Archives of the Sphere Online Judge

challenge problemset

Editors:

Roman Sol
Michał Małafiejski
Adrian Kosowski
.:: Debanjan ::.
Jin Bin
[Trichromatic] XilinX
HWK
XeRon!X
Maciej Boniecki
cegprakash
Spooky
Ibrahim Mesecan
Gogu Marian
Łukasz Kuszner
? [MUSIC] [MUSIC] [MUSIC] [MUSIC]
kawmia institutes problem setters
Race with time
Ruslan Sennov
Krzysztof Kluczek
Ngô Minh Đu+'c
Vimal
Michael Suchacz
Konrad Piwakowski
Srivatsan B
Yandry Pérez
Clemente
Jargon
Tony Beta
Lambda
Thanh-Vy Hua
Alfonso2 Peterssen
Michael Mudge
Adam Dzedzej
Pawel Gawrychowski
???
Maxim Sukhov

Last updated: 2011-05-24 23:07:09

Preface

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

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

Table of Contents

46. Problem BRAINF_K (1742. Brainf_ck)
47. Problem EXPR2 (2004. Analyse Simple Arithmetical Expressions)
48. Problem ERRORMIN (2335. Error Minimization)
49. Problem MAGIC2 (2471. Magic Program II)
50. Problem JAWB (2624. JawBreaker Game)
51. Problem MAR (2628. Markov Algorithm)
52. Problem EYES (2629. Robo Eye)
53. Problem CARL (2659. Carl)
54. Problem SPOJTEST (2824. Glenbow Museum)
55. Problem SELF (3099. Super Quine)
56. Problem NOP (3880. Nop)
57. Problem BFWRITE (3947. Brainf F##k Writing)
58. Problem PLCNUM2 (4246. Place the Numbers II)
59. Problem DDATE (6125. Dos Date)
60. Problem DBP (6173. Burned Pancakes Tower)
61. Problem SQDIFF (6295. Area Difference)
62. Problem MONS (6338. Monster)
63. Problem BRACKETS (6646. Fully Parenthesized Expression)
64. Problem LQP (6802. Budget)
65. Problem DEC_BIN (6877. Terminating or Non-Terminating)
66. Problem PARKBUS (6954. Parking Busses)
67. Problem FINDMAX3 (6955. Find the max of the 3 numbers)
68. Problem CYCLE (7063. Cycles, More Cycles)
69. Problem REVINPUT (7105. Reverse the Input)
70. Problem WORDNUM (7225. Word To Number)
71. Problem MB2 (7289. Reverse text)
72. Problem MB1CH (7334. PP numbers)
73. Problem NPRIME (7480. Nth Prime)
74. Problem PPROF (7513. Painful Prof)
75. Problem BENDA (7958. Prisoner of Benda (Challenge))
76. Problem DIE_PIP (7965. The Electronic Dice)
77. Problem RCLUSTER (8101. Rectangular Cluster)
78. Problem NUMWORD (8315. Number to Word)
79. Problem LINC (8399. Linear Congruences)
80. Problem PRINTF (8405. Printf Parser)
81. Problem WAYS (8473. PATHS)
82. Problem PL_QUINE (8548. Brainf_cked!)
83. Problem MODULUS2 (8579. BF_MODULUS)
84. Problem FIBSUM (8638. Sum of Fibonacci numbers)
85. Problem BALLS (8672. Distributing the balls)
86. Problem EQCHECK (8675. Equation Check)
87. Problem PHIVAL (8676. Digits of Phi (Golden Ratio))
88. Problem PLDGTS (8733. Summing up Last digits)
89. Problem GRIDPNT (8950. Grid points)

SPOJ Problem Set (challenge)

53. Kamil

Problem code: KAMIL

Some kids cannot pronounce all letters, some of them they sometimes pronounce correctly and sometimes incorrectly. Kamil sometimes says T instead of K, but he never says K instead of T. Similarly he sometimes says D instead of G. Instead of R he sometimes says L and sometimes F. Of course it happens that he pronounces the letter correctly. Kamil's father always thinks how many words can mean the word spoken by his son (it doesn't matter if they are real English words).

Task

Write a program which

- reads from standard input the words spoken by Kamil
- counts how many different words can that mean
- writes the outcome on standard output

Input

Ten test cases (given one under another, you have to process all!). Every test case is a single line - a word spoken by Kamil. Only 26 capital leters are used. The length of the word is at most 20.

Output

For every testcase write an integer in a single line with a single integer, denoting the number of words which Kamil's word can mean.

Score

The score awarded to your program is the number of bytes the source code you submit. The fewer points you score, the better. Submissions are not allowed to exceed 256 bytes.

Remark. It may turn out impossible to solve this problem in some languages.

Example

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

Added by:	Adam Dzedzej
Date:	2004-06-08
Time limit:	3s
Source limit:	256B
Languages:	All except: TECS
Resource:	Internet Contest Pogromcy Algorytmow (Algorithm Tamers) Round I, 2003

SPOJ Problem Set (main)

72. Food Shortage in Byteland

Problem code: BYTEFOOD

Fanatics from the BBFO blew up all the food factories in the Bytelandian capital! Hurry up! There is still some food left in shops. Some shops are located in the centre, others in the suburbs, so Johnny has to decide which of them are worth visiting. Some shops can be very big and have plenty of food in them, others may be so small that food dissappears from them at an alarming rate... So? Help Johnny buy as much food as possible.
There are *n* open shops, each of them located at position (x_i, y_i), for i=1,...,n, where $0 <= x_i, y_i <= 250$. The distances between shops are measured using the Manhattan metric (i.e. as sums of absolute values of differences of x and y coordinates). Besides, every shop is characterized by a linear time function describing how much food is left in the shop at the moment:

$$f_i = \mathbf{max}\{0, a_i - b_i * time\}$$

where $0 <= a_i <= 1000000$, $0 <= b_i <= 1000$, while *time* is the time (in minutes) that has elapsed from the moment Johnny left the house (assume that Johnny does not live in the same place as any shop). If Johnny decides to stay in a shop, he can buy at most b_i units of food per minute. Otherwise, he can move along the ortogonal system of streets of the city at a constant speed of unit distance per minute. Johnny only ever changes the action he is performing at the full minute. Because his family is slowly beginning to starve, he should be back at home not later than *m* minutes after he left. Since there are thousands of starving families in the capital, Johnny can't spent more that $1 <= c_i <= 10$ minutes in a shop. Moreover, he will never go into the same shop twice for fear of being lynched...

Input

The first line of input contains a single positive integer $t <= 1000$, the number of test cases. Each test case begins with the number of shops in the city $1 <= n <= 1000$ and the deadline $1 <= m <= 5000$. Then the following *n* lines consist of four integers x_i y_i a_i b_i c_i each, describing the position and the parameters of the function for food availability of the i-th shop. At the end of every test case comes a line with two integers*p q* (between 0 and 250), corresponding to the x and y coordinates of the position of Johnny's house.
All the input data are integers.

Output

Process all test cases. The correct output for the i-th test case takes the following form:
i [the number of the test case, in the input order]
s m [*s* is the number of the target shop and $m > 0$ is the number of minutes spent in it].
At the end of the series of moves you should always write a line conisting of two zeros ('0 0').
All the output data should be integers.

Scoring

The score of your program is the total amount of food that Johnny bought (summed over all the testcases in which he managed to come back home before the deadline).

Example

Input
```
4
2 20
0 0 100 5 5
10 0 200 10 10
5 0
2 20
0 0 180 15 10
10 0 200 20 10
5 0
4 101
0 0 1000 20 5
20 0 200 1 5
0 20 5000 200 5
20 20 300 5 10
10 10
1 15
1 0 10 1 5
5 0
```

Output
```
1
2 10
0 0
2
1 10
0 0
3
3 5
4 10
2 1
0 0
4
1 5
0 0
```

Score
```
Score = 1261
```

Added by:	Michał Małafiejski
Date:	2004-06-09
Time limit:	17s
Source limit:	50000B
Languages:	All
Resource:	-

SPOJ Problem Set (challenge)

126. Solovay-Strassen Inverted

Problem code: SOLSTRAS

Let us denote the set of all prime numbers by the symbol **P**. The Solovay-Strassen algorithm determines whether a given positive odd integer n>2 belongs to **P**.
The *Legendre function* sig for number $n$in **N** with parameter $s$in **N** (s<n) is defined by the formula sig(n,s)=$s^{(n\text{-}1)/2}$ mod n. The symbol mod is defined in such a way as to return the result with the smallest possible absolute value, from the range (-n/2, n/2].
The *Jacobi function* jac for number $n$in **N** with parameter $s$in **N** (s<n) is given as:

$$\mathrm{jac}(n,s)=\text{cases}\begin{matrix} \mathrm{sig}(n,s), & \text{if } n\text{in } \mathbf{P} \\ \underset{i=1}{\overset{k}{\text{product}}}\mathrm{sig}(p_i,s), & \text{if } n=\underset{i=1}{\overset{k}{\text{product}}}p_i, \text{where all } p_i \text{ in } \mathbf{P} \end{matrix}$$

It is interesting to note that for given n and s, the values of sig(n,s) and jac(n,s) can be computed in $O((\log_2 n)^2)$ time. For particulars consult an encyclopedia, such as MathWorld.

The deterministic version of the Solovay-Strassen primality-test algorithm is given below.

```
algorithm Solovay-Strassen (n)
var s;
begin
 for s in {1,2,3,4,...,n} do
 if sig(n,s)neq jac(n,s)
 then return "n is composite (detected at attempt <s>)";
 return "n is prime";
end.
```

Task

We are not asking you to implement the Solovay-Strassen algorithm, this would be far too easy :). Instead, try to find values of n, for which the output of the algorithm would be "n is composite (detected at attempt 1)", "n is composite (detected at attempt 2)", and so on. Write out as many of these values as you can in consecutive lines, and try to keep them as small as possible.

Scoring

The score awarded to your program is the total of all points given for its individual lines.

The i-th line output by your program should contain exactly one positive odd integer $n>2$ of no more than 500 decimal digits. If Solovay-Strassen(n) yields the answer "n is composite (detected at attempt i)", you will receive $i/\log_{10} n$ points for this line, if not - your program will be considered incorrect. Output 0 if you don't want a line to be assessed. Only the first 1000 lines of output are taken into account.

Example

A program outputing:

```
0
0
561
```

will receive 3/$\log_{10}$ 561 = 1.091 points.

Added by:	Konrad Piwakowski
Date:	2004-07-10
Time limit:	3s
Source limit:	50000B
Languages:	All
Resource:	Thanks to Daniel Grzonkowski for valuable comments

SPOJ Problem Set (challenge)

127. Johnny Goes Shopping

Problem code: JOHNNY

Johnny visited his favourite supermarket to purchase as many sweets as he could afford. Since daddy had left his credit card at home untended, this was not really a problem. Once he had (barely) managed to push the trolley laden with chocolate bars past the cash desk, he began to wonder how to carry all the shopping home without breaking his back.

You must know that Johnny is a perfectly normal child, and has exactly 2 hands. Help him distribute his load between both hands so as to minimise the difference in load between both hands.

Input

The first line of input contains a single integer n<= 10000 denoting the number of sweet packets Johnny has bought. In each of the next n lines a single positive integer is given, describing the weight of the respective packet (the weight is an integer value never exceeding 10^{14}).

Output

In separate lines, output the numbers of the packets which Johnny should carry in his left hand. Assume packets are numbered in the input order from 1 to n.

Scoring

Your program shall receive $\log_{10}(s/(|d|+1))$ points, where s is the total weight of all parcels, while d denotes the difference in weight between the load carried in Johnny's left and right hand.

Example

For the sample input:

```
3
5
8
4
```

a program outputting:

```
2
3
```

will score $\log_{10}((5+8+4)/(|8+4-5|+1))$= 0.327 points.

Added by:	Adrian Kosowski
Date:	2004-07-10
Time limit:	5s
Source limit:	50000B
Languages:	All

SPOJ Problem Set (challenge)

155. Solving the Puzzle

Problem code: SOLVING

The 15 puzzle is a classic puzzle made famous in the 19th century. It consists of 4x4 board with 15 sliding tiles numbered from 1 to 15. The objective is to get them into this pattern:

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	

Here we will deal with a generalized version of the above puzzle. You should write a program that given some initial state of the nxn board finds a sequence of moves that transforms it so that in the i-th row there are tiles with numbers i*n+1,i*n+2,...,i*n+n (from left to right) - with the exception of the lower right corner where the hole should be. The less moves you use, the more points you get.

Input

The first line of input contains the number of test cases c (c<=200). Then c test cases follow, each of them begins with a line with a single integer n (3<=n<=10) in it. The next n lines describe the initial state of the board - the i-th line consists of exactly n integers describing the i-th row. The position of the hole is indicated by 0.

Output

For each test case output one line - the found sequence of moves. Write 'D' to move the hole down, 'U' to move it up, 'R' to move it right and 'L' to move it left. You shouldn't use more than 10000 moves. All moves should be valid (so for example don't try to move the hole up when it is in the first row).

Scoring

Your program will receive n^3/(m+1) points for each test case where m is the number of moves.

Example

```
Input:
2

4
1   2  7  3
5   6  0  4
9  10 11  8
```

```
13 14 15  12

3
0 1 2
4 5 3
7 8 6

Output:
URDDD
RRDD
```

Added by: Pawel Gawrychowski
Date: 2004-07-27
Time limit: 2s
Source limit:10000B
Languages: All

SPOJ Problem Set (challenge)

218. Points on a Sphere

Problem code: PSPHERE

Imagine a number of identically charged weightless dimensionless particles placed on the surface of a ball. They will instantly reach a state of equilibrium (a stable state of minimum energy), becoming distributed fairly evenly all round the sphere.

You probably won't be surprised to hear that Byteland has a sadly distorted electrostatic field, and the energy of the system is not governed by ordinary laws. Instead, it is inversely proportional to the distance between the closest pair of charges on the sphere.

Please help the charges find positions in which they will feel as comfortable as possible. Charges should be regarded as points in 3D space, located on the surface of the unitary sphere (with center (0,0,0) and a radius of 1).

Input

An integer t denoting the number of test cases (t<=10), followed by t test cases, each consisting of a line with a single integer n - the number of points on the sphere (2<=n<=1000).

Output

For each test case, output n lines consisting of three floating point numbers, corresponding to the x y z coordinates of successive points.

Scoring

The score of your program is the total of scores awarded for individual test cases.

For each test case you will receive n*d points, where d denotes the minimum distance between the closest pair of points in the solution output by your program. If you place some of the points further than 10^{-5} from the surface of the sphere, your solution will be regarded as incorrect.

Example

For the sample input:
1
2
a program outputting:
0.0 0.0 1.0
0.0 1.0 0.0
will receive 2.828 points.

Added by:	Adrian Kosowski
Date:	2004-10-09
Time limit:	5s
Source limit:	50000B
Languages:	All
Resource:	DASM Programming League 2004 (problemset 1)

SPOJ Problem Set (challenge)

219. Pawns Gone Wild

Problem code: PAWNS

Imagine a game played on an *n x n* chessboard by two players sitting at opposite ends, one having n white pawns, the other - n black pawns. Pawns are arranged in the row closest to the player. Moves are made in turn by both players and resemble those in chess: in a single move, a player can move exactly one pawn, either a square forward (if the square it is moving onto is free), or on the bias, one square forward and one square to the left or right (if the square it is moving onto is occupied by an enemy's pawn, which is considered beaten and removed from the game).

[IMAGE]

Pawns may never be moved backwards or off the board, and if a pawn reaches the final line it just has to stay there. The game ends if a player can't make a move. The winner is then the player who... oh, it doesn't matter really (possibly the players have a fight with beer bottles, and the one who isn't knocked out, wins). Your task is different - seeing snapshots of a game at two moments of time, try to reenact a sequence of moves that may have led from the first situation to the second.

Input

The first line of input contains a single positive integer t<=25, the number of test cases. t test cases (of successively increasing size) follow.

Each test case begins with an integer n (2<=n<=26) denoting the size of the board. Then, the snapshot of the earlier situation is given, followed by a snapshot of the later situation. Each snapshot is a sequence of n lines of n characters, corresponding to the squares of a chessboard oriented as in the figure above. Character '.' - denotes an empty square, 'W' - a square with a white pawn, 'B' - a square with a black pawn. Assume that it is black's turn to move after the earlier position (though black needn't have necessarily started the game as such).

Output

For each test case, output the number k of moves which could have led from the first to the second position (output 0 if you don't know a solution, even though such a solution exists for certain). In the next k lines, print the determined sequence of moves. Each move should be given in a seperate line, using the format: `old_column old_row new_column` to describe the change of coordinates of a pawn (assume board orientation as in the exemplary figures).

Scoring

The score of your program is the total of scores awarded for individual test cases. For each test case for which you find a solution in k moves you will receive k points.

Example

For the sample input:

```
1
6

....W.
.WWW.W
B.B...
......
......
B..BBB

......
.BWW..
.....W
......
.....B
B..BB.
```

a program outputting:

```
5
A 3 B
F 2 F
C 3 D
E 1 D
F 6 F
```

will receive 5 points.

Added by:	Adrian Kosowski
Date:	2004-10-10
Time limit:	21s
Source limit:	50000B
Languages:	All
Resource:	DASM Programming League 2004 (problemset 1)

SPOJ Problem Set (challenge)

222. The Burning City

Problem code: BURNCITY

Terrorists from the BBFO have raised fires in the capital of Byteland! As is it is a hot summer day, most of the fire brigade have quite naturally taken a day off, and so the noble task of extinguishing all the fires falls to the only officer on duty. By now you will probably not be surprised to learn that he is in fact... Johnny. This enterprising youth remains undaunted by the challenge facing him, and, taking advantage of the absence of his superiors, he decides to use his favourite fire fighting technique. So, he loads the fire station's helicopter with as many dynamite charges as it can carry, and takes off on his errand of mercy.

From up there in the sky Johnny can see the city as a square, sliced into smaller, identical squares by a regular grid of streets. Every square contains one of three kinds of terrain - buildings, grassland or water (perhaps most firemen would go into further detail when analysing terrain, but you really can't expect that from a firefighter whose preferred method of extinguishing fires is dynamite, can you?).

Johnny starts out in the centre of the square corresponding to the fire station. In the time from the start of a minute to the end of that minute he can move to the center of one of the four adjacent squares (but he is not allowed to leave the city). While over the center of a square he can choose to drop a single dynamite charge on it. He starts preparing the charge at the beginning of a minute, and it is dropped from the helicopter at the end of the same minute. Everything on the square on which the bomb was dropped is blown apart, and in its place a crater is formed and instantly flooded by subterranean waters.

The fire spreads in a most predictable way: if a square starts burning at the beginning of minute m, then all four adjacent squares will catch fire at the start of minute (m+2). The only exception is a square filled with water (either naturally, or by Johnny's bombs) which never catches fire. If a square starts burning, all property on it is instantly destroyed. Once a square starts burning it will only stop burning if Johnny blows it up, or when the monsoon rain comes and floods the city, at the end of the h-th minute of firefighting.

Johnny's main objective is to save as many squares with buildings as possible (from fire and dynamite).

An example of the fire fighting process is presented below.

An example of the fire fighting process

Input

The first line of input contains a single integer t<=500, the number of test cases.

The first line of every test case contains five integers n c h s_x s_y, respectively denoting: n - the length of one side of the city (measured in squares), c - the number of dynamite charges Johnny can use, h - the number of minutes after which the rain falls, s_x, s_y - the x and y coordinates of the square containing the fire-station from which Johnny starts, measured relative to the North-West corner of the city (1<=s_x, s_y<=*n*<=50, 0<=*c*<=*h*<=5**n*; there are about 10 test cases for all possible values of *n*). Finally, the map of the city is given in the form of *n* lines of *n* characters each, each corresponding to the state of a square at the start of the fire fighting ('b' - building, 'g' - grassland, 'w' - water, 'f' - fire).

Output

For the *i*-th test case output a line containing the text 'city *i* Y' if you want to solve the test case or 'city *i* N' if you wish to leave it out.

If you chose to the solve the test case, in the next line output a sequence of exactly *h* characters 'N', 'S', 'W', 'E', '+' or '-', corresponding to Johnny's actions in successive minutes (moving North, South, West and East on the map, dropping dynamite, not doing anything, respectively).

Score

The total score is the total number of rescued squares with buildings taken over all test cases.

Example

Input:
```
5
3 2 9 1 1
bgg
bbg
bbf
4 2 8 3 1
bbbb
bgwg
fwgg
gbbb
4 3 15 2 1
bbbb
bbbb
bbbb
fbbb
4 3 15 2 1
bbbf
bbbb
bbbb
fbbb
4 3 15 2 1
bbbf
bbbb
bbbb
fbbb
```

Output:
```
city 1 Y
S+E---N+-
city 2 Y
W+SSS+--
city 3 Y
ESE+SW+S+------
city 4 Y
+EES+W+--------
city 5 Y
```

```
+ES+-E+--------
```

Score:

```
9
```

(The first test case is illustrated in the figure and Johnny can save one building. In testcases 2, 3, 4, 5 Johnny saves 4, 2, 2 and 0 buildings, respectively).

Bonus info: The three digit number after the decimal point of your score denotes the number of test cases you have solved correctly, rescuing at least one building.

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

---

Added by:	Michał Małafiejski
Date:	2004-10-11
Time limit:	17s
Source limit:	50000B
Languages:	All
Resource:	-

SPOJ Problem Set (challenge)

225. Nightmare in the Towers of Hanoi

Problem code: HANOI

Consider the folowing variation of the well know problem Towers of Hanoi:

We are given *n* towers and *m* disks of sizes 1,2,3,...,*m* stacked on some towers. Your objective is to transfer all the disks to the *k*-th tower in as few moves as you can manage, but taking into account the following rules:

- moving only one disk at a time,
- never moving a larger disk one onto a smaller one,
- moving only between towers at distance at most *d*.

You can assume that all the problems can be solved in not more than 20000 moves.

Input

The first line of input contains a single positive integer $t <= 1000$, the number of test cases.

Each tests case begins with the number of towers $3 <= n <= 100$, the number of target tower $1 <= k <= n$, the number of disks $m <= 100$ and the maximum distance $1 <= d <= n - 1$.
Then, the following *m* lines consists of pairs of numbers describing the initial situation, in the form: the tower and disk on it. Assume according to the rules that on every tower smaller disks are on larger disks.

Output

Process all test cases. The correct output for the *i*-th test case takes the following form:
i [the number of the test case (in input order)]
a b [a sequence of lines of this form, where *a* is the tower with the moved disk on top of it and *b* is the target tower].
The test case is considered solved if after performing the sequence all disks are on the *k*-th tower. At the end of the series of moves you should always write a line consisting of two zeros ('0 0').

Scoring

The score awarded to your program is the sum of scores for individual test cases. For the i-th test case you will receive T_i / (T_i + A_i) points, where T_i <= 20000 and A_i is the number of moves in your solution. If you don't want to solve a test case, you may output the line '0 0' without a list of moves, for which you will not be awarded any points. Your program may not write more than 30000 kB to output (this will result in SIGXFSZ).

Example

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

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

Score
Assuming: T = {7,6,15,7,1} the output will receive **2** points.

Bonus info: If score = *xxx.xxxaaa*, *aaa* means the number of test cases with non-zero score...

Added by:	Michał Małafiejski
Date:	2004-10-27
Time limit:	17s
Source limit:	50000B
Languages:	All
Resource:	-

SPOJ Problem Set (challenge)

232. Bytelandian Telecom

Problem code: BYTELE

King Johnny has a serious drink problem, which has recently become the focus of attention of all Bytelandian tabloids and colour magazines. In a desperate effort to divert the public's attention and ingratiate himself with his subjects, he decides to start giving out valuable gifts. This time he has chosen to harass the peaceful life of the CEO of Bytelandian Telecom, and requested him to create a Metropolitan Area Network for the citizens of the capital of Byteland, as part of an "our King is a good man" campaign.

The CEO has no choice but to obey the orders he receives. This rational and business-minded man would obviously like to perform the installation at the smallest possible cost, and he asks you for your help.

The King has stated the topology of the network plainly enough in the form of a graph (not necessarily connected), with vertices corresponding to nodes (computers), and edges to the cable connections between them. It is now your task to select the points of the city to place the nodes of the network at. The city is a regular mesh of streets (depicted as vertical and horizontal segments on a map), with crossroads located at points with integer coordinates. Nodes may only be located at crossroads of streets (no two nodes at the same crossroad). Cables may only run along streets and must connect nodes by the shortest possible route under this constraint. Moreover, a cable of precisely such length must be currently in stock (you are provided with a list of possible cable lengths). Try to layout the network in such a way as to minimise the total length of cable used.

Input

The input starts with a line containing integer t <= 1000, the number of test cases. t test cases follow.

The first line of a test cases begins with integer k, denoting the number of different available cable lengths, followed by k space separated integers p_j corresponding to the allowed lengths of cables (1 <= k <= 100, 1 <= p_j <= 100). The next line contains two integers n m, denoting the required number of nodes and cables in the network, respectively (1 <= n <= 100, 1 <= m <= 1000). The next m lines contain a pair of integers a_j b_j each, signifying that nodes a_j and b_j should be connected by a cable (1 <= a_j,b_j <= n).

Output

A valid solution to the i-th test case consists of a line with the text '`city` *i* `Y`', followed by n_i lines each containing two integers, the x- and y-coordinates of successive nodes in the solution (0 <= x,y <= 100).

It is guaranteed that for every test case there exists at least one possible solution. You can however leave out a test case by outputting the line '`city` *i* `N`' instead of a valid solution.

Score

For each correctly solved test case you are awarded (m/sum) * (($p_1+p_2+...+p_k$)/k) points, where *sum* is the total length of all cables used.

The score awarded to your program is the sum of scores for individual test cases.

Example

Input:
```
4
2 1 2
4 5
1 2
2 3
3 4
1 4
2 4
1 2
4 5
1 2
2 3
3 4
1 4
2 4
2 1 2
5 8
1 2
1 3
1 4
1 5
2 4
2 5
3 4
3 5
1 1
2 1
1 2
```

Output:
```
city 1 Y
0 0
0 1
1 1
1 0
city 2 Y
2 0
1 1
0 2
0 0
city 3 Y
0 1
0 2
1 1
1 2
0 0
city 4 N
```

Score:
```
score = 3.340003
```

Bonus info: If score = *xxx.xxxaaa*, *aaa* means the number of test cases with non-zero score...

Added by:	Michał Małafiejski
Date:	2004-11-14
Time limit:	17s
Source limit:	50000B
Languages:	All
Resource:	-

SPOJ Problem Set (challenge)

240. Santa Claus and the Presents

Problem code: SANTA

Every year Santa Claus faces a more and more difficult task. The number of children in the world is increasing rapidly, while Santa's old patched up sack can only accommodate a few presents at a time. And every child wants their own very special present... This means that, ever so often, once his sack is partially or completely empty, Santa has to fly back to his base in Lapland to replenish his supplies. So irksome has this become that Santa has decided to resort to modern operational research to aid him in his sack-packing and route planning. Please write a program which will guide Santa through his daily chores.

Input

The input starts with a line containing a single integer t<=100, the number of test cases. t test cases follow.

The first line of every test case consists of four integers n x y S, denoting the number of good children who will receive a present from Santa, the x and y coordinates of Santa's home base, and the amount of space in the sack, respectively (1<=n<=10000, -10000<=x,y<=10000, 1<=S<=100000). n lines follow, each consisting of three integers x_i y_i s_i - the x and y coordinates of the i-th child's home, and the amount of space taken up by this child's present when in Santa's sack, respectively (-10000<=x_i,y_i<=10000, 1<=s_i<=S).

Output

For each test case output a sequence of space separated integers, corresponding to successive actions that should be taken by Santa:

- *-i* (1<=i<=n) signifies that Santa should travel to his base and pack the present for the i-th child into his sack (he needs to have sufficient room in his sack to do this).
- *i* (1<=i<=n) signifies that Santa should travel to the i-th child's home and leave a present for him/her.
- 0 signifies that Santa should travel back to his base and end his Christmas activity (and that you want to proceed to the next test case).

Assume that Santa starts in his base and always travels between points by the shortest possible route (along straight lines). All distances are measured using the Euclidean metric.

Score

The score awarded to your program is the total of all scores obtained for its individual test cases. The score for a test case is calculated as I/P, where P stands for the distance covered by Santa in your solution, while I is a test case specific constant (I= n*d+D*(s_1+...+s_n)/S; d is the mean distance between two homes, while D is the mean distance between Santa's base and a child's home for the given test case).

No points are awarded for incompletely solved test cases (when not all the children receive presents). Any other violation of transport rules results in Wrong Answer.

Example

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

Output:
```
-1 -2 1 2 -3 3 0
```

Score:
```
2/(1+1+1+1)= 0.5
```

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 (challenge)

246. Plant a Christmas Tree

Problem code: CTQUINE

Evergreen trees are really wonderful. They were treasured by all civilisations of the Western world, from ancient Egiptian priests to Celtic druids in the British Isles. In the late Middle Ages a tradition of placing an evergreen tree at home for Christmas developed in Germany and spread throughout the Old and New World, reaching the Asia Pacific and America in the XIX-th century.

Surely, you must have noticed the sad fact that nowadays this global custom, however beautiful it may be, results in the death of millions of coniferous trees worldwide. Help us in our effort to restore the healthy balance. In the Christmas period, draw & plant your own tiny fir tree!

Since we are limited to text mode, there is little room for creative art, and solid, well built trees are definitely favoured. An *ideal tree* consists of several lines (at least 1) of the same length, consisiting of ASCII characters -- both whitespace ("spaces"), and non-whitespace ("relevant characters"). Counting from the top, the number of characters between the first and last relevant characters in a line (inclusive) is equal to 1, 1, 3, 1, 3, 5, 1, 3, 5, 7, 1, 3,... for consecutive lines. The line for which this distance is the largest begins and ends with relevant characters. All other lines contain exactly the same number of spaces to the left of the leftmost relevant character and to the right of the rightmost relevant character (this gives the ideal tree a nice, vertical trunk).

Please write a program which outputs a tree as close to an ideal tree as you can get, and keeps it as small as possible (such a tree has the largest chance of sprouting roots when planted). *And it can hardly come as a surprise to you to learn that the source code of the program you submit has to be identical to the text it writes to output (character by character, there are no exceptions)!*

Score

Your program will be judged as follows: if the program is not a quine (i.e. if it contains no relevant characters or outputs text different than its own source code) it will be judged as a Wrong Answer. Any other program will receive some number of penalty points depending on its size and quality as a tree (the fewer points, the better). One penalty point is given for every line of code used. 10 penalty points are given for a line without any relevant characters (how can you expect a broken tree to grow?). For non-empty lines, the position of the leftmost and rightmost relevant characters in the analyzed tree are compared with respect to corresponding positions in an ideal tree with the same number of lines. The squared differences in position between these two pairs of characters are added to the penalty score.

Technical note: a single newline character (ASCII 10) should be used to terminate all lines. ASCII characters 32 (space) and 9 (tab) are treated as single spaces, all other characters are considered relevant. Notice that the problem description doesn't penalise for left out or excessive spaces after the last relevant character of a line (but doesn't allow any difference between the source code and output text in this respect).

Example

C source code:

```
   ;
   ;
  ;;;
   ;
  main(
 )  {;
   ;
  ; ;
 /* */
;return
   0
  ;;}
```

This code would be judged as Wrong Answer, since it isn't a valid quine. Were it a quine, it would receive 16 penalty points (12 for 12 lines, $4=2^2$ additional penalty points for a misplaced rightmost relevant character in line 5).

Solutions to this problem may only be submitted in the following languages: C, C++, Pascal, Java, C#, Python, Haskell, OCaml, Brainfk, Intercal.**

Added by:	**Adrian Kosowski**
Date:	**2004-12-16**
Time limit:	**17s**
Source limit:	**30000B**
Languages:	**All except: ADA ASM BASH CLPS LISP clisp LISP sbcl D FORT ICON JAR LUA NEM NICE PAS gpc PERL PHP PIKE PRLG RUBY SCM qobi ST WSPC**
Resource:	**DASM Programming League 2004, problemset 4**

SPOJ Problem Set (challenge)

270. Digits of Pi

Problem code: PIVAL

In this problem you have to find as many digits of PI as possible.

Output

Output must contain as many digits of PI as possible (not more than 1000000).

Score

The score awarded to your program will be the first position of the digit where the first difference occured.

Example

```
Output:
3.1415926535897932384626433832795
```

will be awarded with 33 points.

Added by:	Roman Sol
Date:	2004-12-27
Time limit:	25s
Source limit:	4096B
Languages:	All except: LISP sbcl LISP clisp CLPS
Resource:	ZCon 2005

SPOJ Problem Set (set6)

276. Herdkeeper

Problem code: MFENCE

After the tragic end of the watermelon plantation, Johnny has switched to farming. More precisely, he is now a Certified Livestock Supervisor (i.e. shepherd) tending herds of antelope. It is his task to divide the animals into herds, and to build a fence around each herd, trying to keep the total length of all fences as small as possible. Each herd must consist of at least 2 antelope, and the antelope may stand arbitrarily close to the fence itself.

Input

t [the number of test cases <= 1000]
n [2 <= the number of antelope <= 100]
x_1 y_1 [coordinates of the antelope, between -1000 and 1000]
x_2 y_2
.....
x_n y_n
[next test cases]

Output

case *i* Y [N - if you want to skip the testcase]
c [the number of herds]
a s_1 s_2 ... s_a [2 <= *a* - the number of antelope in the first herd, and the numbers of the antelope in this herd in the order from the input sequence]
[next test cases]

Scoring

The score awarded to your program is the sum of scores for individual test cases. For the i-th test case you will receive 1 / (1 + *sum* / *conv*) points, where *sum* is the sum of circumferences of all convex hulls of herds in your solution, and the *conv* is the circumference of the convex hull of the set of all antelope. If you don't want to solve the *i*-th test case, you may output the line 'case *i* N' and nothing else.

Example

```
Input:
6
2
0 0
5 0
3
4 0
-4 -5
2 3
```

```
5
20 10
10 10
40 50
-20 -40
-30 -20
4
2 4
2 -4
2 0
-5 -3
3
2 4
-4 -4
2 3
4
-1 -3
-1 5
3 -5
-1 5
```

Output:

```
case 1 Y
1
2 1 2
case 2 Y
1
3 1 2 3
case 3 Y
2
3 1 2 3
2 4 5
case 4 Y
2
2 1 4
2 2 3
case 5 Y
1
3 1 2 3
case 6 Y
1
4 1 2 3 4
```

Score:

```
3.079001
```

Bonus info: If score = *xxx.xxxaaa*, *aaa* means the number of test cases with score > 0.5

Added by:	Michał Małafiejski
Date:	2005-01-07
Time limit:	17s
Source limit:	50000B
Languages:	All except: C99 strict
Resource:	-

SPOJ Problem Set (challenge)

285. Atomic Shelters

Problem code: ATSHELT

The election campaign of the mayor of Byteland continues. His advisors firmly believe that a military touch might do good to his image. On the other hand, aggressive use of arms might arouse the insane anger of the pacifist part of the electorate. So, investing in national defence seems to be the best solution. And this is why the capital of Byteland will receive its first ever atomic shelters.

The Bytelandian capital consists of exactly *n* buildings and the mayor intends to build shelters underneath exactly *k* of them. Now it is your task to layout the shelters in the city in such a way as to minimise the maximum distance a citizen of Byteland may have to cover to reach the nearest atomic shelter. After all, there is nothing more important than a mayor who guarantees your safety by putting an atomic shelter not far from your house.

Input

t [the number of test cases <= 1000]
n k [2 <= the number of different buildings <= 100, 1 <= the number of shelters <= *n*-1]
x_1 y_1 [-1000 <= coordinates <= 1000]
x_2 y_2
.....
x_n y_n
[next test cases]

Output

case *i* Y [N if you wish to skip this test case]
b_1 b_2... b_k [numbers of buildings to put shelters in, in increasing input order]
[next test cases]

Scoring

The score awarded to your program is the sum of scores for individual test cases. For the i-th test case you will receive *diam* / *dist* points, where *diam* is the distance between the furthest two buildings of the city, while *dist* is the longest distance from a building to the nearest shelter in your solution. If you don't want to solve the *i*-th test case, you may output the line 'case *i* N' and nothing else.

Example

```
Input:
5
5 2
-3 -4
-4 3
2 -3
-2 -3
```

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

Output:

```
case 1 Y
3 4
case 2 Y
1 3 4 5
case 3 Y
4 5
case 4 Y
1 2 3
case 5 N
```

Score:

```
5.592004
```

Bonus info: If score = *xxx.xxxaaa*, *aaa* means the number of test cases with Y answer.

Added by:	Michał Małafiejski
Date:	2005-01-21
Time limit:	17s
Source limit:	50000B
Languages:	All
Resource:	-

SPOJ Problem Set (challenge)

289. The Turing Music Box

Problem code: TMBOX

If you've ever dealt with the theory of information, you are no doubt familiar with the theoretical notion of a *Turing Machine*. But have you ever wondered what you could do if you got a *real* Turing Machine -- one of those big metal things with all the cranks and levers and rolls of infinite tape that looks suspiciously like toilet paper?...

The sad answer is: there are few interesting things that can be done with such a machine. Even problems that have a little charm in the theoretical model (like the intractable Halting Problem) can be solved very efficiently with practical brute-force algorithms (see e.g. the figure at the end of the problem description). But there is one thing that you can do with a practical Turing Machine, and can't do with a theoretical one, and it is: to use it as a music box.

Our Turing Machine has exactly one state variable (an integer in the range 0 to 999) and is equipped with an infinite tape, consisting of cells with symbols from a given alphabet encoded on them. A movable read/write head is positioned over some cell of the tape, and is operated according to the list of rules encoded in the machine. The rules are of the form `S1 C1 S2 C2 M`, which means: if the machine is in state S1 and C1 is written in the current cell, change state to S2, write C2 in the current cell, and move the head as described by move M (one cell left, one cell right, or not at all). If no matching rule is found for the given state the machine should halt.

Now, here is the good bit. The head makes a creaking sound when performing each rule. It goes `da` when moved right, `di` when moved left, and `um` when left in place. Suppose that each cell of the tape can contain one of 16 possible symbols, formed as the concatenation of exactly two of the words: `da`, `di`, `um` and `sh` for silence. Initially, nearly all the cells of the tape are filled with the symbol `shsh`. Only a few (not more than 500) consecutive cells form a piece of music, each cell encoding a pair of sounds (one of 9 combinations of `da`, `di` or `um`, without any silences). The head of the machine is initially positioned over the leftmost of the cells containing sounds.

Now it is your task to use the Turing machine to play the piece of music written on its tape (as read from left to right, starting from the initial position of the head, as far as the first silence) as accurately as possible, using the head itself to produce the sounds required.

Output

The output of your program must contain a set of rules describing the behaviour of the Turing Machine designed for playing music. Each rule must be of the form `S1 C1 S2 C2 M`, where S1 and S2 are integers from the range 0..999, C1 and C2 belong to the 16 symbols of the alphabet, while M describes the move direction of the head by the sound it makes (`da`, `di` or `um`).

Score

Your program will be tested multiple times for different pieces of music written on the tape. The score of your program is equal to the total of non-negative scores, taken over all test cases.

For a test case with n notes (n/2 non-silent cells) your program will receive n-d points, where d denotes the edit distance between the music played and the music required (i.e. the minimum total number of notes that have to be inserted into or changed in both the pieces to obtain the same piece of music).

Example

Consider the following set of rules output by a program:

```
000 dada 000 dada da
000 umda 000 dada da
000 shsh 000 shsh da
000 didi 001 didi di
001 dada 002 didi di
```

Then the results of exemplary testing could be as follows:

```
Music: da da|da da|da da|di di|um um
Plays: da da da di di
Score: 5

Music: um da|um da|um da|da um|di di
Plays: da da da
Score: 3

Total: 5 + 3 = 8 points
```

Bonus info: There are no more than 100 tests. The score format is *s.xxyy*, where *xx* denotes the number of tests for which your machine played the music perfectly, *yy* - the number of tests for which it received a positive score.

While this machine halts, it loops. We just help it in its agony.

Added by:	Adrian Kosowski
Date:	2005-01-26
Time limit:	17s
Source limit:	50000B
Languages:	All
Resource:	DASM Programming League 2004, problemset 6

SPOJ Problem Set (set7)

294. Johnny and the Optimisation of Homework

Problem code: HWORK

One day when Johnny was still a schoolboy he got caught red-handed by his teacher while doing a Very Mischievous Thing (of the sort that you would expect of Johnny). As a punishment he was told off and assigned additional homework. The teacher underlined quite a few words in a dictionary and asked Johnny to rewrite all of them to his notebook.

Johnny wasn't at all pleased about this, since writing by hand is always a painful burden. Fortunately, Johnny's dad took pity on the crying boy and offered to help. He presented his son with a few sheets of carbon paper, thanks to which any text Johnny wrote was at once ready in exactly *k* copies. Some of the characters of particular copies could then be erased using a white correction pen, so as to obtain only the words required by the teacher. All the characters forming a single word have to be directly adjacent, but words can be written in any order on the sheets and different words can be separated by an arbitrary (possibly 0) amount of space.

Johnny has cheered up considerably by now, since the bit with the carbon paper and correction pen sounds rather fun. All that remains to be done is to write down an appropriate text, obviously keeping it as short as possible. Please advise Johnny what to write.

Input

Input begins with a single integer t (t=1000). t test cases follow.

Each test case starts with a line containing two integers n k, respectively denoting the number of words the teacher has asked Johnny to write and the total number of carbon copies that Johnny creates, including the original (1<=k<=n<=512). Each of the next n lines contains a word assigned by the teacher - a string of between 4 and 12 characters 'a', 'b', 'c' or 'd'. All words given at input are Bytelandian nouns in common use.

Output

For the i-th test case output a line with the text `case i Y` or `case i N`, specifying whether you wish to solve the given case. Then in the former case print a single line containing the text that should be written by Johnny. Exactly n lines with a single integer on each should follow, the i-th representing the position of the first letter of the i-th word (on the page on which this word eventually appears, before applying the correction pen).

Scoring

The score awarded to your program is the sum of scores taken over all test cases you chose to solve.

For each test case, the score is the difference between the total number of characters of the words provided by the teacher and the number of characters of the text eventually handwritten by Johnny. Programs which receive a negative score for some test case will be considered incorrect.

Example

Input:
```
1
4 2
aaaa
aaaa
aaaa
bbaaa
```

Output:
```
case 1 Y
aaaabbaaaa
1
1
7
5
```

Score:
```
17 - 10 = 7
```

When given in to the teacher, the 2 pages of homework may look as follows:

```
aaaa__aaaa
aaaabbaaa_
```

Added by:	Michał Małafiejski
Date:	2005-02-11
Time limit:	17s
Source limit:	50000B
Languages:	All except: C99 strict
Resource:	-

SPOJ Problem Set (set8)

295. Bytelandian Origami

Problem code: BRIGAMI

Many of Johnny's school friends have perfected the art of folding a square sheet of paper into beautiful shapes (known as *origami*). Johnny attempted to follow suit, but to his dismay he found that his fingers were a little to clumsy for the task in hand. After spending yet another day creating something especially disastrous (later named "From the series: *Crumpled Pieces of Paper Seen with an Artist's Eye*, No. 27"), Johnny decided he'd had enough. Therefore he proudly proclaimed to all his friends that origami was not fit for serious people, and that he intended to become the master of *kirigami*, the art of cutting paper. But after experimenting with kirigami for a few weeks, he sold the rather miserable results of his labour to the local confetti store, and announced that true beauty lay in convex polygons, and that they were the only shapes a true artist should ever cut. Still, if a person is as lazy and inapt as Johnny, even such a seemingly simple task may turn out a real challenge.

The method Johnny uses to create works of art consists of several steps. First, he takes a sheet of paper in the shape of a convex polygon and uses a ruler and pencil to draw a convex polygon (lying entirely within the sheet). Then, he proceeds to cut it out using a ruler and a razor-edged paper cutter. Every cut is thus a segment of a line, reaching from one edge of the sheet of paper to another, and adjacent to one side of the drawn polygon. Johnny then discards the cut off corner of the sheet and continues cutting until the shape outlined in pencil is completely cut out. Since he is extremely disinclined to perform hard work, please write a program to help him minimise the total length of the lines along which the paper is cut.

Input

Input begins with a single integer t (t=200). t test cases follow.

Each test case starts with a line containing two integers m n, denoting the number of vertices of the sheet of paper and the shape drawn on it, respectively (3<=m,n<=600). The next m lines contain two integers a_i b_i each (-20000<=a_i, b_i <=20000), corresponding to the x and y coordinates of vertices of the sheet of paper (given in clockwise order). The description of the shape drawn on the sheet follows, given in the next n lines in a similar form.

Output

For the i-th test case output a line with the text `case i Y` or `case i N`, specifying whether you wish to solve the given case. In the former case, in the next line output a permutation of the numbers 1...n denoting the order in which Johnny is supposed to cut out the respective sides of the shape drawn on the sheet (vertices are numbered from 1 to n in the input order, and side *s* connects vertices *s* and 1 + *s* mod n).

Score

The score awarded to your program is the sum of scores taken over all test cases you chose to solve.

For each test case, the score awarded to your program is equal to the ratio of the perimeter of the sheet of paper and the total length of the lines along which the paper is cut.

Example

Input:
```
3
4 4
0 0
0 2
2 2
2 0
0 1
1 2
2 1
1 0
4 3
0 0
0 3
3 3
3 0
1 1
1 2
2 2
4 3
0 0
0 3
3 3
3 0
1 1
1 2
2 2
```

Output:
```
case 1 Y
1 2 3 4
case 2 Y
1 2 3
case 3 Y
3 2 1
```

Score:
```
4.94
```

Added by: Michał Małafiejski
Date: 2005-02-11
Time limit: 17s
Source limit:50000B
Languages: All except: C99 strict
Resource: -

SPOJ Problem Set (challenge)

298. Closing down Railway Lines

Problem code: DERAIL

Many cities in Byteland look back on the days when Johnny the First was king, and when nobody bothered about public spending. One of things that the citizens liked most about Johnny was that whenever he had a hangover, he would sign any public petition brought forth to him, just for the sake of peace and quiet. Rail travel was extremely popular, and lots of cities and villages requested railway lines connecting them directly, to which King Johnny always graciously agreed (even if he wasn't quite sure what he was agreeing to). Seeing that money was no object, the railway tracks were built in such a way as to connect pairs of cities directly, along straight lines. If two railroads intersected, a complex intersection involving bridges and tunnels was built and everyone seemed perfectly happy.

And then after Johnny's abdication, democracy returned, and the happy days of Byteland ended. One of the first things that had to be done was closing down most of the railway lines. The new government intends to disassemble a large part of the direct railway connections, preserving barely enough to make travel possible between any two cities (perhaps via other cities on the way). The total cost of maintenance of the lines which remain open, equal to *k* Bytelandian Dollars per kilometer of track open and *l* Bytelandian Dollars per intersection of 2 used tracks, is to be as low as possible. Please help the government decide which railroads should remain open.

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 k l, denoting the number of cities, the number of direct connections between cities, the cost of upkeep of a kilometer of track, and the cost of upkeep of a single railway line intersection, respectively (3<=n<=m<=10000, 0<=k,l<=100000). Each of the next n lines contains two integers x_i y_i, corresponding to the X and Y coordinates of the i-th city, measured in kilometers, respectively (-40000<=x_i,y_i<=40000). Then exactly m lines follow, containing a pair of integers a_i b_i each, which denote that cities a_i and b_i, numbered in input order, are connected by a direct railway track (1<=a_i, b_i<=n). No three cities are collinear and no three tracks intersect at one point. All tracks are bidirectional.

Output

For the i-th test case output a line containing the text `case i Y` or `case i N`, specifying whether you wish to solve the given case. Then in the former case, write exactly n-1 lines containing one integer each -- the numbers of the railway connections that ought to be left open (numbered in input order). It is guaranteed that for the input data some solution always exists.

Score

The score awarded to your program is the sum of scores received for the test cases you chose to solve. For each such test case you will receive (s/c)-1 points, where s is the cost of maintenance of the original configuration, while c is the cost of maintenance of only those railway lines which you've selected.

Example

Input:
```
1
4 5 1 100
0 0
0 1
1 1
1 0
1 2
2 3
1 3
3 4
4 2
```

Output:
```
case 1 Y
3
2
5
```

Score:
```
(100+1+1+1+1.414+1.414) / (100+1+1.414+1.414) - 1 = 0.019
```

Illustration to sample test data

Added by: Adrian Kosowski
Date: 2005-02-19
Time limit: 17s
Source limit:50000B
Languages: All
Resource: DASM Programming League 2004, problemset 7

SPOJ Problem Set (challenge)

313. The Game of Crosses & Crosses

Problem code: CROSSES

The game of gomoku (otherwise known as naughts & crosses), played on an *n x n* board has many interesting variations. One of them is the Game of Crosses & Crosses, with the following set of rules:

- Two players - red and black - take it in turns to place one cross of their respective color on an unoccupied square of the *n x n* gaming board. Red starts the game.
- After each player's move any rectangles with sides equal to at least 2, lying entirely within the gaming board and covered completely by crosses, are simultaneously removed (cut off) from the gaming board and the game continues.
- When all the squares remaining in the gaming board are covered by crosses, the game comes to an end. The score of each player is equal to the number of crosses of his color left standing on the gaming board, and the player with the higher score is considered the winner.

The game of crosses & crosses feels rather like playing a degenerated game of Go with an army of suicide bombers. For many years now it has been the favourite passtime of Bytelandian schoolchildren during their lessons. Little Johnny was no different, and among his friends he actually became a notable crossing champion.

But not many people knew about Johnny's crossing talent, and Johnny often used this to his advantage. So when a few years after Johnny's abdication from the throne of Byteland an unsuspecting publisher signed a million dolar contract with the ex-king for a series of memoirs entitled *The famous victories of Johnny the Great*, he was certainly not prepared for what he received -- a detailed account of Johnny's childhood games of crosses & crosses. To make matters worse, all accounts are written by Johnny in exciting prose, rich in action, e.g.: "`Then I played yet another game on a 3x3 board. I placed my first cross at (1,1). Then I placed a cross at (2,3). The next cross I placed at (2,2). The cross after that I placed at (3,3). Finally, I placed a cross at (1,2) and I won the game 2:1.`".

In a desperate effort to save the day, the publisher employed you to create illustrations for the book. You are given a free hand in reinacting the games (and in particular the oponent's moves, which Johnny has modestly left out), provided your version of events is not an evident contradiction of Johnny's text.

Input

Input begins with a line containing a single integer t (t=100). t test cases follow.

Each test case starts with a line with three integers describing a single game: n sr sb, denoting the length of the side of the playing board, the number of points scored by the red player (Johnny) and the number of points scored by the black player (Johnny's oponent), respectively (3<=n<=250, 0<=sb< sr). The next ceil(n^2/2) lines contain 2 integers x_i y_i each - the coordinates of the squares where Johnny placed his crosses in successive moves (1<= x_i, y_i <= n).

Output

For the i-th test case output a line with the text `case i Y` or `case i N`, specifying whether you wish to solve the given case. Then in the former case print exactly floor(n^2/2) lines containining 2 integers each - the coordinates of the squares where Johnny's anonymous oponent placed his crosses in successive moves.

Scoring

The score awarded to your program is equal to the number of correctly solved test cases. For each case, the game defined by yours and Johnny's description must have the outcome (final score) defined at input.

Example

Input:
```
1
3 2 1
1 1
2 3
2 2
3 3
1 2
```

Output:
```
case 1 Y
3 1
1 3
2 1
3 2
```

Score:
```
1
```

Illustration to sample test data

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

---

Added by:	Adrian Kosowski
Date:	2005-03-09
Time limit:	17s
Source limit:	50000B
Languages:	All
Resource:	DASM Programming League 2004, problemset 8

SPOJ Problem Set (challenge)

314. Digits of e

Problem code: EVAL

In this problem you have to find as many digits of E as possible.

Input

There is no input for this problem

Output

Output must contain as many digits of E as possible (max = 1000000)

Score

The score awarded to your program will be the first position of digit where first difference occured.

Example

Output:

```
2.7182
```

will be awarded with 6 points.

Added by:	Roman Sol
Date:	2005-03-10
Time limit:	25s
Source limit:	4096B
Languages:	All
Resource:	ZCon

SPOJ Problem Set (challenge)

315. The Secret Fellowship of Byteland

Problem code: BFORG

The relationship between The University of Byteland and King Johnny was never a friendly one. The king was the easy-going, open-minded sort of person who is prepared to turn a blind eye to the embezzlement of public funds, but inwardly revolts at the thought of money going to waste, and supporting a university was to the king a perfect example of a waste of money. On the other hand, the chancellor of the university showed no tolerance whatsoever, and frequently stated in public that Byteland was being governed by a monarch who took terrible decisions when he was drunk and even worse ones when he was sober. After some time of bad-tempered coexistence, the king had had enough and decided to close down the university. However, the king's councillors advised against this move, suggesting it might cause social unrest. The king yielded to their advice, and instead established a law which banned all organisations, clubs and associations active at the university.

This action had a rather curious effect on the usually lazy students of the university. They had never before even thought of organising any sort of fellowship, but now they immediately decided they needed to set one up. And this is how the *Secret Fellowship* came to life.

The main problem that faced the management of the Fellowship was to organise members' meetings in such a way as to minimise the risk to the participants. It was decided that the *n* members of the fellowship should be split into *k* secret divisions, each consisting of at least 2 members. All members belonging to the same division would then meet regularly, and they would take it in turns to host the meetings of the division in their houses.

But one more important factor has to be taken into account -- the laziness of students. It is therefore your task to form the divisions in such a way that the furthest distance a student may ever be asked to walk is as short as possible.

Input

The first line of input contains a single integer t, the number of test cases (t=1000). t test cases follow.

Each test cases starts with a line containing two integers *n k*, denoting the number of students and the number of divisions to be formed, respectively (2<=2*k*<=*n*<=200). Each of the next *n* lines contains two integers x_i y_i each (-1000 <= x_i,y_i <= 1000), denoting the coordinates of the houses of successive students.

Output

For the *i*-th test case output a line with the text `case i Y` or `case i N`, specifying whether you wish to solve the given case. Then in the former case print exactly *k* lines. Each line should start with integer n_j (n_j>=2) and be followed by a space separated list of exactly n_j increasing integers s_{jl}, denoting the students belonging to the *j*-th division, numbered in input order (1<=s_{jl}<=n). All divisions must be disjoint and the sum of all numbers n_j must equal *n*.

Score

The score awarded to your program is the total of scores for the test cases you chose to solve.

For each solved test case you will receive *diam* / (*d***k*) points, where *diam* denotes the distance between the two furthest houses of members of the fellowship, and *d* is the distance between the two furthest houses of members belonging to the same division.

Example

Input:
```
2
6 3
0 0
1 0
0 1
1 1
2 0
2 1
6 2
0 0
1 0
0 1
1 1
2 0
2 1
6 2
0 0
1 0
0 1
1 1
2 0
2 1
```

Output:
```
case 1 Y
3 1 2 4
3 3 5 6
case 2 Y
3 1 2 5
3 3 4 6
case 3 Y
2 1 3
4 2 4 5 6
```

Score:
```
1.849003
```

Bonus info: If score = *xxx.xxxaaa*, *aaa* means the number of test cases with Y answer.

Added by:	Michał Małafiejski
Date:	2005-03-11
Time limit:	17s
Source limit:	50000B
Languages:	All
Resource:	-

SPOJ Problem Set (challenge)

316. Japan Crossword

Problem code: JCROSS

Japan crossword is a very popular game. It represents encoded picture which consists of filled block of cells. At the start of game you see empty grid. Each row (column) has some numbers in beginning of the row (column). Each number means how many continious cells are filled in a hidden picture (length of the filled blocks). Filled blocks of cells are arranged from left to right and from top to bottom. Between filled blocks must be at least one empty cell. For example, numbers are 4, 2, 7 mean that there are three groups with 4, 2, and 7 filled cells in it. Your task is decode hidden picture using hints.

[IMAGE] [IMAGE]

Input

The first line of input contains a single positive integer t<=300 - the number of test cases. Then for every test case first line specifies integer numbers R and C (number of rows and columns) of the picture (1<=R<=50, 1<=C<=100). Below R lines are follow. Each line consists of any integers for horizontal hints. The very last number for every line is 0. Then C lines are follow. Each line consists of any integers for vertical hints. And again every line ends with 0.

Output

For every test case you should write decoded picture in the form of rectangle with R rows and C characters in each line. Symbol '#'(sharp) means filled block and symbol '.'(point) means empty cell.

Score

The score awarded to your program is the total of all scores obtained for its individual test cases. The score for a test case is calculated so that for each 'right' row or column you get 1 points. The row(column) is counted as a 'right' if there is a group of filled cells for every number in beginning of the row(or column) and lenght of every cell is equal corresponded number. If All rows and columns are 'right' your score multiply by 1.5 for this test case.

Example

```
Input:
1
10 5
3 0
2 2 0
5 0
5 0
3 0
1 0
1 0
3 0
```

```
2 0
3 0
3 0
5 0
1 8 0
5 3 0
3 1 1 0
```

Output:

```
.###.
##.##
#####
#####
.###.
..#..
..#..
..###
..##.
..###
```

Score:

```
(10+5)*1.5 = 22.500
```

Bonus info: If score = *xxx.xxxaaa*, *aaa* means the number of entirely correct test cases

---

Added by:	Maxim Sukhov
Date:	2005-02-07
Time limit:	21s
Source limit:	50000B
Languages:	All
Resource:	;)

SPOJ Problem Set (challenge)

317. Simple Image Recognition

Problem code: IMGREC1

One of the hard problems that borrows human minds and can find the practical application in creating Artifical Intelegence is problem of Image Recognition. This problem in its simplest form can be applied in many spheres of manufactures. In given problem we interest in one elementary case of Image Recognition. You have to make choise form only two possible images that are represented on a bicoloured picture. This images is "dagger" or "zero". This images can be rotated, deformed, scaled, moved, have some noise or different width of lines on the picture. But human always can correctly define that is represented on a picture.

Input

t - number of test cases, than *t* test cases follows. [*t* <= 100]
[empty line]
Eache test case starts with integer N equals to number of pictures in this test, than N pictures follows. [4 <= N <= 10]
[empty line]
Description of each picture starts from two integers H and W - height and width of picture accordingly. [5 <= H, W <= 50]
than follows exactly H lines each consists of W chars.
Description of picture consists of two simbols only: 'x' - painted square and '.' - empty square. You can be assured, that no other symbols are present at the description of a picture.

Output

For each test it is necessary to deduce on a separate line a string of chars with length equals to N. The string should consist of a set of two chars 'x' and '0'. Where 'x' corresponds to a dagger on a picture, and '0' corresponds to a zero. If answer will contains other chars or length of a string won't equals to N you will receive status "Wrong Answer".

Score

The score awarded to your program is the sum of scores for individual test cases. The score for individual correctly solved test equals to N (Number of pictures in this test).

Example

Input:

```
1

5

5 5
x...x
.x.x.
```

```
..x..
.x.x.
x...x
5 5
xxxxx
x...x
x...x
x...x
xxxxx
6 6
..x...
..x...
xxxxxx
..x...
..x...
......
5 5
.xxx.
x...x
x...x
x...x
.xxx.
5 5
.xxx.
.x.x.
.xxx.
.....
.....
```

Output:

```
x0x00
```

Output:

You will recieve 5 points for this solution

Added by:	Roman Sol
Date:	2005-02-09
Time limit:	21s
Source limit:	50000B
Languages:	All
Resource:	ZCon 2005

SPOJ Problem Set (challenge)

321. X-Words

Problem code: XWORDS

It is quite simple really: I'll give you a list of words and you use them to make a crossword puzzle in a 16x32 grid. You'll be able to use the words more than once in the grid and there is a special "flipper" square you can use as a wild card. The winner will be the program that can create the "best" fully connected crossword in one minute. The original problem appeared here: Programmer of the month contest (Feb. 2005).

The Starting Grid

- The grid will consist of 32 columns and 16 rows

The Word List

- There will be at least one word and fewer than 512 words in the wordlist
- Each word will be two letters long or more (WORDLENGTH >= 2)
- Each word will be sixteen letters long or less (WORDLENGTH <= 16)
- Words in the wordlist will contain only letters "A" through "Z" in upper case letters with no white space
- Words will appear in the wordlist with one word per line
- Words will not be repeated in the wordlist, but they may be used multiple times in your solution
- Do not assume anything (like sorting) about arrangement of the words in the list
- Do not assume anything about whether a "word" is contained in any dictionary: POTM, ABCDEFG, and XYZZY are all possible "words"
- Words in the list may be subsets of one another: SCAT, CAT, CATS and XCATS may all appear in the same wordlist ... there is no bonus for using words containing other words ... see scoring note below
- There may be words in the wordlist which are not possible to connect to any other words in the wordlist

Placement of the Words onto the Grid

- Any word from the word list may be used in your solution as many times as you wish
- You may use any subset of words in the wordlist, or all of them
- All words placed on the grid must read left-to-right or downwards
- All words placed on the grid must be connected to one another
- ONLY words on the wordlist may be used and empty squares or grid boundaries must be used immediately before and after all words
- Words may not "wrap-around" the grid boundaries in any sense
- Your solution does not need to be symmetric in any sense
- Output which is not connected, or contains words which do not appear in the wordlist, will receive a score of zero

The Flipper Square

- There is one (and only one) "flipper" square (denoted by an asterisk) permitted in your output
- You may place the flipper square within any word you place on the grid
- When used, it may represent a different letter in the horizontal and vertical words of which it is a part
- Any words formed using the "flipper" square must be part of the wordlist (if C*T is placed on the grid, then there must be a three letter word in the wordlist that begins with "C" and ends in "T")
- The "flipper" will likely be used at a word intersection, although this is not required (why would you use it elsewhere??)

Input

t - number of test cases [*t* <= 10]
N - number of words for given test case, then *N* lines follows each line contain one word, in upper case. Word will contain no whitespace or characters other than [A-Z].

Output

For each testcase your output must contain exactly 16 lines with 32 characters followed by a line feed as in printf("\n") on each line. The letters in your output must be upper case [A-Z} as in the wordlist. The "Flipper" (if used) in your output should be an asterisk "*". Squares that do not contain a letter or a flipper should contain an underbar "_". There should be no white space in your output. Your output must be exactly t*528 bytes.

Score

Your "SCORE" will be the total number of letters in all the words used in your solution. If a word is contained within a longer word, only the longer word will contribute to the score ... for example, using POTM would not score for the word POT even if both are in the wordlist. The "flipper" square contributes to the word length as though it was a part of each word. The total score will be the sum of scores for individual test cases.

Example

Input:

```
1
28
NECESSARY
POLITICAL
CONNECTED
SEPARATE
OPINIONS
REQUIRES
SEPARATION
SELFEVIDENT
UNALIENABLE
HAPPINESS
GOVERNMENTS
INSTITUTED
DERIVING
GOVERNMENT
DESTRUCTIVE
INSTITUTE
FOUNDATION
PRINCIPLES
ORGANIZING
ESTABLISHED
TRANSIENT
ACCORDINGLY
EXPERIENCE
SUFFERABLE
THEMSELVES
ABOLISHING
ACCUSTOMED
USURPATIONS
```

Output:

```
CONNECTED__USURPATIONS_CONNECTED
O_E___R___R_U____R_P___O_E___R__
NECESSARY_E_FOUNDATION_NECESSARY
N_E___N___Q_F____N_N___N_E_U_N__
E_S_INSTITUTED_INSTITUTE_S_F_S__
C_S_N_I___I_R____I_O___C_S_F_I_E
TRANSIENT_R_A_GOVERNMENT_A_E_E_S
E_R_T_N_H_E_B____N_S_X_E_R_R_N_T
D_Y_I_THEMSELVES_T___P_D_Y_A_T_A
____T___M___E__E_____E_____B___B
HAPP*NESS______P____PRINCIPLES_L
____T___E_SEPARATION_I_____E___I
SUFFERABLE_____R_____E_________S
____D___V_SEPARATION_NECESSARY_H
________E______T_____C_________E
HAPPINESS_THEMSELVES_ESTABLISHED
```

Score:

```
341
```

Added by:	Roman Sol
Date:	2005-04-08
Time limit:	25s
Source limit:	60000B
Languages:	All
Resource:	Programmer of the Month 02.2005

SPOJ Problem Set (challenge)

326. Enjoying a Multiplayer Game

Problem code: MGAME

One of the most popular types of computer multiplayer games in existence is the simple deathmatch shooter, in which it is the player's task to eliminate all other players on the gaming board. Usually, at the start of the game the players are distributed fairly randomly over the board, and run around in order to find and shoot opponents.

But there is a fair percentage of players (especially the younger ones) who enjoy the shooting most and give up the running altogether. To achieve this, at the start of the match all players are arranged very close to each other, and everyone opens fire in the very first second of the game. The gunfire continues until everyone within sight of everyone else is dead, and then the game ends, since no one feels like moving from their selected camping point.

Parents are often helpless when their children get addicted to this sort of entertainment, and don't know how to make them stop playing without causing a major quarrel. But Johnny's dad has developed the perfect method. He always says to his son: *Sure, I'll let you play another round, but tell me please how long it'll take!* And no, the answer *only a minute or two* is just not good enough.

At the start of the game, the players are positioned on the board and each player has a list of other players he is capable of eliminating (from his location). At the start of every second, each living player fires a round towards one of the opponents on his list (provided the list is not yet empty). Players who have been hit are eliminated from the game directly after the shots were fired. The situation continues until the lists of all surviving players are empty.

We are not asking you to give an exact answer the question posed by Johnny's dad, but only for an honest estimate. Given an arrangement of players on the board, try to find scenarios of shooting leading to the longest possible and the shortest possible game.

Input

The first line of input contains a single integer *t*, the number of test cases (t=100). *t* test cases follow. Each test case starts with a line containing integer *n*, denoting the number of players on the board (2<=*n*<=500). Each of the next *n* lines contains a list of integers: first, d_i the length of the i-th player's list, followed by the considered list of exactly d_i other players (numbered in input order from the range 1 to *n*).

Output

For the *i*-th test case output a line with the text `case i Y` or `case i N`, specifying whether you wish to solve the given case. Then in the former case print a description of the longest known game scenario, followed by a description of the shortest known game scenario. Each scenario starts with an integer *t*, the duration of the game measured in seconds (0<=*t*<=*n-1*). Each of the next *t* lines contains a list of integers, representing the identifiers of players eliminated by respective players in the given second (one integer for each player left alive and capable of hitting an enemy at the start of the second, ordered according to the input identifiers of the shooting players).

Score

The score awarded to your program is the total of scores for the test cases you chose to solve.

For each solved test case you will receive t_{max} / t_{min}-1 points, where t_{max} is the length of the first presented scenario, while t_{min} - the length of the second one.

Example

Input:
```
1
4
2 2 3
2 1 3
3 1 2 4
1 3
```

Output:
```
case 1 Y
2
3 3 4 3
2 1
1
2 1 4 3
```

Score:
```
2/1 - 1 = 1.00
```

Added by:	Adrian Kosowski
Date:	2005-04-14
Time limit:	17s
Source limit:	50000B
Languages:	All
Resource:	DASM Programming League 2004, problemset 9

SPOJ Problem Set (challenge)

353. Displace

Problem code: DISPLACE

You are given two strings S1, S2 of not more than 250 characters each. S1 does not contain characters ‘(‘ and ‘)’. You can swap two consecutive characters in S1. Your task is to do it in as small a number of swapping operations as possible to obtain a string which contains S2 as a substring (you can assume that for the given input, this can always be done).

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 S1
- The second line contains S2

Output

For each test case, output 0 iff you do not want to solve this test case. Otherwise, output a line containing the number 1 and two more lines of the following form:

- The first line contains an integer k representing the number of swap operations
- The second line contains k integers p1 p2,..., pk separated by single spaces, pi means that in the i-th operation, you swapped the i-th character and the (i+1)-th character in S1.

Score

Your task is to minimise your score for this problem. If you choose to solve a test case and the number of swap of operations is smaller than 5000, your score is equal to the number of operations. Otherwise, your score is 5000. Your total score is equal to the sum of scores for individual tests.

Example

```
Input:
1
ABCDEFGH
FC

Output:
1
3
5 4 3

Score:
3
```

Added by:	Walrus
Date:	2005-05-05
Time limit:	9s
Source limit:	50000B
Languages:	All

SPOJ Problem Set (set10)

356. Tethering the Mammoths

Problem code: MAMMOTH

Whereas most parks in different parts of the world are inhabited by pleasant little creatures like birds or squirrels, the nature park in Byteland's capital has somewhat larger inhabitants -- a herd of mammoths. As you may well imagine, this does lead to peculiar problems sometimes.

On one occassion the King of Bitland came on a state visit to Byteland, and, to everybody's surprise, decided he would take a stroll in the Mammoth Park. Since mammoths tend to be a little unpredictable and know nothing of the protocol of royal visits, they had to be tied up for the time being. But tying mammoths *properly* is not as easy as it sounds.

The park consists of little clearings connected by alleys, and on every clearing there stands a mammoth. Due to the lack of sterdy trees in the park, the only things you can tie a mammoth to are other mammoths. Since tying a mammoth by too few ropes may actually be more dangerous than leaving them alone, it is required that each mammoth has to be tied to *exactly k* other mammoths (that way all animals are kept safe and none of them has a feeling of being unfairly treated). The ropes connecting two mammoths must run along the park alleys and can only be 1 or 2 alleys long (in the latter case, the rope is assumed not to touch the mammoth in between). Finally, no two ropes may run a long a single alley, since this might result in an awful tangle.

It is your task to design which mammoths to tie together, or to determine that the required tethering is impossible to attain.

Input

The first line of input contains integer t, the number of test cases (t<=100). t test cases follow.

The first line of each test case contains three integers n m k, denoting the number of clearings (each with a mammoth on it), the number of alleys in the park, and the number of mammoths to tie each mammoth to, respectively (1<=k<=n<=m<=2000).

Each of the next m lines contains a pair of integers a_i b_i, denoting that clearings a_i and b_i are connected by an alley ($1<=a_i,b_i<=n$).

Output

For the i-th test case output a line containing the text `case i YES` if you know a solution to the given problem and `case i NO` in the opposite case. In the former case, you should then output exactly (k*n)/2 lines, containing the description of a rope between two mammoths. Each such line should begin with integer l, the length of the rope measured in alleys (l=1 or l=2) and be followed by exactly l+1 integers corresponding to successive clearings on the path of the rope.

It is possible that for the given test case no answer exists; in that case the only allowed solutions is `case i NO`.

Your score is equal to the number of test cases for which you gave the answer `case i YES`.

Example

Input:

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

Output:

```
case 1 NO
case 2 YES
1 2 3
1 1 4
case 3 NO
```

Score:

```
1+0+0 = 1
```

For the presented example, the optimal solution would score 2 points (for test cases 2 and 3).

Added by:	Adrian Kosowski
Date:	2005-05-08
Time limit:	42s
Source limit:	50000B
Languages:	All except: C99 strict
Resource:	DASM Programming League 2004, problemset 10

SPOJ Problem Set (challenge)

525. Fractions Calculator

Problem code: TFRACAL2

The input consists of exactly 1000 test cases in the following format recursive format:

Input

```
case i [i-th test]
< list_eq >:=< eq >'\n'[< list_eq >]
< eq >:=< var >=(< onp >,< fraction >)
< onp >:=(_< var >,< onp >)(_< var >,< onp >)< op >
< var >:=(a,c,g,t)[< var >]
< op >:=(+,*,/)
< fraction >:=< number >/< number >
< number >:=(1-9)[< number >]
```

< id >:=	the definition of the expression
< id >	on the right side: just use the definition of the expression
(*x*,*y*)	choose exactly one from the list: *x* or *y*
[*x*]	process (choose) *x* or not

The definition of every variable (on the left side) appears only once and follows its last appearance on the right side.

Output

The output should contain the list of variables in nondecreasing lexicographic order of identifiers and all values should be represented as simple fractions in lowest terms, i.e. in the form *N* / *D*, where *N* and *D* are relatively prime.

Score

The score is equal to the number of correctly solved test cases divided by 100.

Example

```
Input
case 1
c=_g_a/_cg_g/*
cg=_a_ct_a++
g=_a_a/_ct*
ct=_a_a*
a=2/2
```

```
case 2
t=_ct_ta*_ta*
ta=_c_a_a**
c=_ct_a+
ct=_a_a+_a_a*+
a=2/4
case 3
c=_t_cg_cg//
t=_g_g+_cg_g**
cg=_g_ct/
ct=_g_g/
g=6/71
case 4
g=_tt_tt_gt+*
t=_gt_tt*_tt/
gt=_tt_tt+_a_a*/
a=_tt_tt_tt/*
tt=2/62
case 5
c=_cc_t*
ca=_a_a/_a/
a=_cc_cc_t*+
cc=_t_t_t/*
t=76/13
```

Output

```
case 1 Y [write ’Y’ and the correct answer, write ’N’ if you don’t wish to answer]
a 1 1
c 3 1
cg 3 1
ct 1 1
g 1 1
case 2 Y
a 1 2
c 7 4
ct 5 4
t 245 1024
ta 7 16
case 3 Y
c 432 357911
cg 6 71
ct 1 1
g 6 71
t 432 357911
case 4 Y
a 1 31
g 1923 961
gt 62 1
t 62 1
tt 1 31
case 5 Y
a 6764 169
c 5776 169
ca 169 6764
cc 76 13
t 76 13
```

Score

0.05

Added by:	Michał Małafiejski
Date:	2005-05-14
Time limit:	24s
Source limit:	50000B
Languages:	All

SPOJ Problem Set (challenge)

528. Shortest Superstring

Problem code: TSSTR

Input

Input begins with a single integer *t* (*t* = 1000). t test cases follow.

Each test case starts with a line containing integer *n* denoting the number of words (1 <= *n* <= 100). Each of the next *n* lines contains a word - a string of between 4 and 16 characters ’a’, ’b’, ’c’ or ’d’.

Output

For the *i*-th test case output a line with the text `case i Y` or `case i N`, specifying whether you wish to solve the given case. Then in the former case print a single line containing the shortest superstring. Exactly *n* lines with a single integer on each should follow, the *i*-th representing the position of the first letter of the *i*-th word.

Scoring

The score awarded to your program is the sum of scores taken over all test cases you chose to solve.

For each test case, the score is the fraction of the total number of characters of the words and the number of characters of the text. Programs which receive a negative score for some test case will be considered incorrect.

Example

```
Input:
1
4
aaaa
aaaa
aaaa
bbaaa

Output:
case 1 Y
aaaabbaaaa
1
1
7
5

Score:
17/10 = 1.7
```

Bonus info: If score = *xxx.xxxaaa*, *aaa* means the number of test cases with Y answer.

Added by:	Michał Małafiejski
Date:	2005-05-18
Time limit:	30s
Source limit:	50000B
Languages:	All

SPOJ Problem Set (challenge)

755. Rectangles in a Square

Problem code: RIS

In two fundamental branches of modern science -- electronics and telecommunication -- progress is so marked that it may be perceived nearly as a natural power, controlling the fate of people and companies and transforming human life. Mainframes, computers, LAN, internet, built-in systems, Wi-Fi - generation upon generation of technology has sprung up within a time interval shorter than that of human life. Progress has its own life cycle, and periods of growth of semiconductor device production are interleaved with periods of decline, approximately once every five years. Experts believe that the main reason for such decline is the lack of new tools for Electronic Design Automation (EDAs), which can take full advantage of the latest technological achievements.

You are employed by the designers of a modern EDA and you have been asked by your boss to solve one of the stages of the design process. More specifically, you are to present a piece of software which, given a square-shaped board and a list of rectangular semiconductor devices, tries to place them on the board. No element may lie outside the board (even partially) or overlap with another element.

You are rather vague about the details of your task, and so (surprisingly) is your boss. "Just make sure the guys from Marketing can feature <<*Efficient Automated Semiconductor Placement*>> in our sales brochure" -- he says, and leaves you to it.

Eventually, you decide to pack as many of the listed chips as possible on the given board (leaving out those that simply won't fit in), and go off for the evening to the local whisky bar, wondering whether the next recession in the technological cycle won't come sooner than in 5 years' time...

Input

t - number of test cases, then *t* tests follow. [t <= 500]
In the first line of each test there is an integer N, and in the second line an integer K. N is the length of the side of the square [2 <= N <= 1000], and K is the number of available rectangles [1 <= K <= 10000]. Then exactly K lines follow, with 3 numbers on each of them: wi, hi, li. wi - length of rectangle [wi <= N], hi - height of rectangle [hi <= N], li - number of rectangles of this type [li <= 200000]. You may rotate a rectangle by a multiple of 90 degrees.

Output

For each test case output integer R - number of used rectangles, and then exactly R lines. On each of these lines output integer coordinates of opposite corners of rectangle xi1, yi1, xi2, yi2. Solution will be accepted if all rectangles won't intersect with each other and won't overrun the bounds of square.

Score

The score awarded to your program is the sum of scores for individual test cases. For individual test case you will receive points equals to area cover with rectangles divided by area of square. For test in which square doesn't have empty area, you will receive 4 points. If score = xxx.xxxaaa, aaa means the number of test cases with fully covered square.

Example

Input :

```
1
10
8
3 5 2
2 2 1
2 3 1
2 5 1
4 5 1
1 3 2
3 8 1
1 1 1
```

Output :

```
9
1 1 5 3
6 1 8 5
9 1 10 2
1 4 5 7
6 6 10 7
9 3 10 5
1 8 1 10
2 8 2 10
3 8 10 10
```

Example explanation :

```
Fig.1
```

On the figure rectangles marked with numbers in accordance with position in example output.
For this test case you will receive 4.000001 points, because square fully covered with rectangles.

Added by:	Roman Sol
Date:	2005-03-09
Time limit:	23s
Source limit:	50000B
Languages:	All
Resource:	ZCon 2006

SPOJ Problem Set (challenge)

758. Tetris AI

Problem code: TTR

In the very heart of a well known producer of microelectronic products, a mobile phone with a built in game of Network Tetris is being prepared for release. The owners of such mobile phones can arrange duels when at a small distance from each other. Data transmission between players is carried out using the Bluetooth protocol.

However -- now we are coming to the point -- it sometimes happens that there may be no other similar phone nearby and the player may need to play alone. For this purpose it is necessary to write a computer player (AI) with a very hard difficulty level.

The rules of Network Tetris are pretty simple :
The game has two playing fields, each with the rules of standard Tetris: figures of 4 blocks keep falling from the top of the field, and have to placed in such a way as to form horizontal lines. Once a line is filled up, it is removed and all lines above it are appropriately shifted downwards. There is however one difference with respect to standard Tetris -- a player receives additional penalty lines as soon as his opponent clears a line. The game is over when one of players fills his own field, either on his own or with his opponent's help, to such an extent, that the next figure cannot fully enter the field.
The width of the field is 10, and the height of field is 20. There are 6 types of figures in the game:
I (1) - [IMAGE]
L (2) - [IMAGE]
J (3) - [IMAGE]
Z (4) - [IMAGE]
S (5) - [IMAGE]
O (6) - [IMAGE]
[IMAGE]
It is your task to write a bot which starts with an empty field and, knowing the sequence of figures dropping on its field, plays in such a way as to do as much harm as possible to the opponent.

Input

t - number of test cases [*t* <= 150], then *t* tests follow.
Each test case starts with integer *N* equal to the number of figures which drop onto the field [10 <= N <= 50000]. Then N integers follow, numbered from 1 to 6 - denoting one of the figures, in the same position as they appear in the pictures. (Look at numbers in parenthesis).

Output

For each test you should output line *case 1 Y* if you wish to solve this test case or *case 1 N* otherwise. If you output *Y*, then exactly N lines must follow. Each of them should contain exactly two integers: A and X, where A is the clockwise angle of rotation in the given move, numbered from 0 to 3 (0 - 0 degrees, 1 - 90 degrees, 2 - 180 degrees, 3 - 270 degrees), while X is the horizontal coordinate of leftmost cube of the figure [1 <= x <= 10]. The *i*-th figure at input corresponds to the *i*-th figure at output. If the figure falls outside the field or the parameters have incorrect value, or any falling figure stops with at least one cube in a line of number larger than 20, then the solution will be judged as Wrong Answer.

Score

The score will be equal to the total number of cleared lines, taken over all test cases. For one cleared line your solution will receive 1 point, for two simultaneously cleared lines - 5 points, for three simultaneously cleared lines - 15 points, for four simultaneously cleared lines - 30 points.

Example

Input :
```
1
14
3
2
4
5
3
2
1
6
6
1
1
4
1
5
```

Output :
```
case 1 Y
0 1
0 8
0 1
0 8
1 7
3 3
1 5
0 7
0 9
0 1
1 6
1 6
0 1
1 4
```

Score :
```
score = 30 + 1 = 31
```

Added by:	Roman Sol
Date:	2005-04-13
Time limit:	30s
Source limit:	50000B
Languages:	All
Resource:	ZCon 2006

SPOJ Problem Set (challenge)

761. Minesweeper

Problem code: MSWP

The puzzle "Minesweeper" is based on the widely known game "Minesweeper", available for almost all MS Windows users, starting from version 2.0. The goal is simple -- to discover (or more precisely: uncover) the positions of all mines in rectangular grid. After a field without a mine on it is uncovered, the revealed value shows how many neighboring cells (at most 8) are occupied by mines. In the puzzle, just as in the game, you know the total number of mines on the board. But unlike in the game, you are not asked to risk your life by uncovering fields. Instead, you are given a list of uncovered fields (without mines, with numbers on them) and are requested to hazard a guess at the locations of all the mines.

Minesweeper

Input

t - the number of test cases, and then *t* tests follow [*t* <= 500].
Each test starts with three integers H, W, N equal to the height, width and number of mines on the grid, respectively [5 <= H, W <= 50] [1 <= N <= H*W]
Then exactly H lines follow, each of them consisting of W symbols.
The description of the grid consists of ASCII characters: '1'-'8' - the number of mines in neighbouring cells and '.' - a cell with unknown content. You can be sure that no other characters are present in the grid description.

Output

For each test you should output the text `char Y` in a separate line if you wish to solve this test, or `char N` otherwise. In the former case, you should then output a grid with the same size as that of the test, with mines placed on it in stead of some characters '.'. Mines are defined by the character 'X'. The number of mines should be equal to the number of mines N given in the test description.

Score

The score of your program is the total of scores awarded for individual test cases. The score for a test is equal to the number of cells around which mines are placed correctly (i.e. the number of mines is equal to the integer displayed in the cell). The number of cells for which mines are placed incorrectly is subtracted from this sum. Negative test case scores are treated as zero scores. If a test case is solved entirely correctly, the score is multiplied by 10.
Additional info: The score is given as xxx.aaa, where aaa is number of tests solved entirely correctly.

Example

Input:

```
2
8 8 19
........
2323..2.
..23...2
.3..33..
2.321...
....1.32
.3...3..
1...2..2
6 6 6
111.1.
1.2121
112.21
..112.
122111
1..1..
```

Output:

```
Y
X.X.....
2323X.2X
.X23XX.2
X3X.33.X
2.321X.X
.XX.1.32
.3...3X.
1X.X2XX2
Y
111X1.
1X2121
112X21
..112.
122111
1.X1XX
```

Score:

The first test is solved entirely correctly, for which 23*10 = 230 points are awarded. The score for the second test case is equal to 10-15 = -5, treated as 0. The total score is thus 230.001.

Added by:	Roman Sol
Date:	2006-01-01
Time limit:	30s
Source limit:	50000B
Languages:	All
Resource:	ZCon 2006

SPOJ Problem Set (challenge)

853. Delivery plan

Problem code: DELIVERY

Fry is an intergalactic delivery boy that spends all day trying to impress Lila. He wants to prove her that he's a smart guy so he wants to find routs between the different planets he must travel to by himself. Of course, he wants these routs to be as short as possible. Because of space pirates, meteor showers and other dangerous things, it's safe to travel only between certain pairs of planets. Help him find a good delivery plan so he can win Lila's heart. By Fry's observations, there is a safe route between any 2 destinations and the shortest one always passes through no more then 50 planets.

Input

The first line of input contains 2 integer, N and M (N<=50.000, M<=250.000), representing the number of planets and the number of direct routs between them. The next M lines contain 3 integers, X, Y, L (1<=L<=1.000.000) meaning that there is a safe connexion between planet X and planet Y of length L. The M+2-th line contains a number NQ (NQ<=5.000) and then follow NQ lines with 2 integers A and B meaning that Fry should take a package from planet A to B.

Output

Output must contain NQ lines, each line containing a number Nr (Nr<=100) and Nr integers representing the planets in the order you visit them. If Nr is -1 then that means that you want to skip this query.

Score

For each answered query you will receive (Best/Length)^2 points, where Best is the shortest length you can get when traveling between the planets you should have delivered a package and Length is the length of your route.

Example

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

Output:
3 1 4 5
4 3 2 1 4
3 3 2 1
```

Notes: This solution should receive 3 points. If you don't get AC on one of the 6 input cases, your score on that test case is 0 but you keep the points from the other inputs.

Added by:	Gogu Marian
Date:	2006-05-27
Time limit:	10s
Source limit:	50000B
Languages:	All
Resource:	

SPOJ Problem Set (challenge)

919. Prime checker

Problem code: PRIC

For this task you will have to check as many numbers as possible to see if they are prime. As not to make the problem I/O oriented, consider the numbers you should check in the following order: first take 1 and then construct the numbers in the sequence after the recursion: $a_i=(a_{i-1}+1234567890) \bmod 2^{31}$. Be careful not to use more than 4096 bytes of code.

Output

For each number you should write to output the digit "1" if the number is prime or the digit "0" if it is not prime.

Score

The score of your program will be the index of the first number in the sequence after which you do not have a correct answer. Because of some limitation you should not write more than 33 333 333 characters to output. If you reach this limit, your score will be adjusted in accordance to your runtime.

Example

Output:
```
01000000000000000000000000001000010000000001100000
```

should receive 50 points.

Added by: Gogu Marian
Date: 2006-08-29
Time limit: 25s
Source limit:4096B
Languages: All

SPOJ Problem Set (challenge)

1414. SuDoku Puzzle

Problem code: SUD

The name "Sudoku" is the Japanese abbreviation of a longer phrase, "suji wa dokushin ni kagiru", meaning "the digits must occur only once". Sudoku is a logic-based number placement puzzle. The objective is to fill a 9x9 grid so that each column, each row, and each of the nine 3x3 boxes contains the digits from 1 to 9. The puzzle setter provides a partially completed grid.

[IMAGE]

Unlike in magazines and newspapers, the digital representation of Sudoku a puzzle is a string of length 81, with all rows of the puzzle placed one after another. The representation uses ASCII symbols '1'-'9' for digits and '.' for an empty space. For example, the puzzle from figure above can be represented as:

7..25..98..6....1....61.3..9....1.......8.4.9..75.28.1.94..3.......4923.61.....4.

In this task you are to solve such puzzles automatically. The score will depend on the number of solved puzzles and on the speed of your solution. Some of the puzzles have multiple possible solutions, so be careful. A solution is correct if it satisfies the given puzzle. You can be sure that all given Sudokus are correct.

Input

t - the number of test cases; then *t* test cases follows. [*t* <= 500]
Each test case describes one SuDoku puzzle and consists of an 81-character-long string.

Output

For the i-th test case output a line containing **Y** if you want to solve the test case or **N** if you wish to leave it out. If you chose to solve the test case, in the next line output a sequence of exactly 81 characters corresponding to the solution for the i-th Sudoku puzzle.

Score

The score for this task calculated using the formula: *score = 200*total_solved/(200+time)*, where *total_solved* - number of correctly solved puzzles, *time* - running time of your program. If the score has the following form: *xxx.xxxaaa*, then *aaa* - is the number of correctly solved puzzles.

Example

Input:

```
3
..41..3.8.1....62...82..4.....3.28.9....7....7.16.8...562..17.3.3.....4.1....5...
1.......4....1.38.27.9.4...91.7...........5..86.4.5.9..3......8..9....2.4.......7
7..25..98..6....1....61.3..9....1.......8.4.9..75.28.1.94..3.......4923.61.....4.
```

Output:
```
Y
294167358315489627678253491456312879983574216721698534562941783839726145147835962
Y
198563274654217389273984615915726843347198562862435791731642958589371426426859137
N
```

Score:
In this case *total_solved* = 2. If the program runs for 10 seconds,
then the score of this solution will be equal to 1.905002

Added by:	Roman Sol
Date:	2006-03-30
Time limit:	30s
Source limit:	100000B
Languages:	All except: ERL JS
Resource:	ZCon 2007

SPOJ Problem Set (challenge)

1416. Electrification

Problem code: ELC

We are trying to develop the electrical power infrastructure in the small country of Byteland. For this purpose not far from each city we have built a nuclear power plant (NPP). We have also connected the nearest house to this NPP with a cable. The goal of this project is to connect all houses of each city to the source of electricity. Each house already connected to electricity become a source of electricity. Since there is a severe shortage of electrical cable, the total length of the electricity network should be kept as small as possible. In some places we can set up transformer/splitter boxes to which we can potentially connect several cables; all their endpoints are then considered connected.

Input

t - the number of cities; then follows the description of each of *t* cities. [$t <= 50$]
The description of each city begins with *N* - the number of houses in the city [$3 <= N <= 3000$]. Then exactly *N* lines follow, with two real numbers: *x, y* in each, representing the coordinates of a house. [$0.0 <= x, y <= 10000.0$]

Output

For each test case you must output a connected electrical net, e.g. all houses must be connected with each other, directly, through other houses or through transformers. For each test output integer *M* [$0 <= M <= N$] - the number of required transformers. On each of following *M* lines output the coordinates of the transformers *x, y* [$0.0 <= x, y <= 10000.0$]. Next output the number *K* which is equal to the number of required cables [$N+M-1 <= K <= (M+N)*(M+N-1)/2$]. On the following *K* lines output two integers *i, j* - indexes of houses or transformers. Indexes for houses begins with 0 and end with *N-1*, indexes for transformers begin with *N* and end with *N+M-1*.

Score

The score for the problem is given as: *total_score = (200+time)*(score_1+score_2+...score_t)/200*. In the above formula, *score_i* is equal to the length of the electrical cable used for electrification of the *i*th city, and *time* is the runtime of your solution.

Example

```
Input:
1
4
1.0 1.0
1.0 11.0
11.0 1.0
11.0 11.0

Output:
1
6.0 6.0
```

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

Score:

Suppose that the solution ran for 10 seconds. The length of the cable is *score_1* = 20*sqrt(2). In this case number of points awarded to the program will be equal to 29.698485.

Added by:	Roman Sol
Date:	2006-04-12
Time limit:	45s
Source limit:	60000B
Languages:	All
Resource:	ZCon 2007

SPOJ Problem Set (challenge)

1422. Digital Image Processing

Problem code: DIP

One of the most interesting problems of contemporary times is digital image processing to remove noise. A good solution to this problem is very important e.g. when developing digital cameras. In this task we are given a set of pictures, each of which is a grayscale image, transferred by some communication channel with failures. During the transfer some data was corrupted. A picture is defined as a rectangular matrix of integers from the range from 0 (black) to 255 (white). A number X at position (i, j) means that the pixel in the picture at the point with coordinates x = i and y = j has color RGB(X, X, X). The considered form of corruption generates noise in the following way: each pixel of the picture has its color replaced with probability between 2 and 20% by a random value from the range [0; 255].
Thus, you now receive a set of corrupted pictures, which were originally e.g. avatars, banners or photos. You are to restore the picture with maximum quality. The more exact a picture you obtain, the fewer penalty points you get.

Original Picture	Grayscale Picture	Noised Picture	Denoised Picture
Original Picture	Grayscale Picture	Noised Picture	Denoised Picture

Input

t - the number of test cases [*t* <= 60] (total number about 250), then *t* test cases follows.
Each test case begins with three integers: *Q*, *H* and *W*, denoting the noise probability for the generator in percent, and the height and width of picture respectively [2 <= *Q* <= 20], [10 <= *H, W* <= 200]. Then *H* rows follow with *W* integers in each of them.

Output

For each test case you must output a picture after noise reduction in following format: In the first line output the two integers *H* and *W*. Then *H* rows must follow with *W* integers in each of them. Each integer is the color value of a pixel after the restoration process.

Score

The score of the program will be equal to the sum of scores for each single test case + 1. The number of points for a single test case is calculated from the formula:
score = sqrt((x[1][1]1 - x[1][1]2)^2 + (x[1][2]1 - x[1][2]2)^2 + ... + (x[H][W]1-x[H][W]2)^2),
where x[i][j]1 is the color of the pixel at position i, j in the restored picture
and x[i][j]2 is the color of the pixel at position i, j in the original picture.

Example

Input:

```
1
6 20 20
 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 028 255 255
 255 255 200 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255
 255 000 255 255 096 255 079 079 079 079 255 045 255 045 255 068 043 043 043 255
 255 000 255 255 096 255 079 255 255 255 255 045 129 045 255 068 255 255 043 255
 255 000 255 255 096 255 079 255 255 255 255 045 255 045 255 068 255 255 189 255
 255 058 058 058 096 255 079 079 079 079 255 045 255 045 255 068 255 255 068 255
 255 076 255 255 096 255 079 255 255 255 255 185 255 045 255 068 255 255 068 255
 255 000 255 255 096 255 079 255 242 255 255 045 255 045 255 068 255 255 068 255
 255 000 255 255 096 255 079 079 079 079 255 045 255 043 255 048 048 048 048 255
 255 255 255 255 255 255 255 255 058 255 255 255 198 255 255 255 255 255 255 255
 036 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255
 255 255 100 195 002 167 002 002 002 002 002 002 002 002 002 002 255 255 255 255
 255 255 255 255 002 002 002 002 002 002 002 002 002 002 002 088 255 255 255 255
 255 255 255 255 002 002 002 002 002 002 046 002 002 002 002 002 255 255 143 255
 255 255 255 255 002 002 002 002 002 013 002 002 002 002 002 002 255 255 255 255
 255 255 177 255 255 255 255 104 255 255 255 255 255 255 255 255 012 133 255 255
 022 022 022 022 066 022 022 022 022 022 022 022 022 022 022 022 022 022 022 022
 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255
 255 255 255 079 079 079 079 079 079 079 079 079 079 079 079 079 079 255 255 255
 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 203 255 255
```

Output:

```
20 20
 253 253 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255
 254 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255
 255 000 255 255 096 255 079 079 079 079 255 045 255 045 255 068 043 043 043 255
 255 000 255 255 096 255 079 255 255 255 255 045 255 045 255 068 255 255 043 255
 255 000 255 255 096 255 079 255 255 255 255 045 255 045 255 068 255 255 043 255
 255 058 058 058 096 255 079 079 079 079 255 045 255 045 255 068 255 255 068 255
 255 000 255 255 096 255 079 255 255 255 255 045 255 045 255 068 255 255 068 255
 255 000 255 255 096 255 079 255 255 255 255 045 255 045 255 068 255 255 068 255
 255 000 255 255 096 255 079 079 079 079 255 045 255 043 255 048 048 048 048 255
 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255
 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255
 255 255 255 255 002 002 002 002 002 002 002 002 002 002 002 002 255 255 255 255
 255 255 255 255 002 002 002 002 002 002 002 002 002 002 002 002 255 255 255 255
 255 255 255 255 002 002 002 002 002 002 002 002 002 002 002 002 255 255 255 255
 255 255 255 255 002 002 002 002 002 002 002 002 002 002 002 002 255 255 255 255
 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255
 022 022 022 022 022 022 022 022 022 022 022 022 022 022 022 022 022 022 022 022
 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255
 255 255 255 079 079 079 079 079 079 079 079 079 079 079 079 079 079 255 255 255
 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255
```

Score:

```
Original picture:
20 20
 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255
 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255
 255 000 255 255 096 255 079 079 079 079 255 045 255 045 255 068 043 043 043 255
 255 000 255 255 096 255 079 255 255 255 255 045 255 045 255 068 255 255 043 255
 255 000 255 255 096 255 079 255 255 255 255 045 255 045 255 068 255 255 043 255
 255 058 058 058 096 255 079 079 079 079 255 045 255 045 255 068 255 255 068 255
 255 000 255 255 096 255 079 255 255 255 255 045 255 045 255 068 255 255 068 255
 255 000 255 255 096 255 079 255 255 255 255 045 255 045 255 068 255 255 068 255
 255 000 255 255 096 255 079 079 079 079 255 045 255 043 255 048 048 048 048 255
 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255
 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255
 255 255 255 255 002 002 002 002 002 002 002 002 002 002 002 002 255 255 255 255
 255 255 255 255 002 002 002 002 002 002 002 002 002 002 002 002 255 255 255 255
 255 255 255 255 002 002 002 002 002 002 002 002 002 002 002 002 255 255 255 255
 255 255 255 255 002 002 002 002 002 002 002 002 002 002 002 002 255 255 255 255
 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255
 022 022 022 022 022 022 022 022 022 022 022 022 022 022 022 022 022 022 022 022
 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255
 255 255 255 079 079 079 079 079 079 079 079 079 079 079 079 079 079 255 255 255
 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255 255

score = sqrt(2^2 + 2^2 + 1^2) + 1 = 1 + 3 = 4 (three pixels differ in the top-left corner)

Original picture: [IMAGE] Noisy picture: [IMAGE]
```

Added by:	Roman Sol
Date:	2006-12-19
Time limit:	11s
Source limit:	100000B
Languages:	All
Resource:	ZCon 2007

SPOJ Problem Set (challenge)

1423. Digits of SQRT(2)

Problem code: SQRT2

In this task you are to find as many digits of the square root of 2 as possible. You have to make it within the limit of time and source code length.

Input

There is no input for this problem

Output

The output must contain as many digits of the square root of 2 as possible (max = 2000000)

Score

The score awarded to your program will be the first position of the digit where the first difference occurs.

Example

Output:

```
1.41421356237309504
```

will be awarded with 19 points.

Added by:	Roman Sol
Date:	2007-01-07
Time limit:	20s
Source limit:	4096B
Languages:	All except: LISP sbcl LISP clisp CLPS
Resource:	ZCon 2007

SPOJ Problem Set (challenge)

1481. Yet another computer network problem

Problem code: PT07E

ACRush and Jelly are practising in the computer room for next TCO. Suddenly, they found the network is very slow. After a few diagnoses, they realized that there are many redundant wires. So they plan to repair the network, change it to an optimal tree topology. And they can't spend too much money to purchase wires. Then.. too easy? Are you thinking about minimum spanning tree?

But the real trouble is the connectors have their own limitation. They can only allow one computer connects with at most *B* computers.

There are totally 10 cases, arranged in increasing order of the size of *N* (number of computers). Weight of case i-th is w[i] = i. We define *infinity* = 4 * 10^9. And in a tree, let's call number of computers that computer i connects with is *degree* of computer i.
For case i-th you must show us a satisfied tree with total cost C[i] and maximum degree M[i], considering all computers of that tree.
The formula to compute score is as below:
With case i-th:
If your M[i] <= B then Score[i] = w[i] * C[i]
If your M[i] > B then Score[i] = (w[i] + 10) * C[i] * M[i]

To make the challenge more interesting, with a simple brute force program, we generated 10 upper bound degrees U[i] (1 <= i <= 10) for each of 10 cases.
For any case i-th:
If your M[i] > U[i] then Score[i] = infinity
Finally, TotalScore = (Score[1] + Score[2] + ... + Score[10]) / 10
Try to minimize the *TotalScore*.

Input

First line contains 3 integers *N, M, B* -- number of computers, number of pairs of computers can be connected and the maximum number of computers that a computer can connect with. (1 <= *N* <= 10^4, 1 <= *M* <= 10^5, 1 <= *B* <= N)
Next *M* lines, line i-th contains a triple (u[i], v[i], c[i]) -- means if we want to connect computers u[i] and v[i] we should purchase a wire, cost c[i] (1 <= u[i], v[i] <= N, 1 <= c[i] <= 20000). The wires are bidirectional.

Output

The first line contains 2 numbers --- total cost of your tree and the maximum degree in all computers of that tree. Next, print *N*-1 lines, corresponding to *N*-1 edges of the tree, each edge on one line, forms *u v*.

Example

Input:
```
3 3 2
1 2 1
2 3 1
1 3 5
```

Output:
```
2 2
1 2
2 3
```

Added by:	Thanh-Vy Hua
Date:	2007-04-07
Time limit:	10s
Source limit:	50000B
Languages:	All
Resource:	Co-author Amber

SPOJ Problem Set (challenge)

1492. Lexicographic sort

Problem code: SLEXSORT

Given alphabet *A* and a list of words, sort the list according to the lexicographic order induced by *A*.

Input

The first line of input contains *t*, the number of tests.

Each test begins with a line with alphabet *A*, which consists of lowercase letters arbitrary chosen from the Latin alphabet. The next line contains an integer *n*<100 000 - the number of words. The subsequent *n* contain one word each, which is not longer than 1 000 letters. Additionally, you can assume that the total number of letters in all words of each test does not exceed $4*10^6$.

There is an empty line after each test.

Output

For each test output the sorted list of words in successive lines.

Score

The score is equal to the length of the source code of your program.

Example

Input:
```
2
re
3
ere
rer
re

balujemy
5
bel
luba
lej
bal
leje
```

Output:
```
re
rer
ere

bal
```

```
bel
luba
lej
leje
```

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

---

Added by:	Łukasz Kuszner
Date:	2007-04-10
Time limit:	30s
Source limit:	50000B
Languages:	All

SPOJ Problem Set (challenge)

1558. Math II

Problem code: MATH2

Background

This is a mathematical(?) problem. See problem MATH1 and AMATH.

Input

The first line of the input contains a single integer c(1<=c<=2).

The second line contains 3 integer numbers separated by single spaces, D(1<=D<=1 000 000 000),n(1<=n<=700),m(1<=m<=700).

n lines follow, each contains m space-separated integer numbers.The number which is in the i-th row and j-th column is defined as a(i,j).

Output

You should output n lines, each contains m integers,which is either 0 or 1.We define the number in the i-th row and j-th column of your output b(i,j).

Score

If your answer is valid, the score of your program equals to the sum of the scores of each test case multiply 10000.

The score for each test case is calculated in the following way:

a) c=1

The score S equals to

[IMAGE]

If S>1.5, your score will be multiplied by 10000.

b) c=2

The score S equals to

[IMAGE]

If S>2, your score will be multiplied by 10000.

Example

Input #1:
```
1
7 3 4
1 6 4 6
7 0 3 3
2 5 1 5
```

Output #1:
```
0 1 0 1
1 0 1 0
0 1 0 1
```

Input #2:
```
2
7 3 4
1 6 4 6
7 0 3 3
2 5 1 5
```

Output #2:
```
0 1 0 1
1 0 1 0
0 1 0 1
```

Score:
```
11428.5714
```

Warning: large input/output data, be careful with certain languages.

Blue Mary's Note: Some unofficial tests were added.

Blue Mary's Another note: the score system has been changed to avoid Wrong Answer.

Added by:	Blue Mary
Date:	2007-05-16
Time limit:	10s
Source limit:	50000B
Languages:	All except: C99 strict
Resource:	IOI Chinese Team Selection Contest(CTSC) 2007

SPOJ Problem Set (challenge)

1690. Intercept

Problem code: INTER

Long long ago, so long ago, no body knows how long ago, there was a huge galactic war. There was a very powerful general, General Ramuk, who had every possible soldier and scientist under him. One of his scientists reports that he had intercepted a transmission that he believes is from the aliens. A group of experienced cryptographers believe that in the following hypotheses:

1. The aliens follow use binary system for representing numbers
2. Their '0' should be interpreted as '1' and vice-versa
3. The message is encoded as follows: [32bits of n1][16bits of n2][n2 bits of n3]
4. The retransmission they expect is: [32bits of the remainder] when **n3** is divided by **n1**.
5. All the numbers are written, Most significant bit first.
6. Remainder must be communicated in the following format:
   [remainder for 1st instance]
   [remainder for 2nd instance]
   in their own number system, without leading '1's ('0's).
7. The number of instances is about **200**.

The first transmission was completed. Ramuk is eagerly waiting for the second transmission, which must be replied. Being such a simple problem, he asks you to write a program to do the same.

He says: *"Nee evalovu chinnadha codea ezhudhariyo avalovu parisu onnakku kaathhirriku ",* which translates to: *"The smaller the code you write, that much reward is awaiting you...".*

You want to save the world from a probable Alien Invasion, and get as much money as possible.

Constant bit length numbers will be prefixed by '1's ('0's in their notation).

Scoring

The scoring for this problem is the **length of the source code**.

Sample Input

NOTE: The colons (:) and newlines are for clarity

```
11111111111111111111110001001110:1111111111101111:0011100111000100
11111111111111111111110001001011:1111111111101100:0100100001000011111
```

The actual input will be like:

```
111111111111111111111100010011101111111111101111001110011100010 0
11111111111111111111110001001011111111111110110001001000010000 11111
```

(new line is again, for clarity)

Sample Output

```
0101101001
0010001111
```

Explanation

```
n1=945
n2=16
n3=50747
output=662

n1=948
n2=20
n3=376288
output=880
```

Warning: Large Input.

Added by:	Vimal
Date:	2007-07-16
Time limit:	5s
Source limit:	500B
Languages:	All
Resource:	Own Problem

SPOJ Problem Set (challenge)

1711. Greatest Common Divisor

Problem code: GCD

Consider the decimal representation of a natural number N.
Find the greatest common divisor (GCD) of all numbers that can be obtained by permuting the digits in the given number. **Leading zeroes are allowed.**

Input

Every line of input contains an integer, representing the original number N(0 < N < 10^250).

Output

For every test case, print the GCD of all numbers, which can be obtained from the given one by permuting the digits.

Score

Score is the length of your source.

Example

```
Input:
21
3
Output:
3
3
```

Added by:	Jin Bin
Date:	2007-07-27
Time limit:	2s
Source limit:	2000B
Languages:	All except: C99 strict
Resource:	from an ACM/ICPC regional contest

SPOJ Problem Set (challenge)

1742. Brainf_ck

Problem code: BRAINF_K

brainf*ck is the ungodly creation of Urban Mller, whose goal was apparently to create a Turing-complete language for which he could write the smallest compiler ever. http://en.wikipedia.org defines it as "a computer programming language designed to challenge and amuse programmers, and is not suitable for practical use. Its name has been variously euphemized, as in brainf*ck."

A brainf*ck program has an implicit byte pointer, called "the pointer", which is free to move around within an array of 32768 bytes, initially all set to zero. The pointer itself is initialized to point to the beginning of this array.

The brainf*ck programming language consists of seven commands, each of which is represented as a single character. Note: "Industry standard" brainf*ck actually has eight commands, but for the purposes of this problem one command was intentionally omitted.

COMMAND	OPERATION
>	Increment the pointer. Incrementing a pointer value of 32767 results in a pointer value of 0.
<	Decrement the pointer. Decrementing a pointer value of 0 results in a pointer value of 32767.
+	Increment the byte at the pointer. Incrementing the byte value 255 results in the byte value 0.
−	Decrement the byte at the pointer. Decrementing the byte value 0 results in the byte value 255.
.	Output the character whose ASCII value is the byte at the pointer
[	Jump forward past the matching] if the byte at the pointer is zero.
]	Jump backward to the matching [unless the byte at the pointer is zero.

For this problem, you will write a program that reads in, parses and executes a brainf*ck program.

Input

Input contains exactly one program. The program consists of one or more lines of brainf*ck commands. Your program should ignore any illegal characters (I.E. any character not in the set: <>+-.[]), If a percent sign (%) is encountered during parsing, the remainder of the line should be discarded. This constitutes a comment. The maximum number of commands in a brainf*ck program is 128000.

Output

Your program should output the output generated by the brainf*ck program. The only possible parsing error that can occur is if there is an unmatched [or] in the brainf*ck program. If your program encounters such an error, it should simply print "**COMPILE ERROR**" instead of executing the program. All brainf*ck programs will use no more than the specified 32768 bytes of memory.

Score

Score is the length of your source.

Example

Input:
```
++++++++[>+++++++++ % hello-world.
<-]>.<+++++[>++++++<-]>-.+++++++..
+++.<++++++++[>>++++<<-]>>.<<++++[>
------<-]>.<++++[>++++++<-]>.+++.
------.--------.>+.
```

Output:
```
Hello World!
```

Input:
```
[]]
```

Output:
```
COMPILE ERROR
```

A super hard test case was removed.

Added by:	Jin Bin
Date:	2007-08-20
Time limit:	10s
Source limit:	512B
Languages:	All except: C99 strict
Resource:	ACM/ICPC Greater New York Regional Contest 2005

SPOJ Problem Set (challenge)

2004. Analyse Simple Arithmetical Expressions

Problem code: EXPR2

You are to write a program to analyse some simple arithmetical expressions. The BNF form of the defination of the expression is below.

```
<expression>::=<num><oper><num>
<num>::=0|1|2|...|99
<oper>::=+|-|*
```

Tip: You may find this problem is like the problem GALAXY very much. To get round of the programming problems of using Brainf**k, Whitespace or Intercal, you must use C/C++/Pascal/Java to do the programming. Seems easy? Now there is an additional objective: **there must not be any semicolons ";" in your program!!!**

Input

Multiple test cases, the number of them T is given in the very first line, T<=99.

Each test case contains one line with a correct expression, without leading or trailing spaces.

Output

For each test case you should output one line contains the result of the expression without any leading zeros. You may assume this number is always a non-negative one.

Example

```
Input:
3
6*7
67-25
31+11

Output:
42
42
42
```

Score

Thanks to Jin Bin's suggestion, I've changed this problem from a classical one to a challenge one. Suppose the number of non-whitespace characters(ASCII 33 - 126) in your solution is **K**, the your score is **floor(K^3/1000)+1**.

Note

The judge had something wrong and it has been fixed on Jul.1, 2008. Please accept my apology.

Added by:	Blue Mary
Date:	2007-11-03
Time limit:	17s
Source limit:	17000B
Languages:	C C++ PAS gpc PAS fpc JAVA
Resource:	A copy of problem GALAXY, with language restrictions modified

SPOJ Problem Set (challenge)

2335. Error Minimization

Problem code: ERRORMIN

In this problem, you have to reverse an algorithm. The known algorithm takes inputs A and C, and produces output B. Your task will be to take A and B as inputs, and produce C.

The known algorithm can be expressed several ways:

As a mathematical expression: (image)	As a repeating pattern:
	B[0] = A[0] * C[0]
	B[1] = A[0] * C[1] + A[1] * C[0]
	B[2] = A[0] * C[2] + A[1] * C[1] + A[2] * C[0]
	B[3] = A[0] * C[3] + A[1] * C[2] + A[2] * C[1] + A[3] * C[0]
	B[4] = A[0] * C[4] + A[1] * C[3] + A[2] * C[2] + A[3] * C[1] + A[4] * C[0]
	B[5] = A[0] * C[5] + A[1] * C[4] + A[2] * C[3] + A[3] * C[2] + A[4] * C[1] + A[5] * C[0]
	...
		As a program:
		`for(int iB = 0; iB < len; iB++) {`
		`  for(int iC = 0; iC < len; iC++) {`
		`    B[iB + iC] += A[iB] * C[iC];`
		`  }`
		`}`

When you produce a solution (C), you can check your work by running the known algorithm on it, and the output value (B) will be called Bprime. If B equals Bprime, then your program is working.

However, errors will be introduced into B, such that it is impossible to produce a matching Bprime. In that case, your program must find the *closest match*. Here is the criteria for a success:

1. When given perfect inputs, B must match Bprime exactly.
2. The sum of the deltas between B and Bprime (sum(B[n] - Bprime[n])) must equal zero:
   (image)
3. The sum of the differences between B and Bprime (sum(abs(B[n] - Bprime[n]))) must be as low as possible:
   (image)

Your score is the sum total of the errors between B and Bprime during it's test. The goal is to get as low a score as possible.

Notes:

1. The lengths of arrays A, B and C are the same.
2. The arrays are floating point numbers.

Input

Each request is sent in two lines; each of the values for A separated by spaces, and each of the values for B separated by spaces. More than one request will be made.

Output

Your program must output the value for C on one line, separated by spaces, in response to each pair of input lines.

Example

```
Input:
1 2 0 1 0 0
5 10 1 7 0 1
0 1 0 1 0 0 0
0 0 1 0 -1 0 0
0 1 0 1 0 0 0
0 0 0.5 0 1 0 0

Output:
5 0 1 0 0 0
0 0 0 0 0 0 0
0 0.75 0 0 0 0 0
```

All source code submitted for this problem must be free of intellectual property restrictions, and becomes the intellectual property of Michael Mudge.

To the first user who solved this: You are entitled to the prize, but I have no contact information!

Added by:	Michael Mudge
Date:	2008-01-13
Time limit:	10s
Source limit:	50000B
Languages:	C C99 strict C++ JAVA C#

SPOJ Problem Set (challenge)

2471. Magic Program II

Problem code: MAGIC2

Task

Write a program with minimum length to print the output(119729 bytes).

Score

The score of your program is the length of your source code in bytes.

Notice: Sorry to the Python and PHP user, since these languages support compress functions, there are not allowed. Some submissions in these languages are judged as Wrong Answer now.

Added by:	Jin Bin
Date:	2008-02-21
Time limit:	10s
Source limit:	100000B
Languages:	C C++ PAS gpc PAS fpc JAVA C#
Resource:	own problem

SPOJ Problem Set (main)

2624. JawBreaker Game

Problem code: JAWB

The goal of this popular game is to get maximum number of points, by removing aligned pieces of the same color. Pieces can only be removed if they touch each other by an entire side. The more pieces you remove in one turn, the more points you get. The number of points in one turn is described by the following formula: N*(N-1), where N is the number of pieces (for example 2*(2-1)=2, 10*(10-1)=90). If you remove pieces from the middle of the field, then all pieces located higher fall down and occupy the empty spaces. The game is finished when no pieces which can be removed from field remain.

JawBreak game

In this problem you will be given a field and pieces on it. Your goal is to obtain the maximum number of points.

Note: You can practice a little and plan your strategy with this on-line game [the on-line game is slightly different from the one described above]: http://www.bigfrog.net/jawbreaker/

Input

t - the number of tests, then *t* tests follow. [$t <= 500$]
Each test starts with 3 integers: *H* - the number of rows of the playing field, *W* - the number of columns of the playing field and *C* - the number of different colors of pieces. [$4 <= H, W <= 50$] and [$3 <= C <= 20$]. Then follow *H* rows with *W* numbers in each, separated by spaces. Each number is in the range from 0 to *C-1* and describes the color of a piece.

Output

For each test you must output the letter "Y" if you want to solve this test, or the letter "N" otherwise. If you output Y, you must output a set of lines with 2 integers *x, y* in each. These integers define rows and columns in the field. [$0 <= x < H$], [$0 <= y < W$]. Coordinates are counted from the upper left corner of field. After your last move output the line -1 -1. You'll receive status Wrong Answer if your coordinates are outside the field, or point to an empty space, or to a single piece.

Score

The score received for this problem is calculated as follows: *score = 200*total_score/(200+time)*, where *total_score* - sum of points received for each playing field, *time* - process time for your solution in seconds. The *score* for a playing field is calculated as *(C*C*base_score)/(H*W)*, where *S* - number of different colors, *H* - field height, *W* - field width, *base_score* - number of points calculated as in the description of problem.

Example

Input:

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

Output:

```
Y
1 0
1 0
3 2
-1 -1
```

Explanation:

```
Initial field:

0 0 1 1
1 1 2 2
0 1 2 0
0 1 1 2

After the first turn (removed 5 "ones"):

. . . 1
0 . 1 2
0 . 2 0
0 0 2 2

After the second turn (removed 4 "zeros"):

. . . 1
. . 1 2
. . 2 0
. . 2 2

After the third turn (removed 3 "twos"):

. . . .
. . . 1
. . . 2
. . 1 0
```

Score:

In this case *base_score* = 5*(5-1) + 4*(4-1) + 3*(3-1) = 20 + 12 + 6 = 38,
total_score = (3*3*38)/(4*4) = 21.375. Let's suppose that ittakes 10 seconds to finish calculations, then *score* = 200*21.375/210 = 20.357143

Added by:	Roman Sol
Date:	2007-07-19
Time limit:	50s
Source limit:	50000B
Languages:	All
Resource:	ZCon 2008

SPOJ Problem Set (main)

2628. Markov Algorithm

Problem code: MAR

Markov schemes are well known field of study in theory of algorithms. Let’s define some slightly simplified schemes. We have some character set (alphabet), and a string composed of these symbols. Besides, we have rules of replacement, and each rule specifies the substring to be replaced and the replacement string. The replacement string can be empty. The order of rules is fixed. Everything functions as follows. Rules are looked through in input order. The first rule which can be applied, is applied exactly once (the given replacement which is described by the rule is carried out), and then the operation cycle repeats (the list is looked through anew). If there are many substrings in the line which satisfy a given rule then only the first (i.e., the leftmost) will be replaced. The process comes to an end if after the last rule application the line has not changed. During the process of replacements the length of the initial string can change, increasing or decreasing.
Example:
Let’s consider the string *ababaab*
and the set of rules
ab->c
cc->ab
As a result of the application of rules the initial string will be transformed into "cac". The following intermediate results will appear: *ababaab, cabaab, ccaab, ccac, abac, cac.*
That’s all. And now some tasks.
Task 1: The initial non-empty string was obtained from an ordinary arithmetic formula by the deletion of every symbol except brackets. You are to write the set of rules which transforms this string into the string "RIGHT" or "WRONG" depending on the correctness of brackets positions, following rules of bracket use in arithmetical expressions. For example, the string ()((())()) must be replaced by the string RIGHT, while the string (() must be replaced by the string WRONG with the same replacement rules.
Task 2: The initial string represents an arbitrary string of the following type [integer1]+[integer2]=?, where [integer1] and [integer2] are decimal representation of some positive integers. You are to write a set of rules which translates the initial string into a string of the following type: [integer1]+[integer2]=[sum], where [sum] is the decimal representation of the sum of two numbers integer1 and integer2. For example, the string 2+2=? must be replaced by 2+2=4, and the string 25+76=? must be replaced by 25+76=101. The integers can be up to 100 digits in decimal notation.
Task 3: You are given a string which consists of uppercase letters (A-Z), and ends with the "?" sign. You are to output this string in reverse order without the "?" sign. For example, the string ABBCD? must be transformed into DCBBA
Task 4: You are given a binary integer which consists of 0s and 1s. You are to write it as a string of letters "z", and the total number of such letters must be equal to the given binary number. For example, 110 must be translated by the algorithm into "zzzzzz"
Task 5: You are given a string which consists of uppercase letters (A-Z), and ends with the "?" sign. You are to output this string in non-descending order without the "?" sign. After applying the rules to DFAAS?, the string should look like this: AADFS
Task 6: You are given two decimal positive integers separated by the "_" symbol and followed by "=?", for example 30_42=?. You are to find and output the value of the greatest common divisor for this pair. For example for 30_42=? it’ll be 6.
<u>Note</u>: The limit on the increase of the string during the work of the algorithm is 100000 symbols.

Input

There is no input data for this problem.

Output

You should output your solutions to every task one by one. If you don't want to solve some particular task just output 0, otherwise output the length of your solution N. Then output exactly N lines *abc->def*, where *abc* is the substring for replacement, and *def* is the substring by which it will be replaced.

Score

For each correctly solved task you gain 1 point and additionally 1/N bonus points, where N is the length of the proposed solution.

Example

Output:
```
0
0
0
5
a->b
b->c
d->v
g->l
l->a
0
0
```

Score:
```
In this case (if the set of rules leads to a correct answer)you'll get 1 + 0.2 = 1.2 points for the 4-th task.
```

Added by:	Roman Sol
Date:	2007-09-03
Time limit:	10s
Source limit:	50000B
Languages:	TEXT
Resource:	ZCon 2008

SPOJ Problem Set (challenge)

2629. Robo Eye

Problem code: EYES

A robot which helps to translate old papers into digital format is being prepared for mass production. But it requires special software to work efficiently. All scans which will be analyzed by the robot use one of the standard fonts: Arial, Courier, or Times New Roman. As the press is not ideal and pages can be rotated, some of the scans will be rotated by some degree. Imperfections of the robot's camera ("eye") can also cause some noise. Uniform noise can be up to 2% of the number of pixels. All analyzed documents are in uppercase English. The total number of letters in the robot's eye ranges from 3 to 6. The matrix in the robot's eye is monochrome and its size is 200x200 pixels.

Input

t - the number of tests [*t* <= 500], then *t* tests follows. Each test consists of 200 rows with 200 chars in each of them. Characters can be '.' and 'X', where '.' means the white color of the page, and 'X' is the black color of words.
The input data was generated using an on-line generator. The generator outputs data in 2 formats:
1) as a picture
2) as text
The datasets used for testing are such that all letters of the word are contained inside the picture.

Output

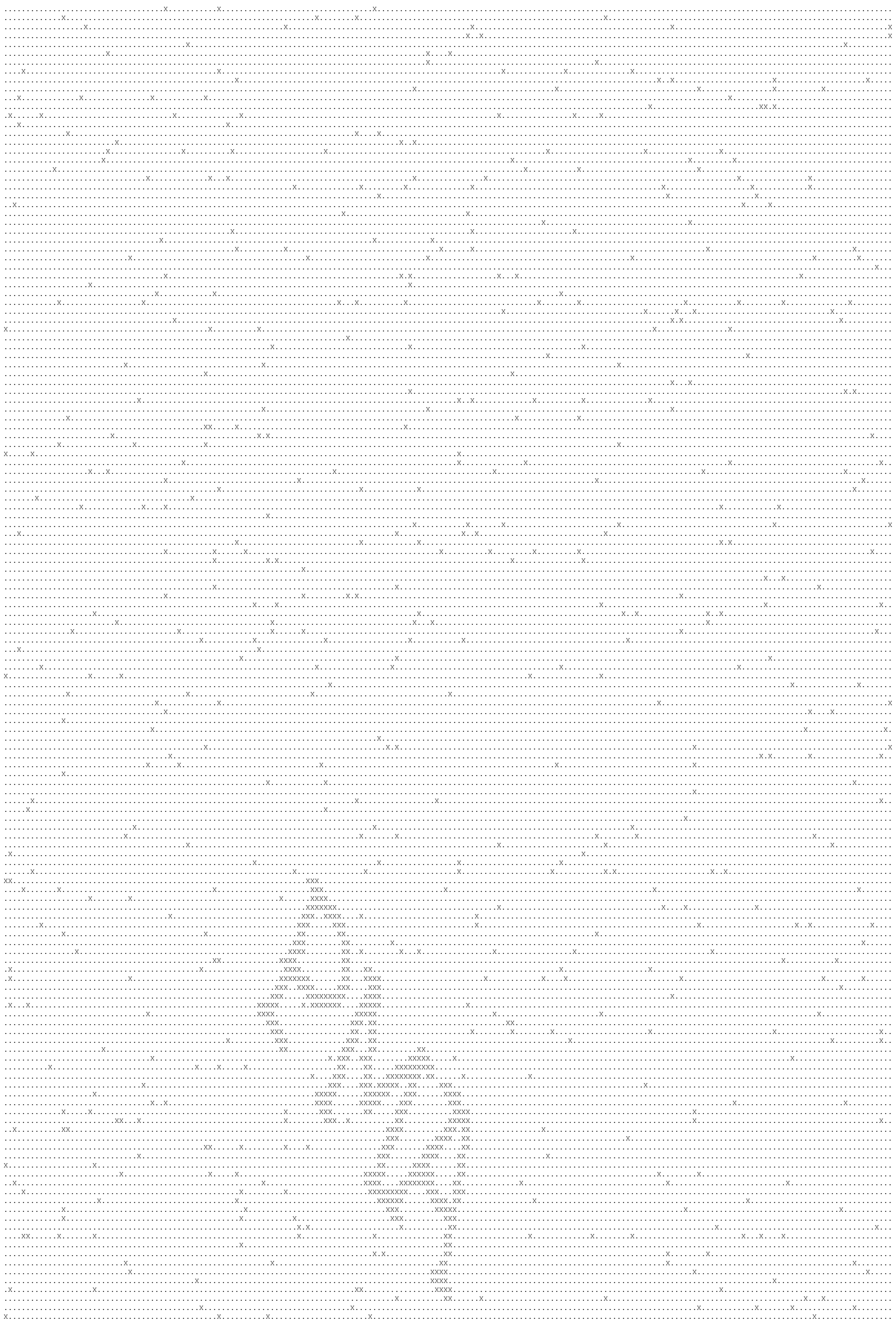
For each test output the recognized word in a separate line.

Score

The number of points you'll receive for each image will be equal to the number of letters of the word, provided that it is correctly recognized.

Example

```
Input:
2...........X...........X..............................................................X.................................................................................................................
.............................................X...........................X......................X.........................................X...............................................X.............
........................................................................................X...............................................................................................................
X............X....X...............................................................................................................X......X..................X......X.......................X..X.........
.........................................................X.........X.............................X.........................X.....XXX........X................................X..........................
................X.................................X...X....X......X.........................X...................................XXXXX.........................X.........................................
..............................X.X...........X.......................................................X......X....X.........X....XXXXXXX.........................................X........X...............
..............................................X...................................................................X..X.........XXXXXXXX.................................................................
......................................................................................................................X..........XXXXXXX................................................................
.....................................................................X............................................................XXXXXXX.......X......................X................................
...................................................X...........................X.....X...........X........X........................XXXXXXX...........................X..................................
.......................X...............................................................X............................................XXXXXXX.......................X...X.................................
.........X.................................................................................................X......X..................XXXXXXX.......................................................X....
.............X.........X.............................................X.....X...X.............X........................................XXXXXXXX...........................X...............X..............
...X....................................X.........................................................................................X....XXXXXXXX.........................................................
......................................X..........X...................................................................................X..XXXXXXXX.......................X................................
...........................................................................X.............................................................XXXXXXXX.......................................................
.....................................................X..X.........................................X.................XX...........X........XXXXXXXX......X.....................................X.........
......................X............................................................................................XXXXXXX.................XXXXXXXX....XX.................X.............................
....................X...........................................................................X..............X..XXXXXXXXXXXXX.............XXXXXXXX......X.....................X.......................
................................................................X................................................XXXXXXXXXXXXXXXXXXX.........XXXXXXXX...............X...................................
.............................................................................X......X..............X..............XXXXXXXXXXXXXXXXXXXXXXX.....XXXXXXXX..........X.......................X...............
......................X...........X.............X.............X....................................................XXXXXXXXXXXXXXXXXXXXXXXXXX..XXXXXXXX..............................................X..
....................................................................................................................XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX....................................X...........
.X...................X.............................................................X.................................XXXXXXX...XXXXXXXXXXXXXXXXXXXXXXXXXX..X.........................................X..
................................X..................................................................................X..XXXXXXX.......XXXXXXXXXXXXXXXXXXXXXX...........X..................................
..................................X..............................................................................X.....XXXXXXX...........XXXXXXXXXXXXXXXXX...........................................X..
.......................................X................................................................................XXXXXXX..............XXXXXXXXXXXX...............................................
.....XX...........................................................................X......................................XXXXXXX..................XXXXXX................................................
..............................................................................................................X...........XXXXXXX..................................................X....................
................X.....................................X.......................................X............................XXXXXXX.X.........X.............X......................X..............X......
......................................................................................................X.....................XXXXXXX.................................X.............................X...X.
....................X..X..................................................X...........................X......................XXXXXXX...............X.....X............................X.................
..................................................X....X.......................................................X..............XXXXXXX...................................................................
...........................X.............................................................X.X...................................XXXXXXX..................................................................
........X.X.......X..........................................................................X......................X...........XXXXXXXX..............................X.................................
.X............................................................................X...................XXX............................XXXXXXXX.........................................X.....................
........X..................................X.....................................................XXXXX............................XXXXXXXX........................................X.................X...
```



```
.................................................X...................X..........X.....X...............................................................X...........X.....................................
.....................................................................................................................................................X..................................................
....................................................................................................................................X..............................................X....................
....................................................................................................................X.........................................................X.......................X.
......X..........................................X...............XX.....................................................................................................................................
.............X..............X............................................................X..............................................................................................................
X............X...X................X...............................................................X.....................................................................................................
........................................................................................X................................................................................X.................X............
.......................X.................X.....................X...............................XX......................................................X..............................................X.
X.XX...........................................................X.....................X............................X..................................X..................................................
.............................................X.....................................................X..........................X........................................X................................
........................................................................X...........X......X............X........................................................................X......................
.........XX......X.................................................X...............................X...........X..X...........................................X..............X..........................
........................................................................................................................................................................................................
.................................................X.................X......................................................................X.............................................................
................................................................................................................................................................X.......................................
......................X.............................................................................................................X..........................................................X........
..X..X...X.......................................................X.....................X.........................................................................X...................X..................
........X.............................................X..............................................................................................................................X..................
.............................................................................................X..............................................................................X....................X......
..........................X.......X................................................X....................................................................................................................
....X..............X.....................X.......................X.................................................................X....................................................................
...........................................................................................................X.....................................................................X......................
........................................................................................................................................................................X.............X..X..............
..X........................................................................X...........................................X........X.X........X...........X................................................
...............X.................X.......X.............................................................................................................X................................................
.........X....X................................X...................................................................................................X.......................................X............
..X.............X.....................X.............................................................................X..X...........................X............................XX..X...................
.....X..........................X......................................................X.............................X..X...............X....X.......................X.............X....................
.......X...............XX.....................X.........................................................................................................................................................
...........................X..............................................................X...............................................X...................X.........................................
.............................................................................................................................................X..........................................................
....................................X...........................................................................X..............X........X.....X.X.......................................................
...............X...................................X.............................X...................................X...............................................................X..................
........................................................................................................................................................................................................
..................................................X.................................................................................................X...................................................
........................................................................................................................................................................................................
......................................X......................................................................................................................X..........................................
......................................X...........................................................................................................X.....................................................
......X................X.........................................................................................................................................................X......................
.X.............X......................................................X.............................................................X...................................................................
......X..........................................X......................................................................................................................................................
...........................................................................................................X...........................X....................X..........X....X...........................
..............................................................................................................................X.........................................................................
..............................................X..X....................................................................................................................................X...............................
.......................X................................X...............................X...X........................................................X..................X...............................
.............................................................................................................X....................X...................X.............X..........................X....X...
........................................X.....................X.......................................................X........X......X....................X............................................
............................................................X........X..X....................X..........................................................X...........................X...................
........................................................................................................................................................................................................
.....X.......X.......X.......................................................X........X.........................................................X.....................................X.................
..................X.............................................................X...........................................................X..........X.......X........................................
..X..............................X.........................X..............................X.....................X.......................................................................................
.....................................................................................X.................................................................................................X................
....................................................X...................................................................................................................................................
...................................X................X..........X...............................................................X..................X...........X.....................X.......X...........
..............X................X..................................................................................X...........X...X.............................X....................X..................
...............X........................................................................................................................................................................................
....................................X................X..........................................................................................................X..............................X........
....................................................................................................................................................................X...................................
..........................................................................................X.............................................................................................................
........................................................X.......................X..........................X............................................................................................
............X......X.......X.........X.........................................................................X.....X........................................X......................................XX..
..............................X.....................................................................................................................................X...................................
...X.........X...........................................XX............................................................................................................................................

......................................................X..................................................................................................................................X..............
```
Output:

```
WNILN
PML
```

Score:

score = 5 (first test is correct and contains 5 letters, the second test has an error in the last letter)

Examples of images:

[IMAGE] [IMAGE] [IMAGE] [IMAGE] [IMAGE] [IMAGE]
Additional pictures can be generated with the on-line generator.

Added by:	Roman Sol
Date:	2008-02-13
Time limit:	50s
Source limit:	50000B
Languages:	All
Resource:	ZCon 2008

SPOJ Problem Set (challenge)

2659. Carl

Problem code: CARL

Professor Octastichs has invented a new programming language, Carl. An expression in Carl may be a positive or negative integer, or may be of the form **(p e1 e2)** where **p** is a real number between 0 and 1 (inclusive) and **e1** and **e2** are Carl expressions. An integer represents itself and **(p e1 e2)** represents **x + y** where **x** is the value of **e1** and **y** is the value of **e2** with probability **p**, otherwise it represents **x - y**.

Given a Carl expression, what is its expected value?

Input

Input consists of several Carl expressions, one per line, followed by a line containing ().

Output

For each expression, output its expected value to two decimal places.

Score

Score is the length of your source program.

Example

```
Input:
7
(.5 3 9)
()

Output:
7.00
3.00
```

Added by:	Blue Mary
Date:	2008-04-19
Time limit:	1s
Source limit:	512B
Languages:	All except: C99 strict
Resource:	Waterloo Local Contest

SPOJ Problem Set (challenge)

3099. Super Quine

Problem code: SELF

Write A program to print itself n(-10 <= n <= 10) times.

Negative count means the reversed program.

Input

A line contains an integer n.

Output

The required string described above.

Score

Score is the length of your code.

Example

```
Input:
-2

Source:
abcdefg

Output:
gfedcbagfedcba

Score:
7

Hint:
You can contact me if you have any question about the Special Judge or your code.
```

Added by:	Jin Bin
Date:	2008-10-03
Time limit:	1s
Source limit:	10240B
Languages:	C C99 strict C++ PAS gpc PAS fpc JAVA C#
Resource:	Wu Zhuojie

SPOJ Problem Set (challenge)

3880. Nop

Problem code: NOP

Mirko purchased a new microprocessor. Unfortunately, he soon learned that many of his programs that he wrote for his old processor didn't work on the new processor.

Deep inside the technical documentation for both processors, he found an explanation. In order to work faster, the new processor imposes certain constraints on the machine code of programs, constraints that never existed on the previous model.

The machine code of a processor consists of instructions that are executed sequentially. Each instruction uses a byte of memory. Also, instructions can have zero or more parameters, each of which uses an additional byte of memory. In machine code, parameters immediately follow an instruction.

When formatted as text, machine code instructions are uppercase letters, while parameters are lowercase letters. For example:

```
A b c b B c c C D e f g h
```

This program consists of four instructions; the first takes three parameters, the second two, the third none and the fourth takes four parameters. The program uses 13 bytes of memory.

The new processor model fetches memory in four-byte chunks so each instruction must start at a memory address that is divisible by four (the first byte in memory is address 0). To achieve that, we can insert NOP (no operation) instructions into the old program, instructions that do nothing and are not limited to memory locations divisible by four. The above program, adapted to run on the new processor, can look like this:

```
A b c b B c c NOP C NOP NOP NOP D e f g h
```

The instructions A, B, C and D are now at memory locations 0, 4, 8 and 12, which satisfies the processor's constraints.

Write a program that determines the smallest number of NOP instructions that need to be inserted for the given program to work on the new processor model.

Input

The input contains the machine code of the program written for the old processor model. The program will consist of at most 200 English letters.

The program will always start in an instruction i.e. the first letter in the machine code will be uppercase. If an instruction appears more than once in the machine code, it will always take the same number of parameters.

Output

Output the smallest number of NOP instructions needed to adapt the program for the new processor.

Example

```
Input
Abcd

Output
0


Input
EaEbFabG

Output
5


Input
AbcbBccCDefgh

Output
4
```

Score is your source code length. Have fun!

Added by:	Race with time
Date:	2009-02-17
Time limit:	1s
Source limit:	360B
Languages:	All
Resource:	COCI 2008/2009 - Croatian Regional

SPOJ Problem Set (challenge)

3947. Brainf F##k Writing

Problem code: BFWRITE

Task is about checking your Brainf**k Skill!! So you have to write "SPOJ is indeed awesome" just using brainf**k. Try to use as few letters as possible.

Input

Nothing

Output

Just: "SPOJ is indeed awesome"

Example

```
Input:


Output:
SPOJ is indeed awesome
```

Added by:	Michael Suchacz
Date:	2009-02-24
Time limit:	0.5s-1s
Source limit:	50000B
Languages:	BF

SPOJ Problem Set (challenge)

4246. Place the Numbers II

Problem code: PLCNUM2

Some days ago, Little Chucha bought a computer game. She is given a NxN board which she has to fill with the numbers 1 to N^2, no repetitions allowed. The computer calculates the sum of distances for each pair of consecutive numbers, that is, 1 -> 2, 2 -> 3, ..., N^2 -> 1. The goal is to make that sum as short as possible.

After many hours spent playing, Chucha has mastered the game. So she bought a new version and now the goal is to make the sum of distances as big as possible. Can you help her?

Input

Input consists of a single integer number 1<=N<=100, the size of the board.

Output

Output one possible placing of the numbers. You are to write N lines, N space separated integers each.

Example

```
Input:
3

Output:
1 2 3
4 5 6
7 8 9

Score:
Score for the example is:
Distance 1 -> 2 : 1
Distance 2 -> 3 : 1
Distance 3 -> 4 : 3
Distance 4 -> 5 : 1
Distance 5 -> 6 : 1
Distance 6 -> 7 : 3
Distance 7 -> 8 : 1
Distance 8 -> 9 : 1
Distance 9 -> 1 : 4
Sum of distances (SOD): 16, Min SOD: 10, Score: 1+16-10=7 points.
```

Added by:	yandry pérez clemente
Date:	2009-04-22
Time limit:	5s
Source limit:	50000B
Languages:	All

SPOJ Problem Set (challenge)

6173. Burned Pancakes Tower

Problem code: DBP

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 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 integer N < 55, 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, 1000.

Output

For each dataset, you should generate one line of output with the following values: The number of flips (K, where 0 <= K <= 3000) 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 and your task is to find the shortest one.

Score

For each test case is (3M-1)/(K+1)

Example

Input:
```
5
12 -2 -1 -10 +7 -4 -8 +12 -3 +5 +11 +6 -9
11 +2 +6 -10 +4 -1 -5 +7 -8 -3 +9 -11
11 +1 +8 +11 +10 +4 +3 -5 +2 -6 -7 -9
13 -2 -12 +8 +9 +10 -6 -5 +3 -7 -4 +11 -1 +13
12 +11 -3 +7 -5 -8 -12 -9 -1 +4 -2 -10 +6
```

Output:
```
25 7 12 3 1 11 4 10 2 1 9 3 1 8 6 7 2 1 6 4 5 1 1 4 1 3
25 11 1 11 3 1 10 1 1 9 7 1 8 2 1 6 1 5 3 4 2 1 3 1 2 1
25 3 11 8 1 10 3 9 7 1 8 7 1 7 6 1 6 4 1 5 3 4 1 1 3 1
24 2 1 12 2 1 11 4 10 3 7 2 1 6 4 5 2 1 4 1 3 1 1 2 1
29 6 1 12 7 11 6 10 1 1 9 5 1 8 6 7 5 6 2 5 2 1 4 1 1 3 1 1 2 1
```

Score:
```
6.4943590
```

Added by:	Ruslan Sennov
Date:	2010-02-18
Time limit:	5s
Source limit:	50000B
Languages:	All except: TECS
Resource:	Soccer Choreography, Flipping Burned Pancakes

SPOJ Problem Set (challenge)

6295. Area Difference

Problem code: SQDIFF

A gardener bought two sprinklers for his new garden. Each sprinkler moistens the soil around a circle with known radius. One will be working in the morning and the other in the evening. To plan how many plants of different species is better to plant, the gardener wants to know what area of the garden will be watered only in the morning, only in the evening or both in the morning and in the evening. After some searching on the Internet, he found a formula for calculating the area of intersection of circles, but for some reason the formula that would help to calculate the remaining two areas was not found.

Input

The first line contains T (1 <= T <= 1000) - the number of tests. The next T lines contain six integers x1, y1, r1, x2, y2 and r2 (0 <= xi, yi, ri <= 10000) - coordinates and radii of the first and second sprinkler, respectively.

Output

For each test case print the absolute value of the difference between the the area that is going to be watered only in the moring and the area that is going to be watered only in the evening rounded to two decimals after the point. Your score is the source length.

Example

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

Output:
```
21.99
```

Added by:	Spooky
Date:	2010-03-09
Time limit:	1s
Source limit:	2000B
Languages:	All except: TECS
Resource:	Advancement Spring 2010, http://sevolymp.uuuq.com/, author: Alexey Shchepin

SPOJ Problem Set (tutorial)

6338. Monster

Problem code: MONS

Your Task is simple. Write the code that will print the content of the following file.

https://www.spoj.pl/content/skydbms:monster

Input

No Input

Output

The content of the above file.

Score

Minimum is the size, better is the score.

Added by:	A@K@$h J0h@rI
Date:	2010-03-14
Time limit:	1s-5s
Source limit:	3000B
Languages:	C C99 strict C++ 4.0.0-8 C++ 4.3.2 JAVA
Resource:	Own Problem

6646. Fully Parenthesized Expression

Problem code: BRACKETS

Clyde has written a program that can evaluate arithmetic expressions. There is just one problem - the expressions must be fully parenthesized!

Help Clyde by making sure all of his expressions are fully parenthesized - he's willing to reward you.

Input

The first line of input contains a single integer `n` (1 <= `n` <= 1000) that indicates the amount of test cases.
On the next `n` lines will be a string `s` representing the expression. `S` will always contain a valid expression and will be strictly less than 256 characters in length.

The operators used in `s` are */% of high precedence and +- of low precedence. All operators are binary. All input strings will consist of only characters in the set "0123456789+-*/%()" (no whitespace).

Output

Your program should output `n` lines, each containing the fully parenthesized expression `s`.

A fully parenthesized expression is one where each operator is one where all operands are surrounded by either one parenthesis and one operator, or two parentheses. There should not be any extra parentheses.

The outputted expression does not need to be a solvable expression.

Fully Parenthesized	**Not Fully Parenthesized**
(42)	42
(1+(2*3))	1+(2*3)
(1+((61%(3+6))*7))	1+61%(3+6)*7

Example

Input
```
6
42
1+2*3
1%6*2
2/0
(1+((61%(3+6))*7))
(((((1)))))
```

Output

```
(42)
(1+(2*3))
((1%6)*2)
(2/0)
(1+((61%(3+6))*7))
(1)
```

Score

Your score is the length of your source code.

Added by:	Jargon
Date:	2010-05-13
Time limit:	1s
Source limit:	50000B
Languages:	All except: PERL 6
Resource:	Own problem

SPOJ Problem Set (challenge)

7105. Reverse the Input

Problem code: REVINPUT

A simple task is a rare thing in SPOJ these days, but if you are looking for one, this task is for you!, all you need to do is to reverse the input N times.

Score is the length of your solution.

Input

The first lines of the input is an integer N (1 <= N <= 100). Next follows the input consisting of all printable ASCII characters.The input is terminated by EOF. You may assume that length of each word is less then 1024 and maximum number of words can be never more than 1024.

Output

Reverse the input N times.There must be a space between two words of the output.

Example

```
Input:2fox jumps over the lazy dog.Output:.god .god yzal yzal eht eht revo revo spmuj spmuj xof xof
```

Added by: .:: Debanjan ::.
Date: 2010-08-13
Time limit: 1s
Source limit:1000B
Languages: All

SPOJ Problem Set (challenge)

7225. Word To Number

Problem code: WORDNUM

In this task you just need to convert a number from its word form to digit form. For example, you should output 1 for ’one’, 37000 for ’thirty seven thousand’ or 99 for ’ninety nine’. You can assume that output will be >= 0 and < 50000. Also all numbers will be valid i.e output will not have numbers like 023.

Input

First line of input has a single positive integer t = number of test cases. In the next t lines, given a number in word form.

Output

Output the number as specified.

Example

```
Input:3one hundred twenty threethirty four thousand one hundred eightelevenOutput:1233410811Score : Source code length
```

Added by:	XeRon!X
Date:	2010-08-24
Time limit:	1s
Source limit:	1000B
Languages:	All except: TECS
Resource:	-

SPOJ Problem Set (challenge)

7965. The Electronic Dice

Problem code: DIE_PIP

A **die** is a small throwable object with multiple resting attitudes, generally used as gambling devices, especially for craps or sic bo, or for use in non-gambling tabletop games.A traditional die is a cube (often with corners slightly rounded), marked on each of its six faces with a different number of circular patches or pits called **pips**. All of these pips have the same appearance within a set of dice, and are sized for ease of recognizing the pattern formed by the pips on a face.Die These pips are typically arranged in patterns denoting the numbers one through six. The sum of opposing faces traditionally adds up to seven.

In this problem we will be developing a module for electronic die,your task is to convert from the decimal die roll to the visible configuration of pips (dots) on the electronic die's face,as this die is electronic we will use 0 and 1 instead of the standard one.

The challenges thrown to you are that you can use any one of your favourite programming language(s) as long it's name starts with 'C' and to keep your solution less than 128 bytes.

Score is the length of your source.

Input

An integer N **(0<N<7),** one in each line. Input is terminated by EOF.

Output

The face of the die giving the appropiate value.

Example

Input:`123456`

Output:

```
0 0 00 1 00 0 00 0 10 0 01 0 01 0 00 1 00 0 11 0 10 0 01 0 11 0 10 1 01 0 11 0 11 0 11 0 1
```

Added by:	.:: Debanjan ::.
Date:	2010-11-29
Time limit:	1s
Source limit:	128B
Languages:	C C# C++ 4.0.0-8 C++ 4.3.2 C99 strict CLPS CLOJ CAML

SPOJ Problem Set (challenge)

8315. Number to Word

Problem code: NUMWORD

In this task you just need to convert a number from its digit form to word form. For example, you should output 'one' for 1,'thirty seven thousand' for 37000 . You can assume that input will be >= 0 and < 50000. Also all numbers will be valid i.e input will not have numbers like 023. **Input** input contains one line which is n 0<=n>=50,000 **Output** Output the number in words as specified. **Example** **Input:**

```
12334108
```

Output:

```
one hundred twenty threethirty four thousand one hundred eight
```

Score : Source code length

```
problem modified on 23 / 2 / 2011
many other languages added as users want!!! and problem statment changed which lead to no accepted solutions
```

```
and that is because increasing in test cases number
```

Added by:	kawmia institutes problem setters
Date:	2011-02-12
Time limit:	1s
Source limit:	1000B
Languages:	C C++ 4.0.0-8 C++ 4.3.2 HASK JAVA PAS gpc PAS fpc PERL PERL 6 PHP PYTH 2.5 PYTH 3.1.2 RUBY
Resource:	XeRon!X problem in opposite way

SPOJ Problem Set (challenge)

8399. Linear Congruences

Problem code: LINC

In this task you have to solve system of linear congruences.

Input

A list of system of Linear congruences,in the form **"x = P (mod Q)"** ,(without qoutes) where [math] 0 \le P \le 2^{31}[/math] and [math] 0 \le Q \le 2^{31}[/math].Every equations of a system is given in exactly one in each line and there is an additional new line between two systems.

The input is terminated by EOF.

Output

Output the solutions in the form of '**A + Bk**'. If the system is not solvable output "**no solutions**"

Example

```
Input:x = 0 (mod 5)x = 3 (mod 10)x = 12 (mod 21)x = 0 (mod 6)x = 12 (mod 35)x = 12 (mod 38)x = 12 (mod 31)x = 12 (mod 14)

Output:
no solutions
12 + 123690k
Score is the length of your source.
```

Added by: .:: Debanjan ::.
Date: 2011-02-23
Time limit: 1s
Source limit:1000B
Languages: All

SPOJ Problem Set (challenge)

8579. BF_MODULUS

Problem code: MODULUS2

See also http://www.spoj.pl/problems/PROBLEM1/

Problem

Given an integer n print all the possibilities of a%n where a can be any positive integer.

Input

The first line consists of an integer t, the number of test cases followed by t lines containing an integer n.

Output

For each test case print all the possibilities of a%n in descending order separated by a single space. After each test case print a new line character. If there are no possibilities print "NOT POSSIBLE".

Input specifications:

0<t<=100

0<=N<=100

Time limit: 1 second

Example

Sample Input

2

1

2

Sample Output

```
0

1 0
```

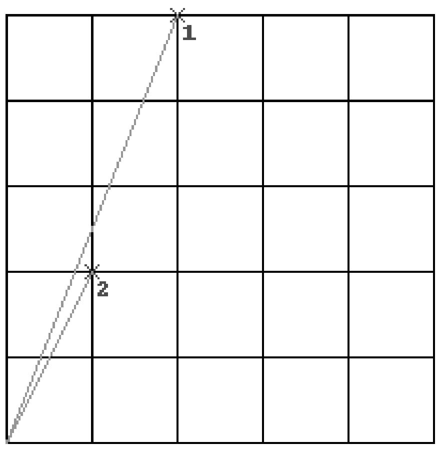
Added by:	cegprakash
Date:	2011-03-18
Time limit:	1s
Source limit:	50000B
Languages:	BF

SPOJ Problem Set (challenge)

8950. Grid points

Problem code: GRIDPNT

There’s a Cartesian lattice with 0<=x,y<=n. Given one point (x1, y1>0) in this lattice rotating clockwise around the originfind the next point (x2, y2). The given and searched points mustn’t haveanother point between the origin (0, 0) and this point itself.
x1, y1, x2, y2 are non-negative integers.



Score is source length.

Input

In the first line the number T (T<100) of test cases.
Then T lines with the space-separated n (1<=n<=50), x1 and y1.

Output

For each test case the space-separated x2 and y2.

Example

Input:
```
31 1 15 3 2100 97 98
```

Output:
```
1 05 398 99
```

Added by:	HWK
Date:	2011-05-24
Time limit:	20s
Source limit:	50000B
Languages:	All