Benelux Algorithm Programming Contest

October 27, 2007

A Average distance

Given a tree, calculate the average distance between two vertices in the tree. For example, the average distance between two vertices in the following tree is $(d_{01} + d_{02} + d_{03} + d_{04} + d_{12} + d_{13} + d_{14} + d_{23} + d_{24} + d_{34})/10 = (6+3+7+9+9+13+15+10+12+2)/10 = 8.6$.

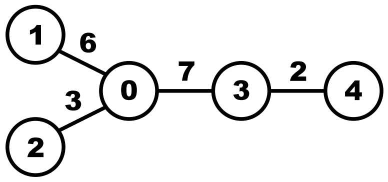


Figure 1: The first sample case

Input

On the first line an integer t ($1 \leq t \leq 100$): the number of test cases. Then for each test case:

- One line with an integer n ($2 \leq n \leq 10\,000$): the number of nodes in the tree. The nodes are numbered from 0 to $n-1$.
- $n-1$ lines, each with three integers a ($0 \leq a < n$), b ($0 \leq b < n$) and d ($1 \leq d \leq 1\,000$). There is an edge between the nodes with numbers a and b of length d. The resulting graph will be a tree.

Output

For each testcase:

- One line with the average distance between two vertices. This value should have either an absolute or a relative error of at most 10^{-6}.

Sample in- and output

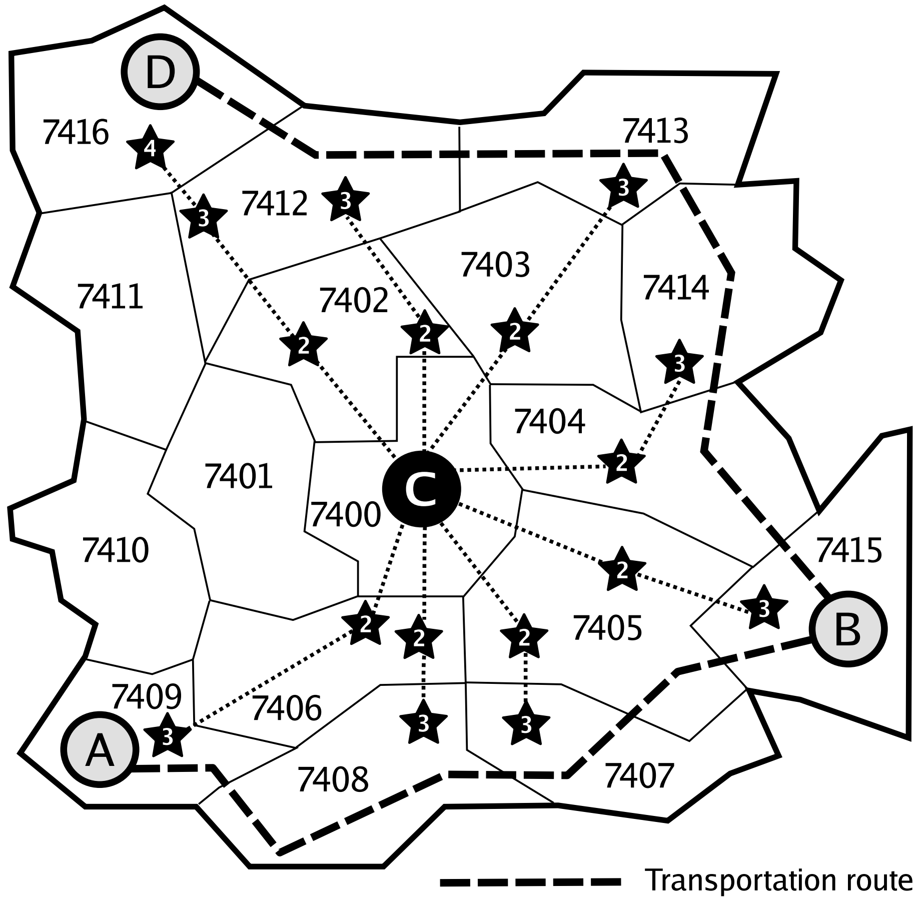
Input	**Output**
1	8.6
5	
0 1 6	
0 2 3	
0 3 7	
3 4 2	

B Bus Pass

You travel a lot by bus and the costs of all the seperate tickets are starting to add up. Therefore you want to see if it might be advantageous for you to buy a bus pass.

The way the bus system works in your country (and also in the Netherlands) is as follows: when you buy a bus pass, you have to indicate a center zone and a star value. You are allowed to travel freely in any zone which has a distance to your center zone which is less than your star value. For example, if you have a star value of one, you can only travel in your center zone. If you have a star value of two, you can also travel in all adjacent zones, et cetera.

You have a list of all bus trips you frequently make, and would like to determine the minimum star value you need to make all these trips using your buss pass. But this is not always an easy task. For example look at the following figure:



Here you want to be able to travel from A to B and from B to D. The best center zone is 7400, for which you only need a star value of 4. Note that you do not even visit this zone on your trips!

Input

On the first line an integer t ($1 \leq t \leq 100$): the number of test cases. Then for each test case:

- One line with two integers nz ($2 \leq nz \leq 9\,999$) and nr ($1 \leq nr \leq 10$): the number of zones and the number of bus trips, respectively.
- nz lines starting with two integers id_i ($1 \leq id_i \leq 9\,999$) and mz_i ($1 \leq mz_i \leq 10$), a number identifying the i-th zone and the number of zones adjacent to it, followed by mz_i integers: the numbers of the adjacent zones.
- nr lines starting with one integer mr_i ($1 \leq mr_i \leq 20$), indicating the number of zones the ith bus trip visits, followed by mr_i integers: the numbers of the zones through which the bus passes in the order in which they are visited.

All zones are connected, either directly or via other zones.

Output

For each test case:

- One line with two integers, the minimum star value and the id of a center zone which achieves this minimum star value. If there are multiple possibilities, choose the zone with the lowest number.

Sample in- and output

Input	Output
1	4 7400
17 2	
7400 6 7401 7402 7403 7404 7405 7406	
7401 6 7412 7402 7400 7406 7410 7411	
7402 5 7412 7403 7400 7401 7411	
7403 6 7413 7414 7404 7400 7402 7412	
7404 5 7403 7414 7415 7405 7400	
7405 6 7404 7415 7407 7408 7406 7400	
7406 7 7400 7405 7407 7408 7409 7410 7401	
7407 4 7408 7406 7405 7415	
7408 4 7409 7406 7405 7407	
7409 3 7410 7406 7408	
7410 4 7411 7401 7406 7409	
7411 5 7416 7412 7402 7401 7410	
7412 6 7416 7411 7401 7402 7403 7413	
7413 3 7412 7403 7414	
7414 3 7413 7403 7404	
7415 3 7404 7405 7407	
7416 2 7411 7412	
5 7409 7408 7407 7405 7415	
6 7415 7404 7414 7413 7412 7416	

C Cutting Banknotes

Philip is often faced with a big problem: after going out for dinner or having a few beers, he owes money to his friends or the other way around. These are often small amounts, but because Philip hates coins, his wallet contains only banknotes. Therefore he usually can't pay the amount exactly. Since he hates coins, he also doesn't allow his friends to return them as change. He does allow for banknotes as change though.

To accomodate for this problem, he and his friends came up with the following idea: let's pay with pieces of banknotes. To make the cutting easy, they only cut banknotes in two equally-sized pieces, cut those pieces in two pieces, and so on. This yields a much larger range of amounts that can be paid. Philip wonders which ones exactly.

Input

On the first line an integer t ($1 \leq t \leq 100$): the number of test cases. Then for each test case:

- One line with a number x ($0.01 \leq x \leq 10\,000.00$): the amount Philip has to pay. This is formatted with two decimal digits and a period as decimal separator.
- One line with a positive integer n ($1 \leq n \leq 1\,000$): the number of different banknotes.
- n lines, each with an integer b_i ($1 \leq b_i \leq 10\,000$): the values of the banknotes.

Output

For each test case:

- One line with "`yes`" if the amount can be paid exactly and "`no`" otherwise.

Sample in- and output

Input

```
4
10.75
3
2
10
20
0.33
1
1
10000.00
1
2500
1.00
2
3
5
```

Output

```
yes
no
yes
yes
```

D Dice Password Security

NCIM Group sponsored problem.

The NCIM Group does a lot of work on IT solutions in defense and security. Good security usually starts with picking a strong password. Generating a password at random is generally a good practice. For example, a password like "2R4eZ9Rqup" is a bit harder to guess than "god", "love", "sex" or "secret".

The problem with passwords consisting of random letters and digits is that they are hard to remember. Instead of using letters and digits it is also possible to generate passwords by putting random words together. Words are easier to remember than letters and digits. Using a dictionary of 7776 (6^5) words, a 5-random-word password is about as strong as a 11-random-character password.

$$7776^5 = 28430288029929701376 \approx 3 \cdot 10^{19}$$
$$62^{11} = 52036560683837093888 \approx 5 \cdot 10^{19}$$

Some applications hide the password you are typing on the screen by printing dots or asterisks. This allows someone watching your screen to count the number of characters in your password. The NCIM Group wants you to find out whether or not this compromises the strength of your password.

You must write a program that calculates the number of possible passwords that can be generated given:

- the dictionary of words,
- the amount of words used to generate the password and
- the length of the password.

Input

On the first line an integer t ($1 \leq t \leq 100$): the number of test cases. Then for each test case:

- One line with three positive integers m ($1 \leq m \leq 7776$), n ($1 \leq n \leq 5$) and q ($1 \leq q \leq 20$): the number of words in the dictionary, the number of words to generate the password, and the number of queries, respectively.
- The dictionary: m lines each containing one word w_i. Each word consists only of lowercase letters. The length of each word will be between 3 and 10 inclusive. No word in the dictionary will be a substring of another word in the dictionary.
- q lines each containing a positive integer l_j ($1 \leq l_j \leq 50$), the length observed.

Output

For each test case:

- q lines with: the number of possible passwords with length l_j. This number will be smaller than 2^{63}.

Sample in- and output

Input	**Output**
`1`	`6`
`4 2 2`	`9`
`aap`	
`noot`	
`mies`	
`piet`	
`7`	
`8`	

E Lingo

You are going to participate in the television show 'lingo'. You are very confident that you will make the finals of the show. This is not only because you are well prepared, but also because you managed to find a way to use your pda unnoticed during the contest, and you have already written a program to help you with the bonuswords.

In the finals of the show, you first solve as many lingo words as possible within the allowed time. This determines how many balls you may take afterwards. The more balls you may take, the higher your probability is of winning the finals. But it is not easy to see what this probability is. Write a program to help you with this.

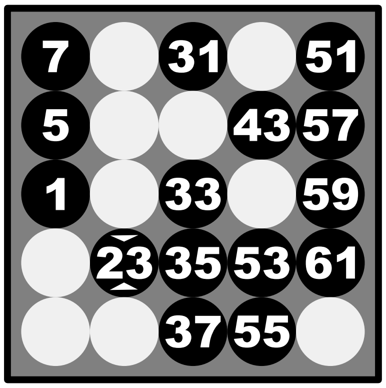


Figure 2: Grid of first sample case

For those who don't know the game of lingo, here follows a description of the last part of the finals, where you take the balls. You are given a square grid. Some squares in this grid are covered and the other squares contain numbers. A hopper in front of you contains numbered balls; there is exactly one ball for each numbered grid square. You take a ball at random (without replacement) from this hopper for each lingo word you solved in the first part of the finals. When you take a ball, the corresponding square in the grid becomes covered. You win the finals if an entire row, column or diagonal consist of only covered squares.

Input

On the first line an integer t ($1 \le t \le 100$): the number of test cases. Then for each test case:

- One line with the integers n ($1 \le n \le 8$) and k ($0 \le k$), where n is the size of the lingo grid and k is the number of words you solved in the first part of the finals.

- n lines, with on each line exactly n characters. Each character will be either '*' or '.', representing a covered square and a numbered square respectively.

There will be at least k numbered squares on the board, and there is no row, column or diagonal covered yet.

Output

For each test case:

- One line with the percentage of getting lingo with either an absolute or a relative error of at most 10^{-6}.

Sample in- and output

Input	Output
 1 5 7 .*.*. .**.. .*.*. *.... **..*	 82.703962704

F Splitting the Loot

After a lucrative enterprise (the details of which are best left untold) a large gold bar has come into your possession. However, since you promised your accomplices a share of the loot, you will need to split it up into several pieces.

Dividing a gold bar is not an easy task. Fortunately, you've found a goldsmith willing to do it without asking questions, under the conditions that he can keep a fixed percentage of the bar being divided as payment for his labour, and he will only divide it into two parts (although they don't have to be equal halves; you can pick the ratio).

For example, suppose you have a 100 gram gold bar, you have promised your two accomplices a share of 15 and 21 gram respectively and the goldsmith asks a 10% fee for each split. You can then first split the bar at a ratio of 5:4, yielding a 50 gram part (which you keep) and a 40 gram piece, which you split at 5:7 to yield the 15 and 21 gram pieces for your accomplices. Note that at each cut, you lose 10% of the gold to the goldsmith.

Since you want to maximize your own share of the loot without being unfair to your accomplices, you must be careful in the way you divide up the gold. In the example, if you had started by cutting off a 15 gram piece first, and then a 21 gram piece off the remaining bar, you would have ended up with only a 46.5 gram piece for yourself.

Your task is to determine how much gold you can keep, if you make the right cuts!

Input

On the first line an integer t ($1 \leq t \leq 100$): the number of test cases. Then for each test case:

- A line with three integers: the weight of the gold bar w ($1 \leq w \leq 1\,000\,000$), the cutting fee as a percentage p ($0 \leq p < 100$), and the number of accomplices n ($1 \leq n \leq 50$).
- n lines, each with the integer share s ($1 \leq s \leq w$) you promised an accomplice.

Output

For each test case:

- One line with the maximum amount of gold you can keep to yourself, or -1 if it is impossible to divide the gold satisfying the requirements.

Note that the answer is not always an integer. Your answer should have either an absolute or a relative error of at most 10^{-6}.

Notes

Your accomplices don't mind if you give them more gold than they bargained for. See the second sample case for a situation in which this is inevitable.

Sample in- and output

Input	Output
`3`	`50`
`100 10 2`	`0`
`15`	`-1`
`21`	
`45 15 3`	
`11`	
`11`	
`11`	
`50 0 3`	
`10`	
`20`	
`25`	

G Pachinko

Pachinko is a Japanese game played for amusement and prizes, and is similar to pinball. The game is very simple: you shoot small metal balls into the machine and they fall down, bouncing off the obstacles until they fall into a gate. The gate into which your ball falls determines the winnings.

Since you can more or less determine the column into which the ball is dropped (by setting its initial speed and direction), you can influence your win chances. You are to calculate your expected winnings.

We model the pachinko machine as follows: you can drop the ball in any column and it falls down until it either reaches the bottom of the machine (which results in no wins), a gate (indicated by a number 1 to 9, which results in that win) or an obstacle (indicated by a asterisk). If a ball hits an obstacle it proceeds falling in the column to the left or to the right of the obstacle, with 50/50 probability. No two obstacles or gates are adjacent, not even diagonally, and none ara located in the leftmost or rightmost column.

Input

On the first line an integer t ($1 \leq t \leq 100$): the number of test cases. Then for each test case:

- One line with two integers h and w ($1 \leq h, w \leq 100$): the width and height of the pachinko machine.
- h lines with w characters describing the pachinko machine. A '`.`' denotes an empty space, '`*`' an obstacle and '`1`'...'`9`' the winning gates.

Output

For each testcase:

- One line with the maximal expected winnings with either an absolute or a relative error of at most 10^{-6}.

Sample in- and output

Input

```
3
7 5
.....
.1...
...2.
.*...
.....
.....
..5..
8 8
...1....
........
..*...*.
....*...
.1......
...*.*..
........
..9.7.7.
10 10
.*.*.*.*..
..........
..*.*.*.*.
..........
.*.*.*.*..
..........
..*.*.*.*.
..........
.*.*.*.*..
....9.....
```

Output

```
5.000000
7.500000
3.375000
```

H Hiking

You have gone on a hiking trip, but now you are getting tired and would like to return home. As a precaution, you always take your mobile phone with you, so you can call for help in case of an emergency. However, it only works as long as you stay in range of (one of) the phone towers in the area. Fortunately, you know precisely where you are and where you are going, and you brought a map of the area showing the locations of the mobile phone towers. You want to take the shortest route home, while staying in range of at least one of those towers.

Input

On the first line an integer t ($1 \leq t \leq 100$): the number of test cases. Then for each test case:

- One line with two integers d ($1 \leq d \leq 1\,000$) and t ($1 \leq t \leq 100$): the maximum distance to the nearest tower and the number of towers on the map, respectively.
- One line containing the starting location.
- One line containing the goal location.
- t lines each containing the location of a tower.

All locations consist of two space-separated integer coordinates x and y ($1 \leq x, y \leq 1\,000$). All locations in the input for a single test case will be distinct. The starting locating will not be more than d units away from the nearest tower.

Output

For each test case:

- One line containing the distance to the goal location, or -1 if the goal location is unreachable.

Your answer should have either an absolute or a relative error of at most 10^{-6}.

Notes

- It's OK to be exactly d units away from the nearest tower.
- The size of the towers is negligible; if you move straight through a tower location, the tower won't block your path.

Sample in- and output

Input	**Output**
`1`	`7.23224071072994`
`2 3`	
`1 1`	
`8 2`	
`2 2`	
`4 4`	
`7 3`	

I Ranking

Hosting a programming contest is fun, but it's also a lot of work. For instance, at the end of the day the jury will have to create a ranking of the teams based on their results during the contest. This can be tedious to do by hand, so we would like you to write a program for this task.

For the BAPC, the rules for the ranking are as follows:

- Teams are ranked according to the most problems solved; teams tied are ordered by increasing total time used.
- The time used for a problem is the number of minutes between the start of the contest and the first accepted run, plus a 20 minute penalty for each rejected run until the first accepted run.
- The total time used is the sum of the times used for each problem solved (as described above). Note that penalty time for problems for which no run was accepted does not count toward the total time used.
- If ties remain at the end of the contest, the point of comparison between tied teams will be the last point in time where their scores differed (e.g. if two teams are tied at the end of the contest, the team which solved their last problem earlier than the other team solved their last problem wins).

During the contest, teams submit solutions for problems, which are processed by the jury as *runs*. Each run has four properties:

- the submission time in minutes since the beginning of the contest (an integer between 1 and 300, inclusive);
- the name of the team that submitted the solution (a non-empty string of at most 20 lower-case letters);
- the identifier of the corresponding problem (an upper-case letter A through J, inclusive);
- the result as determined by the judging software (either *accepted* or *rejected*).

At the end of the contest, we have a list of runs available (ordered by non-decreasing submission times) and we want you to determine the final ranking of the teams.

Teams that are tied will share a position; those teams should be ordered alphabetically in the results.

Input

On the first line an integer t ($1 \leq t \leq 100$): the number of test cases. Then for each test case:

- one line with the number of teams t ($1 \leq t \leq 50$) and the number of runs r ($0 \leq r \leq 5\,000$), seperated by a single space;
- then t lines, each with the name of a team;
- then r lines, each with the description of a run: time, team, problem and result, formatted as described above, and seperated by a single space.

Output

For each test case:

- Print the ordered results: t lines, each with a rank (starting from 1), the team name, the number of correctly solved problems and the total penalty time, separated by single spaces.

Sample in- and output

Input

```
1
8 28
twente
utrecht
groningen
amsterdam
eindhoven
leiden
delft
nijmegen
5 utrecht B rejected
8 eindhoven F accepted
10 utrecht F accepted
17 utrecht B rejected
18 leiden C rejected
23 twente F rejected
25 utrecht B accepted
26 amsterdam D rejected
27 amsterdam D accepted
27 leiden C accepted
27 groningen F accepted
28 twente F rejected
30 nijmegen C rejected
30 nijmegen C accepted
30 delft B accepted
30 delft B rejected
33 twente F accepted
47 groningen D rejected
51 leiden D accepted
51 amsterdam C accepted
51 groningen D accepted
60 utrecht D accepted
65 utrecht J accepted
67 twente F rejected
70 twente F accepted
90 eindhoven D accepted
100 utrecht A rejected
101 utrecht C rejected
```

Output

```
1 utrecht 4 200
2 groningen 2 98
3 amsterdam 2 98
3 leiden 2 98
5 eindhoven 2 98
6 delft 1 30
7 nijmegen 1 50
8 twente 1 73
```

J Stock

Optiver sponsored problem.

After years of hard work Optiver has developed a mathematical model that allows them to predict wether or not a company will be succesful. This obviously gives them a great advantage on the stock market.

In the past, Optiver made a deal with a big company, which forces them to buy shares of the company according to a fixed schedule. Unfortunately, Optiver's model has determined that the company will go bankrupt after exactly n days, after which their shares will become worthless.

Still, Optiver holds a large number of sell options that allows them to sell some of the shares before the company goes bankrupt. However, there is a limit on the number of shares Optiver can sell every day, and price Optiver receives for a share may vary from day to day. Therefore, it is not immediately clear when Optiver should sell their shares to maximize their profit, so they asked you to write a program to calculcate this.

Input

On the first line an integer t ($1 \leq t \leq 100$): the number of test cases. Then for each test case:

- One line with an integer n ($1 \leq n \leq 100\,000$): the number of days before the company goes bankrupt.
- n lines with three integers x_i ($0 \leq x_i \leq 100$), p_i ($0 \leq p_i \leq 100$) and m_i ($0 \leq m_i \leq 10\,000\,000$): the number of shares Optiver receives on day i, the (selling) price per share on day i, and the maximum number of shares Optiver can sell on day i, respectively.

Output

For each test case:

- One line with the maximum profit Optiver can achieve.

Sample in- and output

Input	**Output**
1	76
6	
4 4 2	
2 9 3	
2 6 3	
2 5 9	
2 2 2	
2 3 3	