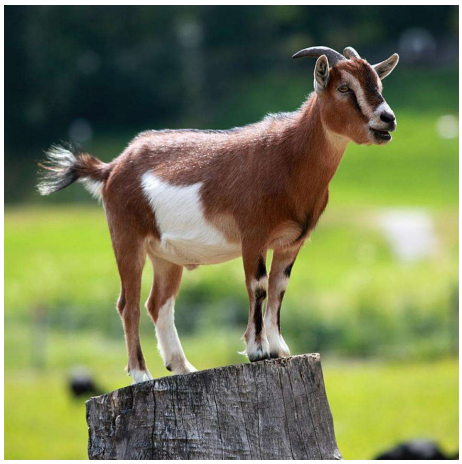
Problem A. Alphabet Animals

Source file name:	animals.c, animals.cpp, animals.java, animals.py
Input:	Standard
Output:	Standard

You are playing a game in which a group of players take turns saying animal names. The animal name you say when it is your turn must start with the same letter as the previously said animal ends with and it must not have been said previously in this round of the game. If there is no valid name or you cannot come up with one you are eliminated.

Given the last animal name said before your turn and a list of all names not yet used, can you make it through this turn? If so, can you make sure to eliminate the next player?



Solution to Sample Input 1 by Kuebi via Wikimedia Commons, cc by-s

Input

The first line of input contains a single word, the animal that the previous player just said. The next line contains a single integer n ($0 \leq n \leq 10^5$), the number of valid unused animal names. Each of the following n lines contains one valid unused animal name. All animal names (including the one the previous player said) are unique and consist of at least 1 and at most 20 lower case letters 'a'-'z'.

Output

If there is any animal name you can play that eliminates the next player, output the first such name from the input list, followed by an exclamation mark. Otherwise, if there is any animal name that you can play, output the first such name. Otherwise, output a question mark (in this case you will just have to make up a fake name in the hope that the others will trust you that this is a real animal).

Example

Input	Output
pig 2 goat toad	goat
dog 2 snake emu	?
hare 3 bee cat eagle	eagle!

Problem B. Building Boundaries

Source file name:	building.c, building.cpp, building.java, building.py
Input:	`Standard`
Output:	`Standard`

Maarja wants to buy a rectangular piece of land and then construct three buildings on that land. The boundaries of the buildings on the ground must have rectangular sizes $a_1 \times b_1$, $a_2 \times b_2$, and $a3 \times b3$. They can touch each other but they may not overlap. They can also be rotated as long as their sides are horizontal and vertical. What is the minimum area of land Maarja has to buy?

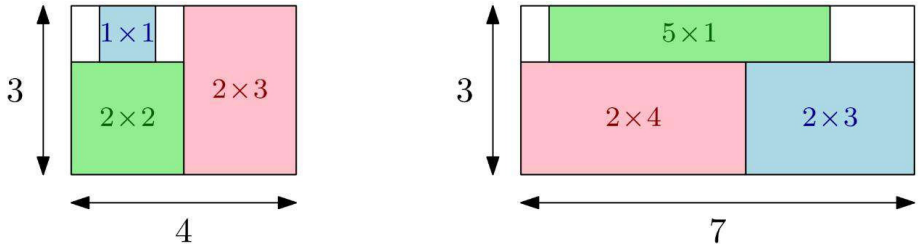


Figure B.1: Illustration of the two test scenarios in Sample Input 1 and their solutions. In the second scenario the 5×1 building has been rotated by 90 degrees.

Input

The input consists of multiple test scenarios. The first line of input contains a single integer t ($1 \leq t \leq 1000$), the number of scenarios. Then follow the t scenarios. Each scenario consists of a single line, containing six integers a_1, b_1, a_2, b_2, a_3 and b_3 ($1 \leq a_1, b_1, a_2, b_2, a_3, b_3 \leq 10^9$).

Output

For each test scenario, output the minimum area of land such that Maarja can construct the three buildings.

Example

Input	Output
2 2 3 2 2 1 1 2 4 5 1 2 3	12 21

Problem C. Cocoa Coalition

Source file name:	cocoa.c, cocoa.cpp, cocoa.java, cocoa.py
Input:	`Standard`
Output:	`Standard`

Alice and Bob decide to share a chocolate bar, which is an n by m rectangular grid of chocolate cells. They decide that Alice should get $a < n \cdot m$ pieces and that Bob should get $b = n \cdot m - a$ pieces. To split the chocolate bar, they repeatedly take a single piece of chocolate and break it either horizontally or vertically, creating two smaller pieces of chocolate. See Figure C.1 for an example.

What is the minimum number of splits that Alice and Bob need to perform in order to split the n-by-m chocolate bar into two piles consisting of a and b chocolate cells?

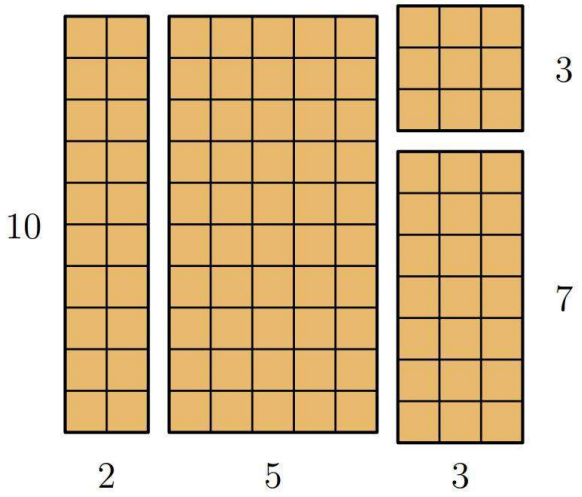


Figure C.1: Illustration of a solution to Sample Input 2, showing the original 10-by-10 chocolate bar split three times into pieces of size 10-by-2, 10-by-5 , 3-by-3 and 7-by-3 . Giving Alice the 10-by-5 and 7-by-3 pieces, she gets a total of $50 + 21 = 71$ chocolate cells.

Input

The input consists of a single line, containing the three integers n, m and a ($1 \leq n, m \leq 10^6,\ 1 \leq a < n \cdot m$).

Output

Output the minimum number of splits needed to achieve the desired division of chocolate.

Example

Input	Output
3 10 9	1
10 10 71	3

Problem D. Darts

Source file name:	darts.c, darts.cpp, darts.java, darts.py
Input:	`Standard`
Output:	`Standard`

A staple to many game rooms, darts is a fun game that doesn't require a lot of physical space. The standard dartboard (see Figure 1) consists of 6 concentric rings and 20 wedges, dividing the board into a number of regions, each with well-defined scores (note: for a higher picture resolution/clarity, fewer than 20 wedges are depicted in Figure1). The centermost ring is scored as a double bullseye, while the second ring gives the score of a single bullseye. Beyond the second ring, each wedge has a score between 1 and 20 with the spaces between certain rings having double and triple score modifiers. A new, simpler version of the game is being proposed to attract younger players to the game.

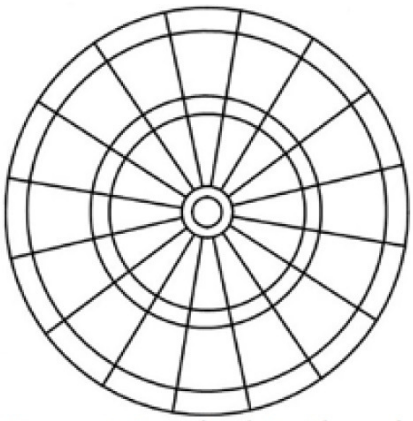


Figure 1, Standard Dartboard

The simpler version of the game is illustrated in Figure 2. More specifically, the board consists of exactly 3 (instead of 6) concentric rings and it may have fewer (instead of exactly 20) wedges. Also, in the simpler board, the wedges are of equal size. The scoring for this simple version is as follows:

Assume the board is centered at the origin ("0,0" in Cartesian plane). Let w refer to the number of wedges on the board (8 in Figure 2), and let b, d, s refer to (respectively) the radii of the smallest, second smallest, and the largest rings.

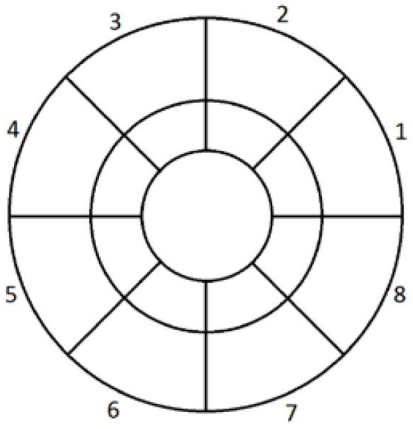


Figure 2, Example Simple Dartboard

The wedge immediately above the positive x axis (Wedge 1 in Figure 2) is scored at 1 point and the score for each wedge is increased by 1 in a counterclockwise fashion, thus resulting in the wedge below the positive x axis (Wedge 8 in Figure 2) having a score of w. The area within the circle centered at the origin with radius b represents the bullseye and is always scored at 50 points. The area between radii b and d denotes the double ring and any dart landing within the double ring is scored at twice the value of the surrounding wedge. The area between radii d and s denotes the single ring and any dart landing within the single ring is scored at the value of the surrounding wedge. Any dart landing outside the dartboard carries a score of zero (0).

Given the description (layout) of such a board centered at the origin and a number of dart throws, you are to calculate the total score.

Input

The first input line contains a positive integer, n, indicating the number of test cases to process. Each test case will contain multiple input lines. The first line of each test case will contain four integers separated by a space: w ($2 \leq w \leq 20$), representing the number of equal-sized wedges on the board, followed by b, d, s ($0 < b < d < s < 100$), representing the radii of the bullseye, double ring and single ring regions of the board. The second input line of each test case will contain an integer, t ($1 \leq t \leq 100$), indicating the number of dart throws. Each of the following t input lines contains two floating point numbers (three decimal places), x and y ($-100 \leq x,\ y \leq 100$), providing the Cartesian coordinates of a dart throw. Assume that no dart will land within 10-5 of any boundary (line or arc/curve).

Output

For each test case, output a single integer on a line by itself indicating the total score for all dart throws.

Example

Input	Output
3 4 7 13 10 2 4.000 4.000 6.000 -4.000 10 1 6 10 1 20.000 -0.500 8 3 7 50 5 -0.750 1.207 1.180 3.132 27.111 -44.630 -43.912 -22.104 2.000 -6.000	58 0 73

Problem E. Eeny Meeny

Source file name:	meeny.c, meeny.cpp, meeny.java, meeny.py
Input:	`Standard`
Output:	`Standard`

"Eeny meeny miny moe" is a well-known nursery rhyme in English, used (among other things) by kids to "randomly" select members of a team. It exists in many variations, one of which goes like this:

Eeny, meeny, miny, moe,
Catch a tiger by the toe.
If he hollers, let him go,
Eeny, meeny, miny, moe.

Picture by Gina Harbach Glenn Green on Flickr, cc by-nd

Similar verses exist in most languages, such as "Ulle dulle dof" in Finnish, "Akka bakka bonka rakka" in Norwegian, and "Ole dole doff" in Swedish.

Two teams are to be selected for a game and the rhyme is used to select one kid for a team at a time, alternating between the two teams, until all kids have been selected. The kids are standing in a circle. In each selection round we start counting the kids in clockwise order around the circle, skipping one kid for every word in the rhyme, until the last word. The kid matching the last word is chosen for the current team and then the next round starts. In all rounds but the first, the counting starts at the next remaining kid (in clockwise order) after the one that was selected in the previous round. See Figure E.1 for an example. Given such a rhyme, and a group of kids, can you tell which kids will be in which team?

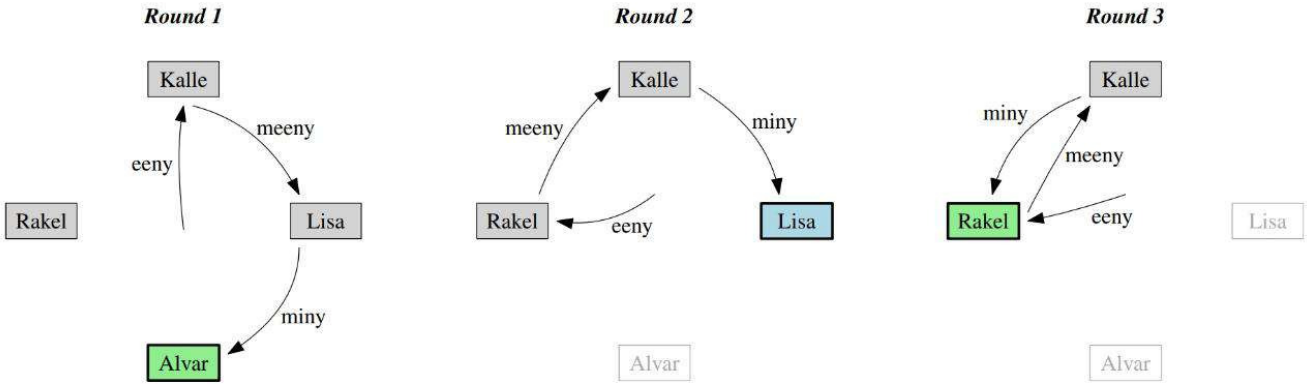


Figure E.1: Illustration of the first three rounds of Sample Input 1. In rounds 1 and 3, Alvar and Rakel get selected for the first team, and in round 2, Lisa is selected for the second team. In round 4 (not shown), only Kalle remains and is selected for the second team.

Input

The first line of input contains the rhyme, consisting of a list of words separated by spaces. The second line of input contains an integer n ($1 \leq n \leq 100$), the number of kids. Then follow the names of the kids, one per line. The kids are given in clockwise order and the first kid listed is the one at which counting starts in the first round.

All words and names consist only of upper and lower case letters '`A`'-'`Z`' and '`a`'-'`z`'. No input line is empty or longer than 100 characters (excluding the newline character at the end of the line).

Output

Output the two teams, starting with the one whose first member is chosen first. For each team, output the number of kids in the team, followed by the names of the kids in the team, in the same order as they were chosen for the team.

Example

Input	Output
eeny meeny miny 4 Kalle Lisa Alvar Rakel	2 Alvar Rakel 2 Lisa Kalle
Every Other 3 a b c	2 b c 1 a

Problem F. Fygon 2.0

Source file name:	fygon20.c, fygon20.cpp, fygon20.java, fygon20.py
Input:	`Standard`
Output:	`Standard`

The new version of beloved programming language Fygon has been released! The brand new Fygon 2.0 still has only two statements. The first statement is `lag`. It substitutes almost any other statement. Second statement is a `for` loop:

```
for <variable> in range(<from>, <to>):
    <body>
```

- The `for` loop makes `<variable>` iterate from `<from>` to `<to>`, **both inclusive**.
- If `<from>` is greater than `<to>`, `<body>` is not executed at all.
- `<variable>` is a lowercase letter from `a` to `z`, except for `n`, which is a variable that is defined prior to the given code snippet.
- `<from>` and `<to>` can be equal to any variable defined in outer loop. In addition to that, `<from>` can be `1` and `<to>` can be `n`.
- The `<body>` of the loop is indented by four spaces and contains at least one statement.

If you are familiar with Fygon 1.0, you can notice that, in the spirit of the best programming practices, Fygon 2.0 is not backwards compatible, since the `range` function now requires two parameters.

The performance of the new version is significantly improved, so you can write more nested `for` loops. That is why we are no longer interested in the exact number of operations, but in the *asymptotic complexity* of the program instead. For simplicity, all `for` loops are nested in a single chain and there is exactly one `lag` statement that is inside all `for` loops. All loop variables are different and are not equal to `n`.

Let's define $f(n)$ as the number of `lag` operations exectuted by a given Fygon program as the function of n. For non-negative integer k and positive rational number C we say that $C \cdot n^k$ is the *asymptotic complexity* of the program if

$$\lim_{n \to \infty} \frac{f(n)}{C \cdot n^k} = 1.$$

Given a Fygon 2.0 program, find its asymptotic complexity.

Input

The first line of the input contains single integer m — the number of lines in Fygon 2.0 program. Next m lines contain the program itself. The program has at least 1 and at most 20 `for` statements. Each `for` statement contains either single nested `for` statement or `lag` statement.

Output

Output numbers k and C. C should be output in the form of irreducible fraction p/q, where p and q are coprime.

Example

Input	Output

4
for i in range(1, n):
  for j in range(1, i):
  for k in range(j, n):
  lag
	3 1/3

Problem G. Game of Gnomes

Source file name:	gnomes.c, gnomes.cpp, gnomes.java, gnomes.py
Input:	`Standard`
Output:	`Standard`

The enemy and their massive army is approaching your fortress, and all you have to defend it is a legion of guardian gnomes. There is no hope of winning the battle, so your focus will instead be on causing as much damage as possible to the enemy.

Picture by Danielle Walquist Lynch via flickr, cc by

You have n gnomes at your disposal. Before the battle, they must be divided into at most m non-empty groups. The battle will then proceed in turns. Each turn, your gnomes will attack the enemy, causing one unit of damage for each living gnome. Then the enemy will attack by throwing a lightning bolt at one of the m groups. The lightning bolt kills k of the gnomes in that group, or all of them if the number of living gnomes in the group is less than k . The battle ends when all gnomes are dead. The enemy will always throw the lightning bolts in an optimal way such that the total damage caused by the gnomes is minimized.

Now you wonder, what is the maximum amount of damage you can cause to the enemy if you divide the gnomes into groups in an optimal way?

For example, if as in Sample Input 1 you have $n = 10$ gnomes that are to be divided into $m = 4$ groups, and the lightning bolt does at most $k = 3$ damage, then an optimal solution would be to create one large group of size 7 and three small groups of size 1 . In the first round, you cause 10 damage and the lightning bolt reduces the large group by 3 . In the next round, you cause 7 damage and the large group is reduced down to size 1 . In the remaining four rounds you do $4, 3, 2$, and 1 damage respectively and the lightning bolt removes one group each round. In total you do $10 + 7 + 4 + 3 + 2 + 1 = 27$ damage.

Input

The input consists of a single line containing the three integers n , m , and k ($1 \leq n \leq 10^9, 1 \leq m, k \leq 10^7$), with meanings as described above.

Output

Output the maximum amount of damage you can cause to the enemy.

Example

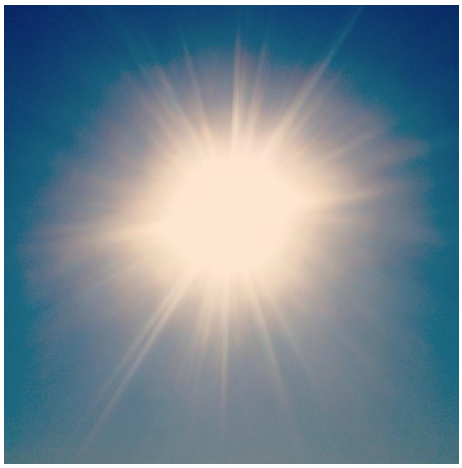
Input	Output
10 4 3	27
5 10 100	15

Problem H. Hot Hike

Source file name:	hike.c, hike.cpp, hike.java, hike.py
Input:	Standard
Output:	Standard

In order to pass time during your vacation, you decided to go on a hike to visit a scenic lake up in the mountains. Hiking to the lake will take you a full day, then you will stay there for a day to rest and enjoy the scenery, and then spend another day hiking home, for a total of three days. However, the accursed weather this summer is ridiculously warm and sunny, and since severe dehydration is not at the top of your priority list you want to schedule the three-day trip during some days where the two hiking days are the least warm. In particular you want to minimize the maximum temperature during the two hiking days.

Given the current forecast of daily maximum temperatures during your vacation, what are the best days for your trip?



Picture by dan lundmark on flickr, cc by

Input

The first line of input contains an integer n ($3 \leq n \leq 50$), the length of your vacation in days. Then follows a line containing n integers $t_1, t_2, ..., t_n$ ($-20 \leq t_i \leq 40$), where t_i is the temperature forecast for the i'th day of your vacation.

Output

Output two integers d and t, where d is the best day to start your trip, and t is the resulting maximum temperature during the two hiking days. If there are many choices of d that minimize the value of t, then output the smallest such d.

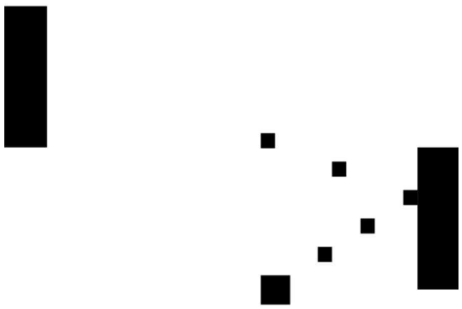
Example

Input	Output
5 23 27 31 28 30	2 28
4 30 20 20 30	1 30

Problem I. Incremental Induction

Source file name:	iInduction.c, iInduction.cpp, iInduction.java, iInduction.py
Input:	`Standard`
Output:	`Standard`

The Nordic Collegiate Pong Championship (NCPC) is an insanely competive tournament where every contestant plays exactly one game of Pong against every other contestant. The last game of the tourna- ment just finished, so only one item now remains on the programme: the traditional diploma ceremony, where all this year's participants get inducted into the NCPC Hall of Fame.



Pong via Wikimedia Commons, public domain

According to the ancient customs, contestants who have not been inducted into the Hall of Fame yet (the pathetic nobodies) must stay on the left side of the stage, whereas contestants who have been inducted (the awesome legends) must be on the right side of the stage. Then, when a contestant is receiving their diploma, they will symbolically walk from the left to the right side of the stage and thus become an awesome legend. Only one contestant is inducted into the Hall of Fame at a time, and every contestant starts on the left side initially.

The NCPC Head of Jury believes it reflects badly on her if too many of the awesome legends on the right have lost matches against pathetic nobodies on the left, but she quickly realizes that it might be impossible to avoid this at every point in time during the diploma ceremony. However, she certainly wants to keep such atrocities at a minimum. Specifically, she wants to find the smallest number k for which there exists an order of handing out diplomas to the contestants, such that at no point there were more than k games played where an awesome legend lost against a pathetic nobody.

Input

The first line of input contains a single integer n ($1 \leq n \leq 5000$), the number of contestants. Then follows $n-1$ lines, the i^{th} of which contains a binary string of length i. The j^{th} character on the i^{th} line is 1 if contestant $i+1$ defeated contestant j , and 0 if contestant j defeated contestant $i+1$.

Output

Output a single integer k, the smallest number according to the requirements above.

Example

Input	Output
4 1 01 100	1
5 0 00 100 1100	3

Problem J. Joker

Source file name:	joker.c, joker.cpp, joker.java, joker.py
Input:	`Standard`
Output:	`Standard`

Joker prepares a new card trick with a strong mathematical background. You are asked to help Joker with calculations.

There is a row of n cards with non-zero numbers a_i written on them. Let's call the sum of all positive numbers P and the sum of all negative numbers N. Every card i has a weight $w_i = \frac{a_i}{P}$ if $a_i > 0$ and $\frac{a_i}{|N|}$ otherwise.

Let's denote $s_i = \sum_{j=1}^{j \le i} w_j$. Joker needs to know positive i with the largest s_i. If there is more than one such i, he is interested in the smallest one.

But static tricks are boring, so Joker wants to change numbers on some cards, and after each change he needs to known where is the largest s_i is.

Input

The first line of the input contains two integers n and m — the number of cards and the number of changes ($1 \le n, m \le 50\,000$).

The second line consists of n integers a_i — numbers written on cards at the beginning ($-10^9 \le a_i \le 10^9$; $a_i \neq 0$).

The following m lines contain two integers each: p_i and v_i, that means value of card at position p_i is changed to v_i ($1 \le p_i \le n$; $-10^9 \le v_i \le 10^9$; $v_i \neq 0$).

It is guaranteed that at each moment there is at least one card with positive number and at least one card with negative number. The sum of all positive cards will never exceed 10^9 and the sum of all negative cards will never exceed -10^9.

Output

You should output $m+1$ integers. The first integer is the position of the largest s_i for the initial numbers. Next m numbers are positions of the largest s_i after each change.

Example

Input	Output
4 7 1 -5 3 -5 4 -1 2 -1 3 10 4 10 1 -1 2 1 3 -1	3 1 3 3 1 4 4 4

Problem K. Keyboard

Source file name:	keyboard.c, keyboard.cpp, keyboard.java, keyboard.py
Input:	`Standard`
Output:	`Standard`

Consider a simplified keyboard consisting of the 26 lowercase letters as illustrated below:

a	b	c	d	e	f	g	h	i
j	k	l	m	n	o	p	q	r
s	t	u	v	w	x	y	z	

We define the neighbors of a key (letter) as all the letters adjacent to it. For example, the neighbors of 'a' are b, k, j, neighbors of 'b' are a, c, l, k, j, neighbors of 'n' are d, e, f, o, x, w, v, m, and neighbors of 'z' are p, q, r, y.

Given two words consisting of lowercase letters only, you are to determine which of the following three cases applies to them:

1. identical: this is when the two words are of the same length and they match letter-by-letter. For example, "cool" and "cool" are identical, "cool" and "col" are not, and "cool" and "colo" are not.
2. similar: this is when the two words are of the same length, they are not identical words, and each corresponding two letters either match or are neighbors. For example, "aaaaa" and "abkja" are similar, "moon" and "done" are similar, "knq" and "bxz" are similar, but "ab" and "cb" are not (because of 'a' in the first word and the corresponding 'c' in the second word).
3. different: this is when neither of the above two cases applies to the two words, i.e., they are not identical and they are not similar. For example, "ab" and "abc" are different, "ab" and "az" are different, and "az" and "za" are different.

Input

The first input line contains a positive integer, n, indicating the number of test cases to process. Each of the following n input lines represents a test case, consisting of two words separated by one space. Each word consists of lowercase letters only and will be between 1 and 20 letters, inclusive.

Output

For each test case, output one line. That line should contain the digit (number) 1, 2, or 3, to indicate which of the above three cases applies to the two input words.

Example

Input	Output
7 a k a a a z cool cool aaaaa abkja ab abc az za	2 1 3 1 2 3 3

Problem L. Lucky Draw

Source file name:	lucky.c, lucky.cpp, lucky.java, lucky.py
Input:	`Standard`
Output:	`Standard`

You and your friends at the Betting against All Probability Club are visiting a casino where the following game is played.

Each of the n players starts with k lives and puts in a fixed amount of money. In each round of the game, each player flips a biased coin and loses a life if she gets tails. The game ends when only one player remains, in which case this person wins, or ends in a draw if all remaining players lose their last life in the same round. If there is a winner, she wins n times her original bet. In case of a draw, no one wins anything.

Being a BAPC member you quickly realize the casino has an edge here: whenever the game ends in a draw all of the contestants lose the money they bet. You are now wondering what exactly is the probability that this game ends in a draw, so you can figure out how much the casino profits on average.

Input

One line containing two integers, $2 \leq n \leq 50$, the number of players, $1 \leq k \leq 50$, the number of lives each player has, and a floating point number $0.1 \leq p \leq 0.9$, the probability the coin lands heads.

Output

Output a single floating point number: the probability of the game ending in a draw. Your answer should have an absolute error of at most 10^{-6}.

Example

Input	Output
2 2 0.5	0.185185185
2 2 0.8	0.056241426
5 3 0.85	0.045463964

Problem M. Multimodal Transport

Source file name:	transport.c, transport.cpp, transport.java, transport.py
Input:	`Standard`
Output:	`Standard`

New methods of shipping goods have transformed the transportation industry and helped usher in an era of global commerce. Nowadays, someone can click a few buttons and have an item leave a factory on the other side of the world and show up at their doorstep the next day. To help accomplish this, there are a number of modes of transport within the shipping industry. The four most common are: air, boat, rail and truck.

Transportation companies tend to keep the mode of transport consistent throughout a packages journey (route/path). However, when customers are not very time sensitive, often times the cheapest way to move a package from one location to another is to use more than one mode of transport. One has to be careful, though, as additional costs are incurred when switching between transport modes within a city on the package path. (Switching transport mode refers to a package leaving a city in a different mode than it arrived at the city, e.g., the package arrived by air and leaves by truck.)

A new startup, KnightShip, has asked for your help in figuring out the cheapest way to ship packages when switching between transportation modes is acceptable.

Given the costs for various modes of transport between cities, and the cost of switching mode within a city, calculate the lowest cost to transport an item from an origin to a destination.

Input

The first input line contains a positive integer, n, indicating the number of test cases to process. Each test case will contain multiple input lines. The first line of each test case will contain an integer, c ($2 \leq c \leq 400$), representing the number of cities within the transportation network. Each of the following c input lines contains two values: a string (1 to 20 uppercase letters, inclusive), representing the name of a city and an integer (between 1 and 1000, inclusive), representing the cost of changing the transport mode within that city.

The city information for a test case is followed by the route information for the test case. There will be an input line containing one integer, r ($1 \leq r \leq 40000$), representing the number of route segments in the network. This will be followed by a listing of all route segments, one per line. Each route segment will contain four values: p, q, representing two cities from the previous list, m, representing the transport mode (one of four values AIR, BOAT, RAIL, TRUCK) for that route segment, and an integer v ($1 \leq v \leq 1000$), representing the cost to ship a package between the two cities (either direction). Note that there may be no route segments for a particular transport mode. There will be no duplicate city pair within a given mode of transport, but different transport modes (even all four modes) can exist between the same two cities.

The last input line for a test case contains two distinct cities o and d which represent the origin and destination cities for which we want the minimum cost to ship a package. Cities o and d are guaranteed to have a path (of finite cost) that exists between them. Any mode of transport can be used to leave city o and any mode can be used to reach city d (they don't necessarily need to match). The transport mode can change in the intermediate stages as well.

Output

For each test case, output a single integer on a line by itself indicating the minimum cost to move a package from city o to city d.

Example

Input	Output
2 4 ORLANDO 10 TAMPA 15 MIAMI 5 JACKSONVILLE 10 7 TAMPA JACKSONVILLE AIR 100 MIAMI TAMPA SEA 70 JACKSONVILLE MIAMI RAIL 45 ORLANDO JACKSONVILLE TRUCK 85 TAMPA ORLANDO RAIL 10 MIAMI JACKSONVILLE SEA 15 ORLANDO MIAMI TRUCK 15 JACKSONVILLE TAMPA 2 ORLANDO 15 TAMPA 10 3 ORLANDO TAMPA AIR 7 TAMPA ORLANDO TRUCK 3 ORLANDO TAMPA RAIL 19 ORLANDO TAMPA	55 3