you a hint. The possibilities will be equal to the sum only if one of the teams scored a certain number of goals.

Input

Your program will be tested on one or more test cases. Each test case specifies two natural numbers (A and B) (separated by one or more spaces) representing the score of the first half. No team will be able to score more than 10 goals. The last line of the input file contains two -1's (which is not part of the test cases.)

Output

Format For each test case where the number of possibilities is equal to the sum, print:
A+B=C
Where A and B are as above and C is their sum. If the number of possibilities is not equal to the sum, replace the '=' sign with '!=' (without the quotes.)

Example

Input:

```
2 1
1 0
-1 -1
```

Output:

```
2+1=3
1+0=1
```

Added by: Ahmed Aly
Date: 2009-07-04
Time limit: 1s
Source limit: 50000B
Languages: All
Resource: ANARC 2008

SPOJ Problem Set (classical)

4555. Einbahnstrasse

Problem code: ANARC08F

Einbahnstrasse (German for a one-way street) is a street on which vehicles should only move in one direction. One reason for having one-way streets is to facilitate a smoother flow of traffic through crowded areas. This is useful in city centers, especially old cities like Cairo and Damascus. Careful planning guarantees that you can get to any location starting from any point. Nevertheless, drivers must carefully plan their route in order to avoid prolonging their trip due to one-way streets. Experienced drivers know that there are multiple paths to travel between any two locations. Not only that, there might be multiple roads between the same two locations. Knowing the shortest way between any two locations is a must! This is even more important when driving vehicles that are hard to maneuver (garbage trucks, towing trucks, etc.)
You just started a new job at a car-towing company. The company has a number of towing trucks parked at the company's garage. A tow-truck lifts the front or back wheels of a broken car in order to pull it straight back to the company's garage. You receive calls from various parts of the city about broken cars that need to be towed. The cars have to be towed in the same order as you receive the calls. Your job is to advise the tow-truck drivers regarding the shortest way in order to collect all broken cars back in to the company's garage. At the end of the day, you have to report to the management the total distance traveled by the trucks.

Input

Your program will be tested on one or more test cases. The first line of each test case specifies three numbers (N , C , and R) separated by one or more spaces. The city has N locations with distinct names, including the company's garage. C is the number of broken cars. R is the number of roads in the city. Note that 0 < N < 100 , 0 <= C < 1000 , and R < 10000 . The second line is made of C + 1 words, the first being the location of the company's garage, and the rest being the locations of the broken cars. A location is a word made of 10 letters or less. Letter case is significant. After the second line, there will be exactly R lines, each describing a road. A road is described using one of these three formats:
A --v-> B
A <-v-- B
A <-v-> B
A and B are names of two different locations, while v is a positive integer (not exceeding 1000) denoting the length of the road. The first format specifies a one-way street from location A to B , the second specifies a one-way street from B to A , while the last specifies a two-way street between them. A , "the arrow", and B are separated by one or more spaces. The end of the test cases is specified with a line having three zeros (for N , C , and R .)
The test case in the example below is the same as the one in the figure.

[IMAGE]

Output

For each test case, print the total distance traveled using the following format:
k. V
Where k is test case number (starting at 1,) is a space, and V is the result.

Example

Input:

```
4 2 5
NewTroy Midvale Metrodale
NewTroy   <-20-> Midvale
Midvale   --50-> Bakerline
NewTroy    <-5-- Bakerline
Metrodale <-30-> NewTroy
Metrodale  --5-> Bakerline
0 0 0
```

Output:

```
1. 80
```

Added by: Ahmed Aly
Date: 2009-07-04
Time limit: 1s
Source limit: 50000B
Languages: All
Resource: ANARC 2008

SPOJ Problem Set (classical)

4556. Think I will Buy Me a Football Team

Problem code: ANARC08G

Falling Stocks. Bankrupted companies. Banks with no Cash. Seems like the best time to invest: "Think I'll buy me a football team!"
No seriously, I think I have the solution to at least the problem of cash in banks. Banks nowadays are all owing each other great amounts of money and no bank has enough cash to pay other banks' debts even though, on paper at least, they should have enough money to do so. Take for example the inter-bank loans shown in figure (a). The graph shows the amounts owed between four banks (A ...D). For example, A owes B 50M while, at the same time, B owes A 150M. (It is quite common for two banks to owe each other at the same time.) A total amount of 380M in cash is needed to settle all debts between the banks.
[IMAGE]
In an attempt to decrease the need for cash, and after studying the example carefully, I concluded that there's a lot of cash being transferred unnecessarily. Take a look:
1. C owes D the same amount as D owes A, so we can say that C owes A an amount of 30M and get D out of the picture.
2. But since A already owes C 100M, we can say that A owes C an amount of 70M.
3. Similarly, B owes A 100M only, (since A already owes B 50M.) This reduces the above graph to the one shown in figure (b) which reduces the needed cash amount to 190M (A reduction of 200M, or 53%.)
4. I can still do better. Rather than B paying A 100M and A paying 70M to C, B can pay 70M (out of A's 100M) directly to C. This reduces the graph to the one shown in figure (c). Banks can settle all their debts with only 120M in cash. A total reduction of 260M or 68%. Amazing!
I have data about inter-bank debts but I can't seem to be able to process it to obtain the minimum amount of cash needed to settle all the debts. Could you please write a program to do that?

Input

Your program will be tested on one or more test cases. Each test case is specified on N + 1 lines where N < 1, 000 is the number of banks and is specified on the first line. The remaining N lines specifies the inter-bank debts using an NXN adjacency matrix (with zero diagonal) specified in row-major order. The ith row specifies the amounts owed by the ith bank. Amounts are separated by one or more spaces. All amounts are less than 1000. The last line of the input file has a single 0.

Output

For each test case, print the result using the following format:
k. B A
where k is the test case number (starting at 1,) is a space character, B is the amount of cash needed before reduction and A is the amount of cash after reduction.

Example

Input:

```
4
  0  50 100   0
150   0  20   0
  0   0   0  30
 30   0   0   0
0
```

Output:

```
1. 380 120
```

Added by: Ahmed Aly
Date: 2009-07-04
Time limit: 1s
Source limit: 50000B
Languages: All
Resource: ANARC 2008

SPOJ Problem Set (classical)

4557. Musical Chairs

Problem code: ANARC08H

In the traditional game of Musical Chairs, N + 1 children run around N chairs (placed in a circle) as long as music is playing. The moment the music stops, children run and try to sit on an available chair. The child still standing leaves the game, a chair is removed, and the game continues with N children. The last child to sit is the winner.
In an attempt to create a similar game on these days' game consoles, you modify the game in the following manner: N Children are seated on N chairs arranged around a circle. The chairs are numbered from 1 to N . Your program pre-selects a positive number D . The program starts going in circles counting the children starting with the first chair. Once the count reaches D , that child leaves the game, removing his/her chair. The program starts counting again, beginning with the next chair in the circle. The last child remaining in the circle is the winner.

[IMAGE]

For example, consider the game illustrated in the figure above for N = 5 and D = 3 . In the figure, the dot indicates where counting starts and × indicates the child leaving. Starting off, child #3 leaves the game, and counting restarts with child #4. Child #1 is the second child to leave and counting restart with child #2 resulting in child #5 leaving. Child #2 is the last to leave, and child #4 is the winner.
Write a program to determine the winning child given both N and D .

Input

Your program will be tested on one or more test cases. Each test case specifies two positive integers N and D on a single line, separated by one or more spaces, where N, D < 1,000,000 .
The last line of the input file contains two 0's and is not part of the test cases.

Output

For each test case, write the winner using the following format:
N D W
Where N and D are as above, is a space character, and W is the winner of that game.

Example

Input:
```
5 3
7 4
0 0
```

Output:
```
5 3 4
7 4 2
```

Added by: Ahmed Aly
Date: 2009-07-04
Time limit: 15s
Source limit:50000B
Languages: All
Resource: ANARC 2008

SPOJ Problem Set (classical)

4558. I Speak Whales

Problem code: ANARC08I

According to Wikipedia, a Walsh matrix is a specific square matrix, with dimensions equal to a power of 2, the entries of which are +1 or -1, and the property that the dot product of any two distinct rows (or columns) is zero. Below are the first three Walsh Matrices. (The gray lines are imaginary lines for illustration purpose only.)

[IMAGE]

A Walsh Matrix of size 2^(N+1) can be constructed as the "union" of 4 Walsh Matrices of size 2^N arranged such that the lower right matrix is inverted whereas the other 3 matrices are not, i.e.:

[IMAGE]

Let’s number the rows of a given Walsh Matrix from the top starting with row 0. Similarly, let’s number the columns of the matrix from the left starting with column 0. Given the four integers N , R , S , and E , write a program that will construct a Walsh Matrix of size 2^N and will print the sum of all the numbers in row #R between columns #S and #E (inclusive.)

Input

Your program will be tested on one or more test cases. Each test case is specified using a single line listing four integers in the following order: N , R , S , and E , where 0 <= N <= 60 , 0 <= R < 2^N , 0 <= S <= E < 2N , and E - S <= 10,000 . The last line of the input file has four -1’s and is not part of the test cases.

Output

For each test case, print the output on a single line.

Example

Input:

```
2 1 0 1
48 0 0 47
-1 -1 -1 -1
```

Output:

```
0
48
```

Added by:	Ahmed Aly
Date:	2009-07-04
Time limit:	3s
Source limit:	50000B
Languages:	All
Resource:	ANARC 2008

SPOJ Problem Set (classical)

4559. A Day at the Races

Problem code: ANARC08J

Formula One is the highest class of car racing sports. A typical Formula One season consists of a series of races called "Grands Prix" which constructors like Ferrari, Renault, etc. and others participate with one or more cars driven by the best drivers in the world. During the season, teams compete in two parallel championships: the drivers championship and the teams championship.
In the drivers championship, drivers compete to achieve the maximum total number of points by the end of the season, the rules of the competition states that the top eight drivers at each Grand Prix receive 10,8,6,5,4,3,2,1 points respectively. In case of points tie, the driver with the highest number of first places leads. If still tied, then the highest second places, and so on till the highest 8th places. If still tied, then drivers are sorted lexicographically by their last and then by their first names.
After each race, the points received by each driver are added to his team's pocket, and at the end of the season the team with the highest number of points wins the teams championship. To add excitement to the season, team sponsors are allowed to buy drivers from other teams even within the same season. In case of points tie between teams, teams are sorted lexicographically by their names. In this problem, you are given data of a formula one season and you're asked to process these data according to the rules above to determine both the drivers and teams standings.

Input

Your program will be tested on one or more data-sets, each representing a Formula One season. All input lines are 255 characters or less. Studying the sample I/O you'll discover that the first line of each season has an integer N , where $0 < N < 32$ and representing the number of Grands Prix in that season. For each Grand Prix, the name of the Grand Prix appears on a line by itself (maximum length is 64 characters) followed by a table of the first name, last name and team name of the top eight drivers, from 1 to 8, in that Grand Prix. Each of the first and last names is a sequence of printable ASCII characters, no longer than 12 characters, and contains no spaces. Each team name is a sequence of printable ASCII characters, no longer than 18 characters, and may contain spaces (but no leading or trailing spaces.) Each team name is followed by a single period '.' which is not part of the name. Trailing white space may follow. A line of three -'s follows the listing of each Grand Prix. The last line of the input file contains a single zero.

Output

For each data set in the input you must print "Season k :" where k is the data-set number (starting from 1.) The next line must state "Drivers Standing:". On subsequent lines list the drivers standing for that season. For each driver, print their first and last names separated by exactly one space and left justified in a field of width 25, followed by a single space, followed by the total number of points achieved by the driver during the season. The drivers standing should be followed by a blank line.
The next line must state "Teams Standing:" On subsequent lines list the teams standing for the that season. For each team, print the team name left justified in a field of width 25, followed by a single space, followed by the total number of points the team has scored during the season. The teams standing should be followed by a blank line.

Example

Input:

```
2
FORMULA 1 Gran Premio Telefonica de Espana 2006
Pos  Driver                    Team
1    Fernando Alonso           Renault.
2    Michael Schumacher        Ferrari.
3    Giancarlo Fisichella      Renault.
4    Felipe Massa              Ferrari.
5    Kimi Raikkonen            McLaren-Mercedes.
6    Jenson Button             Honda.
7    Rubens Barrichello        Honda.
8    Nick Heidfeld             Sauber-BMW.
---
FORMULA 1 Grand Prix de Monaco 2006
Pos  Driver                    Team
1    Fernando Alonso           Renault.
2    Jaun-Pablo Montoya        McLaren-Mercedes.
3    David Coulthard           RBR-Ferrari.
4    Rubens Barrichello        Honda.
5    Michael Schumacher        Ferrari.
6    Giancarlo Fisichella      Renault.
7    Nick Heidfeld             Sauber-BMW.
8    Ralf Schumacher           Toyota.
---
0
```

Output:

```
Season 1:
Drivers Standing:
Fernando Alonso          20
Michael Schumacher       12
Giancarlo Fisichella     9
Jaun-Pablo Montoya       8
Rubens Barrichello       7
David Coulthard          6
Felipe Massa             5
Kimi Raikkonen           4
Jenson Button            3
Nick Heidfeld            3
Ralf Schumacher          1

Teams Standing:
Renault                  29
Ferrari                  17
McLaren-Mercedes         12
Honda                    10
RBR-Ferrari              6
Sauber-BMW               3
Toyota                   1
```

Added by: Ahmed Aly
Date: 2009-07-04
Time limit: 1s
Source limit:50000B
Languages: All
Resource: ANARC 2008

SPOJ Problem Set (acm)

4566. Đe^’m hinh chu+~ nhât

Problem code: LQDRECT

Cho môt ba?ng kích thu+o+’c MxN, đu+o+.c chia thanh lu+o+’i ô vuông đo+n vi. M dong N côt. (1<=M<=1000;1<=N<=300)
Tren các ô cu?a ba?ng ghi sô 0 hoăc 1. Các dong cu?a ba?ng đu+o+.c đánh sô 1, 2... M theo thu+’ tu+. tu+‘ tren xuông du+o+’i va các côt cu?a ba?ng đu+o+.c đánh sô 1, 2..., N theo thu+’ tu+. tu+‘ trái qua pha?i
Yeu câu:
Đe^’m sô hinh chu+~ nhât gôm các ô cu?a ba?ng thoa? man các đie^‘u kie^.n sau:
1 - Hinh chu+~ nhât đó có 4 ô o+? 4 đi?nh la 4 ô khác nhau
2- 4 ô o+? đi?nh đe^‘u la sô 1
3- Ca.nh hinh chu+~ nhât song song vo+’i ca.nh ba?ng

Input

Dong 1: Ghi hai sô M, N
M dong tie^’p theo, dong thu+’ i ghi N sô ma sô thu+’ j la sô ghi tren ô (i, j) cu?a ba?ng

Output

Gôm 1 dong duy nhât ghi sô hinh chu+~ nhât tho?a man yeu câu .
Example

```
Input:4 41 0 0 10 1 1 11 1 1 11 1 1 1Output:14Ke^’t qua? trong pha.m vi int64 cu?a pascal va long long cu?a C,C++
```

Added by: Ruan Shee Zhong
Date: 2009-07-05
Time limit: 1s
Source limit:50000B
Languages: All

SPOJ Problem Set (classical)

4574. Riding in cycles

Problem code: CYCLERUN

There are N cities, numbered from 1 to N,in the country you are living. **Each pair** of the cities is connected with exactly one road. However, **each road is a one-way road**, so it is either possible to godirectly from A to B or from B to A for each pair of cities (A, B).

You are living in city #1 and you are practicing for upcoming cycling marathon, so you want to construct the following training plan:

First day you have to ride over 3 roads starting from and finishing in city #1.

Second day you have to ride over 4 roads in the same manner.

Third day you have to ride over 5 roads.

...

The last, (N-2)-th, day you have to ride over N roads starting from and finishing in city #1.

You don't like to visit the same city more than once per day, so you have to find a training route for each day that passes through **each city at most once**. City #1 should appear only at the start and at the end of each route.

Write the program that, given the layout of the network, outputs training route for each day, or writes "impossible" if such training plan is not achievable.

Input

The first line of input contains the integer N (3 <= N <= 1000), number of cities.

Each of the next N lines contains exactly N characters that describes network layout. *j*-th character in*i*-th of these lines is '1' if it is possible to ride from city number *i* to city number *j*, or '0' otherwise.

Output

You should output training route for each day in a separate line. Training route consists of space separated integers - numbers of the cities in order they should be visited. Each training route starts and ends with 1.

If there is no achievable training plan, output word 'impossible' in a single line, instead.

Example

Input:

```
5
01000
00011
11001
10100
10010
```

Output:

```
1 2 5 1
1 2 4 3 1
1 2 4 3 5 1
```

Added by: Luka Kalinovcic
Date: 2009-07-07
Time limit: 1s
Source limit: 50000B
Languages: All
Resource: own problem

SPOJ Problem Set (classical)

4580. ABCDEF

Problem code: ABCDEF

You are given a set S of integers between -30000 and 30000 (inclusive).

Find the total number of sextuples that satisfy:

Input

The first line contains integer N (1 <= N <= 100), the size of a set S.

Elements of S are given in the next N lines, one integer per line. Given numbers will be distinct.

Output

Output the total number of plausiblesextuples.

Examples

Input:	Input:	Input:	Input:
1	2	2	3
1	2	-1	5
	3	1	7
Output:			10
1	Output:	Output:	
	4	24	Output:
			10

Added by: Luka Kalinovcic
Date: 2009-07-13
Time limit: 2s
Source limit:50000B
Languages: All
Resource: own problem

SPOJ Problem Set (classical)

4585. Star Wars

Problem code: GCJ08C

Near the planet Mars, in a faraway galaxy eerily similar to our own, there is a fight to the death between the imperial forces and the rebels. The rebel army has N ships which we will consider as points (xi, yi, zi). Each ship has a receiver with power pi. The rebel army needs to be able to send messages from the central cruiser to all the ships, but they are tight on finances, so they cannot afford a strong transmitter.
If the cruiser is placed at (x, y, z), and one of the other ships is at (xi, yi, zi) and has a receiver of power pi, then the power of the cruiser's transmitter needs to be at least:
(|xi - x| + |yi - y| + |zi - z|) / pi
Your task is to find the position for the cruiser that minimizes the power required for its transmitter, and to output that power.

Input

The first line of input gives the number of cases, T. T test cases follow.
Each test case contains on the first line the integer N, the number of ships in the test case.
N lines follow, each line containing four integer numbers xi, yi, zi and pi, separated by single spaces. These are the coordinates of the i-th ship, and the power of its receiver. There may be more than one ship at the same coordinates.
1 <= T <= 20
0 <= xi, yi, zi <= 10^6
1 <= pi <= 10^6
1 <= N <= 1000

Output

For each input case, you should output:
Case #X: Y
where X is the number of the test case and Y is the minimal power that is enough to reach all the fleet's ships. Answers with a relative or absolute error of at most 10-6 will be considered correct.

Example

Input:

```
3
4
0 0 0 1
1 2 0 1
3 4 0 1
2 1 0 1
1
1 1 1 1
3
1 0 0 1
2 1 1 4
```

```
3 2 3 2
```

Output:

```
Case #1: 3.500000
Case #2: 0.000000
Case #3: 2.333333
```

Added by: Ahmed Aly
Date: 2009-07-15
Time limit: 3s
Source limit:50000B
Languages: All
Resource: Google Code Jam 2008

SPOJ Problem Set (classical)

4586. Texas Trip

Problem code: WLOO0707

After a day trip with his friend Dick, Harry noticed a strange pattern of tiny holes in the door of his SUV. The local American Tire store sells fiberglass patching material only in square sheets. What is the smallest patch that Harry needs to fix his door?
Assume that the holes are points on the integer lattice in the plane. Your job is to find the area of the smallest square that will cover all the holes.

Input

The first line of input contains a single integer T expressed in decimal with no leading zeroes, denoting the number of test cases to follow. The subsequent lines of input describe the test cases.
Each test case begins with a single line, containing a single integer n expressed in decimal with no leading zeroes, the number of points to follow; each of the following n lines contains two integers x and y, both expressed in decimal with no leading zeroes, giving the coordinates of one of your points.
You are guaranteed that T <= 30 and that no data set contains more than 30 points. All points in each data set will be no more than 500 units away from (0,0).

Output

Print, on a single line with two decimal places of precision, the area of the smallest square containing all of your points. An answer will be accepted if it lies within 0.01 of the correct answer.

Example

Input:
```
2
4
-1 -1
1 -1
1 1
-1 1
4
10 1
10 -1
-10 1
-10 -1
```

Output:
```
4.00
242.00
```

Added by: Ahmed Aly
Date: 2009-07-15
Time limit: 3s
Source limit: 50000B
Languages: All
Resource: 14 July, 2007 - Waterloo local contest

SPOJ Problem Set (classical)

4587. Electric Fences

Problem code: FENCE3

Farmer John has decided to construct electric fences. He has fenced his fields into a number of bizarre shapes and now must find the optimal place to locate the electrical supply to each of the fences.
A single wire must run from some point on each and every fence to the source of electricity. Wires can run through other fences or across other wires. Wires can run at any angle. Wires can run from any point on a fence (i.e., the ends or anywhere in between) to the electrical supply.
Given the locations of all F (1 <= F <= 150) fences (fences are always parallel to a grid axis and run from one integer gridpoint to another, 0 <= X,Y <= 100), your program must calculate both the total length of wire required to connect every fence to the central source of electricity and also the optimal location for the electrical source.
The optimal location for the electrical source might be anywhere in Farmer John's field, not necessarily on a grid point.

Input

The first line contains F, the number of fences.
F subsequent lines each contain two X,Y pairs each of which denotes the endpoints of a fence.

Output

On a single line, print three space-separated floating point numbers, each with a single decimal place. Presume that your computer's output library will round the number correctly.
The three numbers are:
* the X value of the optimal location for the electricity,
* the Y value for the optimal location for the electricity, and
* the total (minimum) length of the wire required.

Example

Input:
```
3
0 0 0 1
2 0 2 1
0 3 2 3
```

Output:
```
1.0 1.6 3.7
```

Added by: Ahmed Aly
Date: 2009-07-15
Time limit: 1s
Source limit:50000B
Languages: All
Resource: USACO

SPOJ Problem Set (classical)

4588. SETI

Problem code: NWERC04H

For some years, quite a lot of work has been put into listening to electromagnetic radio signals received from space, in order to understand what civilizations in distant galaxies might be trying to tell us. One signal source that has been of particular interest to the scientists at Universit´e de Technologie Spatiale is the Nebula Stupidicus.
Recently, it was discovered that if each message is assumed to be transmitted as a sequence of integers a0, a1, ...an-1 the function f(k) = [IMAGE] (mod p) always evaluates to values 0 <= f(k) <= 26 for 1 <= k <= n, provided that the correct value of p is used. n is of course the length of the transmitted message, and the ai denote integers such that 0 <= ai < p. p is a prime number that is guaranteed to be larger than n as well as larger than 26. It is, however, known to never exceed 30 000.
These relationships altogether have been considered too peculiar for being pure coincidences, which calls for further investigation.
The linguists at the faculty of Langues et Cultures Extraterrestres transcribe these messages to strings in the English alphabet to make the messages easier to handle while trying to interpret their meanings. The transcription procedure simply assigns the letters a..z to the values 1..26 that f(k) might evaluate to, such that 1 = a, 2 = b etc. The value 0 is transcribed to '*' (an asterisk). While transcribing messages, the linguists simply loop from k = 1 to n, and append the character corresponding to the value of f(k) at the end of the string.
The backward transcription procedure, has however, turned out to be too complex for the linguists to handle by themselves. You are therefore assigned the task of writing a program that converts a set of strings to their corresponding Extra Terrestial number sequences.

Input

On the first line of the input there is a single positive integer N, telling the number of test cases to follow. Each case consists of one line containing the value of p to use during the transcription of the string, followed by the actual string to be transcribed. The only allowed characters in the string are the lower case letters 'a'..'z' and '*' (asterisk). No string will be longer than 70 characters.

Output

For each transcribed string, output a line with the corresponding list of integers, separated by space, with each integer given in the order of ascending values of i.

Example

Input:

```
3
31 aaa
37 abc
29 hello*earth
```

Output:

```
1 0 0
0 1 0
8 13 9 13 4 27 18 10 12 24 15
```

Added by: Ahmed Aly
Date: 2009-07-15
Time limit: 2s
Source limit: 50000B
Languages: All
Resource: NWERC 2004

SPOJ Problem Set (classical)

4644. Proving Equivalences

Problem code: PMATRIX

Consider the following exercise, found in a generic linear algebra textbook.Let A be an n × n matrix. Prove that the following statements are equivalent:

- (a) A is invertible.
- (b) Ax = b has exactly one solution for every n × 1 matrix b.
- (c) Ax = b is consistent for every n × 1 matrix b.
- (d) Ax = 0 has only the trivial solution x = 0.

The typical way to solve such an exercise is to show a series of implications. For instance,one can proceed by showing that (a) implies (b), that (b) implies (c), that (c) implies (d),and finally that (d) implies (a). These four implications show that the four statements are equivalent. Another way would be to show that (a) is equivalent to (b) (by proving that (a) implies (b) and that (b) implies (a)), that (b) is equivalent to (c), and that (c) is equivalent to (d).

However, this way requires proving six implications, which is clearly a lot more work than just proving four implications! I have been given some similar tasks, and have already started proving some implications. Now I wonder, how many more implications do I have to prove? Can you help me determine this ?

Input

On the first line one positive number: the number of testcases, at most 100. After that per testcase:

- One line containing two integers n (1 <= n <= 20 000) and m (0 <= m <= 50 000): the number of statements and the number of implications that have already been proved.
- m lines with two integers s1 and s2 (1 <= s1, s2 <= n and s1 = s2) each, indicating that it has been proved that statement s1 implies statement s2.

Output

Per testcase:
One line with the minimum number of additional implications that need to be proved in order to prove that all statements are equivalent.

Example

Input:
```
24
03
21
21
3
```
Output:
```
42
```

Added by: ?????
Date: 2009-07-25
Time limit: 1s
Source limit:50000B
Languages: All
Resource: NWERC 2008

SPOJ Problem Set (classical)

4656. Cross Mountain Climb

Problem code: CCROSS

Somewhere in the neighborhood we have a very nice mountain that gives a splendid view over the surrounding area. There is one problem though: climbing this mountain is very difficult, because of rather large height differences. To make more people able to climb the mountain and enjoy the view, we would like to make the climb easier. To do so, we will model the mountain as follows: the mountain consists of n adjacent stacks of stones, and each of the stacks is hi high. The successive height differences are therefore $h_{i+1} - h_i$ (for 1 <= i <= n - 1). We would like all absolute values of these height differences to be smaller than or equal to some number d.

We can do this by increasing or decreasing the height of some of the stacks. The first stack (the starting point) and the last stack (the ending point) should remain at the same height as they are initially. Since adding and removing stones requires a lot of effort, we would like to minimize the total number of added stones plus the total number of removed stones. What is this minimum number?

Input

On the first line one positive number: the number of testcases, at most 100. After that per testcase:

- One line with two integers n (2 <= n <= 100) and d (0 <= d <= 10^9): the number of stacks of stones and the maximum allowed height difference.
- One line with n integers hi (0 <= h_i <= 10^9): the heights of the stacks.

Output

Per testcase:
One line with the minimum number of stones that have to be added or removed or "impossible" if it is impossible to achieve the goal.

Example

Input:
```
3
10 24 5
10 6 6 9 4 7 9 8
3 16 4
0
4 23 0
6 3
```
Output:
```
6
impossible
4
```

Problem text...

Problem text...

Added by: ????? [FameofLight]
Date: 2009-07-27
Time limit: 15s
Source limit:50000B
Languages: All
Resource: NWERC 2008 Regionals

SPOJ Problem Set (classical)

4657. Gas Wars

Problem code: GASWARS

As the result of the gas wars the following agreement was made. The transit of the gas was allowed under the following conditions. There are **n** transit nodes and **m** pipes connecting those nodes. There **k** nodes where the gas enters and **l** nodes where it should be moved. Each pipe has a carrying capacity of $\mathbf{c_i}$ cubic meters of gas per day. Gas can go through the pipes in either direction. It is needed to move **g** cubic meters of gas in total through the pipeline every day. The cost of the transit is defined as **maxC***100 thousand dollars, where **maxC** - maximum of carrying capacities of the used in the transit pipes (even those which are not fully used). You are to find the minimum possible cost of transit for the given pipeline.

Input

The first line of the input file contains **t** - the amount of test cases. The description of each test case follows. The first line of each test case contains five integers separated by spaces - **n**, **m**, **k**, **l**, **g**. Then **m** lines containing three integers **a**, **b**, **c** follow. Each lines means that nodes with numbers **a** and **b** are connected by the pipe with the carrying capacity of **c**. Next line contains **k** integers - the numbers of nodes where the gas should enter the pipeline. The last line of the test case contains **l** integers - the numbers of nodes where the gas should be moved. The gas can enter the pipeline in any of the **k** entrance nodes and can be moved to any of the **l** exit nodes. The nodes are numbered from 1 to **n**.

Constraints

1 <= **t** <= 20
2 <= **n** <= 100
1 <= **m** <= **n***(**n**-1)/2
1 <= **k**, **l** <= **n**/2
1 <= **g**, **c** <= 1000000

Output

For each test case output a single integer on a separate line - the minimum cost of transit in thousands of dollars. If the transit of the needed volume is impossible, then output -1.

Example

Input:

```
1
6 8 1 1 1
1 2 1
1 3 2
2 4 3
2 5 3
3 4 4
3 5 2
4 6 4
```

```
5 6 1
1
6
```

Output:

```
200
```

Added by: Spooky
Date: 2009-07-28
Time limit: 4s
Source limit: 50000B
Languages: All
Resource: Advancement Spring 2009, http://sevolymp.uuuq.com/

SPOJ Problem Set (classical)

4658. Help Hemant Verma

Problem code: HHEMANT

Steganography is a method of cryptography where a message or entire document can be hidden inside of another file or image which shows no evidence that there is data hidden in it. Typically, the message or document to be sent is first encrypted and compressed, and then combined with an existing file in the bits that are less significant.
Hemant Verma is a secret agent who wants to send some classified information to his head office , the amount of data is very large , he needs your help to encode the message into image . You will be given an "image" and you will encode a given message into it and return the new image. The returned image should be in the same format as the original image.
The image will be in the format of a string of various lines where each three digits represent a number from 0 to 255, inclusive (leading zeros will be added as necessary), which is a pixel value in the image.You will also be given a message which contains the message you would like to encode into the image.
You will first encode the message into numbers representing the characters in the message - spaces will be 0, 'A'-'Z' will be 1-26, 'a'-'z' will be 27-52, '0'-'9' will be 53-62, and 63 will be used for any space after the message. All these numbers can be represented in binary with 6 digits. You will put each pair of bits (representing a number between 0 and 3) into the lowest two bits of the values in the image. For each character, you will put in the lowest two bits, then the middle two, then the highest two, and then continue to the next character. You will put them in the lowest two bits of the first pixel on the first row, then the second pixel on the first row, and so on until you get to the end of the first row, then the first pixel on the second row, and so on. Once you are out of characters, continue substituting the lowest two bits of each pixel value as if the current character were represented by number 63.

Input

First line contain T representing number of test case , for each test case first line contain message then follow N the number of lines for image then following N line contain image.

$$0<\text{Length(message)}<1000$$
$$0 < T,N < 10000$$

Output

For each test case output the image in same format as input.

Example

Input:2hi125512321200120122 2hi225512321200120122 2**Output:**254120214003200222254120214003200222

Added by:	????? [FameofLight]
Date:	2009-07-28
Time limit:	0.5s
Source limit:	50000B
Languages:	All
Resource:	Modified Topcoder Problem