Problem A: Welcome Party

For many summers, the Agile Crystal Mining company ran an internship program for students. They greatly valued interns' ability to self-organize into teams. So as a get-to-know-you activity during orientation, they asked the interns to form teams such that all members of a given team either have first names beginning with the same letter, or last names beginning with the same letter. To make it interesting, they asked the interns to do this while forming as few teams as possible.

As an example, one year there were six interns: Stephen Cook, Vinton Cerf, Edmund Clarke, Judea Pearl, Shafi Goldwasser, and Silvio Micali. They were able to self-organize into three teams:

- Stephen Cook, Vinton Cerf, and Edmund Clarke (whose last names all begin with C)
- Shafi Goldwasser and Silvio Micali (whose first names begin with S)
- Judea Pearl (not an interesting group, but everyone's first name in this group starts with J)

As a historical note, the company was eventually shut down due to a rather strange (and illegal) hiring practice---they refused to hire any interns whose last names began with the letter S, T, U, V, W, X, Y, or Z. (First names were not subject to such a whim, which was fortunate for our friend Vinton Cerf.)

Input: Each year's group of interns is considered as a separate trial. A trial begins with a line containing a single integer N, such that $1 \leq N \leq 300$, designating the number of interns that year. Following that are N lines---one for each intern---with a line having a first and last name separated by one space. Names will not have any punctuation, and both the first name and last name will begin with an uppercase letter. In the case of last names, that letter will have an additional constraint that it be in the range from 'A' to 'R' inclusive. The end of the input is designated by a line containing the value 0. There will be at most 20 trials.

Output: For each trial, output a single integer, k, designating the minimum number of teams that were necessary.

Example Input:	**Example Output:**
6 Stephen Cook Vinton Cerf Edmund Clarke Judea Pearl Shafi Goldwasser Silvio Micali 9 Richard Hamming Marvin Minskey John McCarthy Edsger Dijkstra Donald Knuth Michael Rabin John Backus Robert Floyd Tony Hoare 0	3 6

B Bribe

After having done a lot of spying and infiltrating a criminal network, you are now ready to try and dismantle it. This, however, requires the cooperation of a certain number of the henchmen. This in turn requires money in order to bribe them, but due to budget cuts, you only have a limited amount of money.

Fortunately, you are an excellent judge of character, so for each of the henchmen you are considering to bribe, you know what amount of money they will ask for. Furthermore, you know the probability that they will then successfully convert, as opposed to taking the money and making a run for it. There is no particular rush, so after each attempted conversion you can establish whether it was successful or not, before you move on to someone else. Of course, if it was not successful, then you cannot try to bribe this henchman a second time.

Given all this information on the henchmen, the amount of money that you have at your disposal, and the number of henchmen you need to convert, can you work out the probability that this operation will be a success?

Input

On the first line one positive number: the number of test cases, at most 100. After that per test case:

- one line with three space-separated integers n, c and m ($1 \leq n, c \leq 16$ and $1 \leq m \leq 1\,000$): the number of henchmen that are susceptible to bribe, the number you need to convert, and the amount of money that you have, respectively.
- n lines with two space-separated integers b and p ($0 \leq b \leq 1\,000$ and $0 \leq p \leq 100$): the amount of money you need to bribe each henchman, and the probability (as a percentage) that he will be successfully converted, respectively.

Output

Per test case:

- one line with a single floating point number: the probability that you will succeed in converting c henchmen, if you take an optimal approach. This number should be accurate up to 10^{-6} relative or absolute precision.

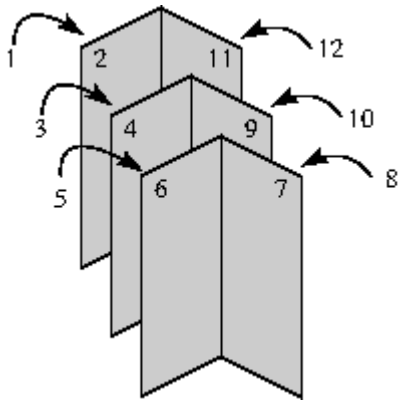
Sample in- and output

Input	Output
2 4 3 1000 300 40 300 50 300 60 300 70 4 2 1000 100 80 700 50 400 20 500 20	0.21 0.408

Problem C: Missing Pages

Long ago, there were periodicals called *newspapers*, and these newspapers were printed on *paper*, and people used to *read* them, and perhaps even share them. One unfortunate thing about this form of media is that every so often, someone would like an article so much, they would take it with them, leaving the rest of the newspaper behind for others to enjoy. Unfortunately, because of the way that paper was folded, not only would the page with that article be gone, so would the page on the reverse side and also two other pages that were physically on the same sheet of folded paper.

For this problem we assume the classic approach is used for folding paper to make a booklet that has a number of pages that is a multiple of four. As an example, a newspaper with 12 pages would be made of three sheets of paper (see figure below). One sheet would have pages 1 and 12 printed on one side, and pages 2 and 11 printed on the other. Another piece of paper would have pages 3 and 10 printed on one side and 4 and 9 printed on the other. The third sheet would have pages 5, 6, 7, and 8.



When one numbered page is taken from the newspaper, the question is what other pages disappear.

Input: Each test case will be described with two integers N and P, on a line, where $4 \leq N \leq 1000$ is a multiple of four that designates the length of the newspaper in terms of numbered pages, and $1 \leq P \leq N$ is a page that has been taken. The end of the input is designated by a line containing only the value 0.

Output: For each case, output, in increasing order, the page numbers for the other three pages that will be missing.

Example input:	**Example output:**
12 2 12 9 8 3 0	1 11 12 3 4 10 4 5 6

ACM Mid-Central Programming Competition 2013

Problem D: Probability Paradox

This problem considers repeated tosses of a fair coin. Each outcome, H or T, has probability 1/2. Any specific sequence of tosses of the same length, like HHH or THH, has the same probability (for example, 1/8 for a sequence of length 3).

Now consider the following two-player game. Each player chooses a distinct pattern of possible coin flips having a common fixed length. For example, the first player might predict HHH while the second predicts THH. The game then begins with both players observing the same coin as it is repeatedly flipped, until one of them witnesses their pattern. For example, if the sequence of observed flips begins HHTHTTHH... the second player in our example wins the game, having just witnessed the pattern THH.

The question that interests us is, given the two players' patterns, how likely is it that the first player wins the game? Because we are flipping a fair coin, many would assume that the patterns are irrelevant and that each player has probability 1/2 of winning the game. However, this is not the case, leading to what is known as the **Probability Paradox**.

For some patterns, it will be that the first player wins with probability precisely 1/2. For example, by symmetry, the patterns TT and HH have equal chances of occurring first.

However, consider again our original example in which the first player chooses HHH and the second player chooses THH. For this particular match-up, the only way that the first player can win is if each of the first three tosses are H. For if the *earliest* HHH were to come somewhere other than at the beginning of the game, the pattern could be represented as

...**?**HHH

where "..." means a possibly empty earliest part of the sequence, and "?" refers to the toss immediately before the HHH. The "?" can not refer to an H, as in

...**H**HHH

because there would have been an earlier HHH that ended the game, the underlined part of ...HHHH. Yet if the preceding toss were a T, as in

...**T**HHH.

then the second player would have already won, having observed pattern THH at the underlined ...THHH. Therefore, when considering pattern HHH vs. THH, the first player wins if and only if the first three flips are H, an event that happens with probability 1/8.

As one more example, if the first player chooses TTH and the second chooses THH, the first player will win with probability 2/3. Your job is to write a program that computes such a probability.

Input: The input will contain one or more datasets, each on a single line. Each dataset will consist of two equal-length yet distinct patterns using only characters H and T. The common pattern length will be in the range from 1 to 9, inclusive. There is a space between the two patterns. The input ends with a line containing only "$".

Output: For each data set, output a single line with the *exact* probability that the first sequence precedes the second in a random sequence of fair coin tosses. The probability must be stated as a rational number, reduced to lowest terms, with a "/" between the numerator and denominator. Because each player has a nonzero chance of winning, this probability will always be strictly between 0 and 1.

Warning: The numerator and denominators for all of the final probabilities can be expressed as 32-bit integers. However, depending on your approach, you may need 64-bit integers (type **`long`** in Java or **`long long`** in C++) for some intermediate calculations, and even then you must be careful.

Example input:	Example output:
TT HH HHH THH TTH THH THHTH HTHTT HTTHHHTHT THHHTHTHH $	1/2 1/8 2/3 15/28 259/452

ACM Mid-Central Programming Competition 2013

Problem E: Letter Cubes

This problem is based on a puzzle by Randall L. Whipkey.

In the game of Letter Cubes, there are a set of cubes, with each face of each cube having a letter of the alphabet, such that no letter appears more than once within the entire set. The maximum number of cubes is 4, allowing for up to 24 of the 26 letters of the alphabet to occur.

Words are formed by rearranging and turning the cubes so that the top letters of all the cubes together spell a word. The 13 words below have been made using a particular set of cubes.

CLIP
CLOG
CONE
DISH
FAZE
FURL
MARE
MOCK
QUIP
STEW
TONY
VICE
WARD

Only 23 distinct letters were used in the above words, so we will tell you the extra information that a B is included on one cube. Can you now determine the letters on each cube? For the above set of words, there is indeed a unique set of cubes. We will state this solution in canonical form as

ABCHTU DEKLQY FGIMNW OPRSVZ

Note that the letters on each individual cube are stated as a string of characters in alphabetical order, and the four 6-letter strings representing the four cubes are also listed in alphabetical order.

A simpler example relies on two cubes, forming the following 11 two-character strings (although the puzzles are more fun when the strings are actual words, they do not need to be):

PI
MU
HO
WE
WO
BE
MA
HI
RE
AB
PY

The only solution for the two cubes forming these strings is

AEIOUY BHMPRW

The same two cubes could be determined without the last pair PY being listed, as long as you were told that there was a Y on one cube. Your job is to make similar deductions.

Input: The input will contain from 1 to 20 datasets. The first line of each dataset will include a positive integer n ($6 \le n \le 30$) and a character c, described below. The next n lines will each contain a string of uppercase letters. Each string will be the same length, call it k, with $2 \le k \le 4$. Following the last dataset is a line containing only 0.

Returning to the issue of the special character, c, on the first input line for each dataset, there will be two cases to consider. Recall that the implicit set of k cubes must use $6*k$ distinct letters on their collective faces If all $6*k$ of those letters appear within the set of strings, then the character c on the first line of input is a hyphen, '-'. Otherwise, the strings have been chosen so that only one letter on the cubes does not appear. In this case, the character c on the first line of input will be that undisplayed letter. (For example, the `B` in our opening puzzle.)

Output: There is one line of output for each dataset, containing a 6-letter string for each cube, showing the letters on the faces of that cube. Each of those strings should have its letters in alphabetical order, and the set of strings should be given in alphabetical order with respect to each other, with one space between each pair. *We have chosen datasets so that each has a unique solution.*

Example input:	Example output:
13 B CLIP CLOG CONE DISH FAZE FURL MARE MOCK QUIP STEW TONY VICE WARD 11 - PI MU HO WE WO BE MA HI RE AB PY 10 Y PI MU HO WE WO BE MA HI RE AB 0	ABCHTU DEKLQY FGIMNW OPRSVZ AEIOUY BHMPRW AEIOUY BHMPRW

Problem F: Digit Sum

When Grace was in third grade, her elementary school teacher assigned her the following problem:

> *What is the smallest possible sum of two numbers that together use the numerals 1, 2, 7, 8, and 9?*

Grace figured out that the answer to this problem is 207 (for example, as 78 + 129), but when the teacher assigned four pages of similar problems as homework, Grace got bored. It turns out that Grace was a rather advanced third grader, so she decided that it would be more fun to write a computer program to solve such problems. Surely you can do the same!

Input: Each problem is described on a single line. The line begins with an integer N, such that $2 \leq N \leq 14$, designating the number of numerals included in the problem. Following that are those N numerals. There will always be at least 2 numerals that are nonzero. The end of the input is designated by a line containing only the value 0.

Output: For each case, output a line with the minimum sum S that can be achieved. Please keep in mind that by standard convention, the numeral *0* cannot appear as the first digit of either summand.

Example input:	**Example output:**
5 1 2 7 8 9 6 3 4 2 2 2 2 9 0 1 2 3 4 0 1 2 3 0	207 447 11257

ACM Mid-Central Programming Competition 2013

Problem G: Cash Cow

Years before Candy Crush became the wildly popular game that may lead developer Saga to a multi-billion dollar IPO, there was an online game named Cash Cow, which remains part of the Webkinz platform.

This single-player game has a board with 12 rows and 10 columns, as shown in Figure 1. We label the rows 1 through 12, starting at the bottom, and the columns *a* through *j*, starting at the left. Each grid location can either have a colored circle or be empty. (We use uppercase characters to denote distinct colors, for example with B=blue, R=red, and Y=yellow.) On each turn, the player clicks on a circle. The computer determines the largest "cluster" to which that circle belongs, where a cluster is defined to include the initial circle, any of its immediate horizontal and vertical neighbors with matching color, those circles' neighbors with matching colors, and so forth. For example, if a user were to click on the blue circle at cell (h10) in Figure 1, its cluster consists of those cells shown with empty circles in Figure 2.

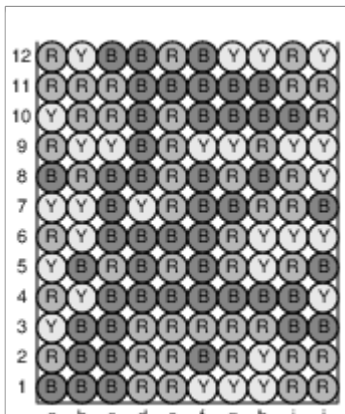


Figure 1

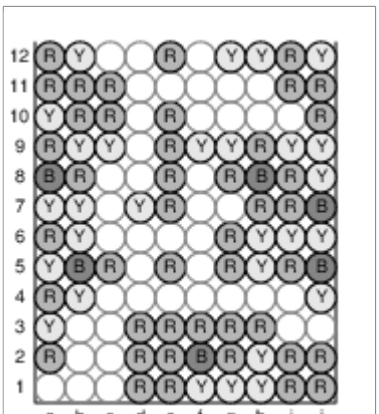


Figure 2

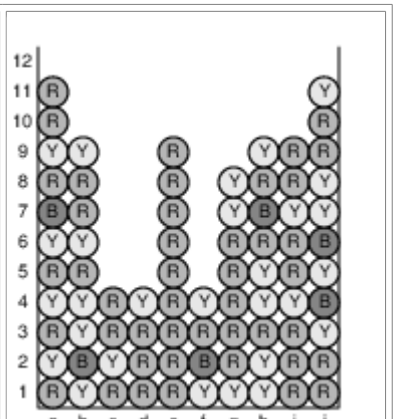


Figure 3

Processing a click on cell h10.

The player's turn is processed as follows. If the indicated grid cell belongs to a cluster of only one or two circles (or if there is no circle at that cell), the turn is wasted. Otherwise, with a cluster of 3 or more circles, all circles in the cluster are removed from the board. Remaining circles are then compacted as follows:

1. Circles fall vertically, to fill in any holes in their column.
2. If one or more columns have become empty, all remaining columns slide leftward (with each nonempty column remaining intact), such that they are packed against the left edge of the board.

For example, Figure 3 shows the board after the cluster of Figure 2 was removed after the click on (h10).

As another example, Figure 4 below, portrays the processing of a subsequent click on cell (j1). During that turn, column (e) becomes empty, and the resulting columns (f) through (j) slide to become columns (e) through (i), respectively. Figure 5 provides one further example in which several columns are compacted.

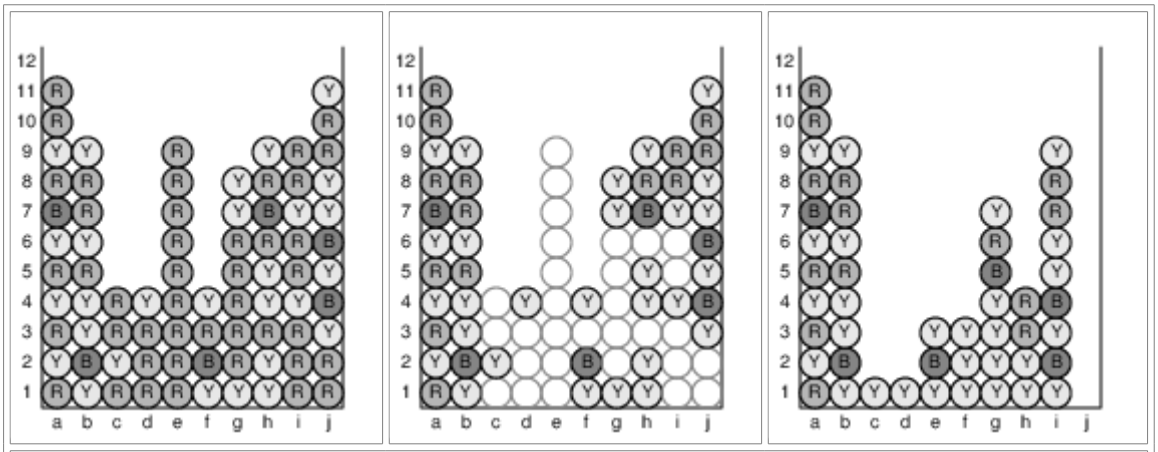


Figure 4: Processing a click on cell j1.

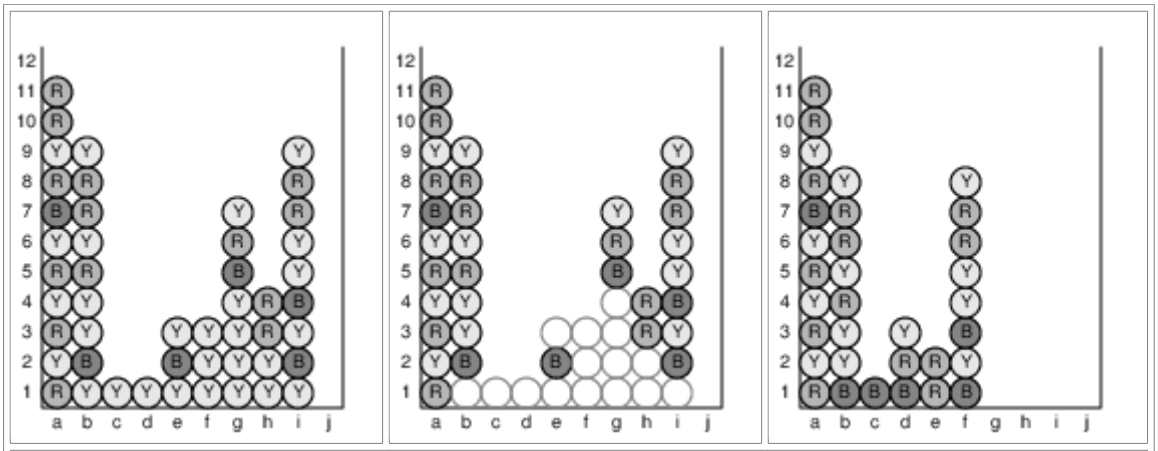


Figure 5: Processing a click on cell g2.

Input: The input will consist of multiple games, each played with a new board. For each game, the input begins with a number T that denotes the number of turns that the player will be making, with $1 \leq T \leq 20$. Following that will be an initial board configuration, which always has 12 rows and 10 columns per row, with uppercase letters used to denote distinct colors. There will never be empty cells within the initial board. Following the presentation of the initial board will be T additional lines of input, each designating a cell of the grid; we rely on the coordinate system illustrated in the above figures, with a lowercase letter, from `a` to `j`, denoting a column and a number from 1 to 12 that denotes a row. We note that if a player clicks on a grid cell that does not currently have any circle, that turn is simply wasted.

The end of the entire input will be designated by a line with the number 0.

Output: For each game, output a single line designating the the number of circles that remain on the board after all of the player's turns are processed.

Example input:	**Example output:**
3 RYBBRBYYRY RRRBBBBBRR YRRBRBBBBR RYYBRYYRYY BRBBRBRBRY YYBYRBBRRB RYBBBBRYYY YBRBRBRYRB RYBBBBBBBY YBBRRRRRBB RBBRRBRYRR BBBRRYYYRR h 10 j 1 g 2 3 YYYYYBBBBB YYYYYBBBBB YYYYYBBBBB YYYYYBBBBB YYYYYBBBBB YYYYYBBBBB YYYYYBBBBB YYYYYBBBBB YYYYYBBBBB YYBYYBBBBB YYBYYBBBBB c 2 c 12 g 1 2 YYYYYBBBBB YYYYYBBBBB YYYYYBBBBB YYYYYBBBBB YYYYYBBBBB YYYYYBBBBB YYYYYBBBBB YYYYYBBBBB YYYYYBBBBB YYYYYBBBBB YYBYYBBBBB YYBYYBBBBB g 1 c 12 0	33 62 2

Problem H: Sort Me

We know the normal alphabetical order of the English alphabet, and we can then sort words or other letter sequences. For instance these words are sorted:

ANTLER
ANY
COW
HILL
HOW
HOWEVER
WHATEVER
ZONE

The standard rules for sorting letter sequences are used:

1. The first letters are in alphabetical order.
2. Among strings with the same prefix, like the prefix AN in ANTLER and ANY, they are ordered by the first character that is different, T or Y here.
3. One whole string may be a prefix of another string, like HOW and HOWEVER. In this case the longer sequence comes after the shorter one.

The Gorellians, at the far end of our galaxy, have discovered various samples of English text from our electronic transmissions, but they did not find the order of our alphabet. Being a very organized and orderly species, they want to have a way of ordering words, even in the strange symbols of English. Hence they must determine their own order. Unfortunately they cannot agree, and every Gorellian year, they argue and settle on a new order.

For instance, if they agree on the alphabetical order

UVWXYZNOPQRSTHIJKLMABCDEFG

then the words above would be sorted as

WHATEVER
ZONE
HOW
HOWEVER
HILL
ANY
ANTLER
COW

The first letters of the words are in *their* alphabetical order. Where words have the same prefix, the first differing letter determines the order, so the order goes ANY, then ANTLER, since Y is before T in *their* choice of alphabet. Still HOWEVER comes after HOW, since HOW is a prefix of HOWEVER.

Dealing with the different alphabetical orders each year by hand (or tentacle) is tedious. Your job is to implement sorting with the English letters in a specified sequence.

Input: The input will contain one or more datasets. Each dataset will start with a line containing an integer *n* and a string *s*, where s is a permutation of the English uppercase alphabet, used as the Gorellians' alphabet in the coming year. The next *n* lines ($1 \le n \le 20$) will each contain one non-empty string of letters. The length of each string will be no more than 30. Following the last dataset is a line containing only 0.

Output: The first line of output of each dataset will contain "year " followed by the number of the dataset, starting from 1. The remaining n lines are the *n* input strings sorted assuming the alphabet has the order in *s*.

Example input:	**Example output:**
8 UVWXYZNOPQRSTHIJKLMABCDEFG ANTLER ANY COW HILL HOW HOWEVER WHATEVER ZONE 5 ZYXWVUTSRQPONMLKJIHGFEDCBA GO ALL ACM TEAMS GO 10 ZOTFISENWABCDGHJKLMPQRUVXY THREE ONE NINE FIVE SEVEN ZERO TWO FOUR EIGHT SIX 0	year 1 WHATEVER ZONE HOW HOWEVER HILL ANY ANTLER COW year 2 TEAMS GO GO ALL ACM year 3 ZERO ONE TWO THREE FOUR FIVE SIX SEVEN EIGHT NINE

I Administrative Difficulties

It is difficult for a spy to do his job without a decent car. The Bureau of Administrative Personnel for Cars (BAPC) spy-car rental company has a large collection of cars that spies can use and handles the resulting administration. Using cars obviously costs money: they require gasoline to operate and repairs are needed quite often, because spies tend to cause accidents a little more often than other drivers.

At the end of the year, all spies need to be billed for the usage of cars in the past year. Last week, there was a major crash in the billing system, rendering it unusable. All that could be recovered was a list of all available types of cars and a log of events for the past year. Using this information, the spy-car rental company wants to obtain a list of the costs for car usage for every spy on record. This list can then be used to send out the bills manually.

Every type of car is registered with a catalog price, the cost to pick up the car and the cost of driving that car per kilometer. The event list contains three types of events: pick-ups, returns and accidents. When a spy picks up a car, he or she must pay the pick-up cost for that car. Once the car is returned, the number of kilometers driven in the car is recorded and the spy must pay for these kilometers. If an accident occurred when the spy was using the car, repairs need to be paid for. Every accident is rated with a severity as a percentage. To repair the car, this percentage of the catalog price is billed to the spy who caused the accident. If any billed cost is fractional, it is rounded up before being added to the bill.

The list of all available types of cars is complete. However, because of the crash, some events in the recovered event log might be missing. The spy-car rental company does not want to present spies with an inconsistent bill, so you should detect inconsistencies in the log entries for each spy. The following conditions hold for a consistent event log:

- A spy will pick up a car before returning it.
- A spy will always return a car they picked up.
- A spy can use at most one car at a time.
- Accidents can only happen when a spy is using a car.

Input

On the first line one positive number: the number of test cases, at most 100. After that per test case:

- one line with two space-separated integers n and m ($0 \leq n \leq 500$ and $0 \leq m \leq 10\,000$): the number of types of cars, and the number of events, respectively.
- n lines with a string N and three integers p, q and k ($1 \leq p \leq 100\,000, 1 \leq q \leq 1\,000, 1 \leq k \leq 100$), all separated by a space: for each type of car, its unique name, its catalog price, its pick-up cost, and its cost per kilometer driven, respectively.

- m lines starting with one integer t ($0 \leq t \leq 100\,000$), a string S and one character e, all separated by a space: the time of the event, the name of the involved spy, and the type of event, followed by:
  - if e = 'p' (pick-up): a string C: the name of the type of car picked up.
  - if e = 'r' (return): an integer d ($0 \leq d \leq 1\,000$): the distance covered in the car last picked up by spy S, in kilometers.
  - if e = 'a' (accident): an integer s ($0 \leq s \leq 100$): the severity of the accident, in percents.

All names of cars and spies consist of at least 1 and at most 40 lowercase letters. There will be at most 500 unique spy names in each test case. The events are given in chronological order.

Output

Per test case:

- one line for every spy referenced in any of the events, containing a string and one integer, separated by a space: the name of the spy and his total car cost. If the event log for the spy is inconsistent, the total cost should be replaced by the string "INCONSISTENT". The lines should be sorted alphabetically by the name of the spy.

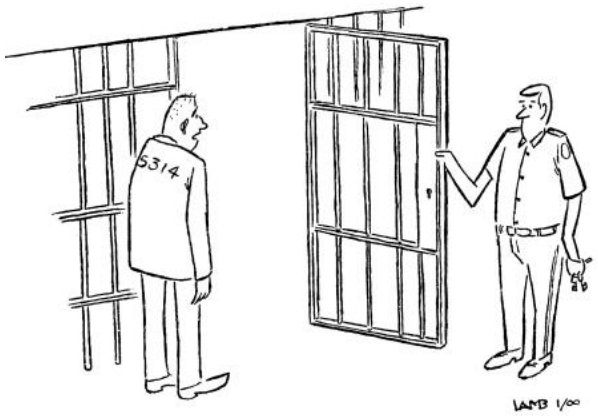
Sample in- and output

Input	Output
1 2 8 bmw 5000 150 10 jaguar 7000 200 25 10 mallory p bmw 15 jb p jaguar 20 jb r 500 35 badluckbrian a 100 50 mallory a 10 55 silva p jaguar 60 mallory r 100 110 silva a 30	badluckbrian INCONSISTENT jb 12700 mallory 1650 silva INCONSISTENT

J Jailbreak

John is on a mission to get two people out of prison. This particular prison is a one-story building. He has managed to get hold of a detailed floor plan, indicating all the walls and doors. He also knows the locations of the two people he needs to set free. The prison guards are not the problem – he has planned a diversion that should leave the building practically void.

The doors are his main concern. All doors are normally opened remotely from a control room, but John can open them by other means. Once he has managed to open a door, it remains open. However, opening a door takes time, which he does not have much of, since his diversion will only work for so long. He therefore wants to minimize the number of doors he needs to open. Can you help him plan the optimal route to get to the two prisoners?



Input

On the first line one positive number: the number of test cases, at most 100. After that per test case:

- one line with two space-separated integers h and w ($2 \leq h, w \leq 100$): the width and height of the map.
- h lines with w characters describing the prison building:
  - '`.`' is an empty space.
  - '`*`' is an impenetrable wall.
  - '`#`' is a door.
  - '`$`' is one of the two people to be liberated.

John can freely move around the outside of the building. There are exactly two people on the map. For each person, a path from the outside to that person is guaranteed to exist.

Output

Per test case:

- one line with a single integer: the minimum number of doors John needs to open in order to get to both prisoners.

Sample in- and output

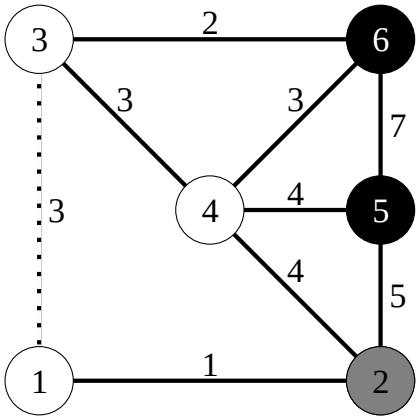
Input	Output
3 5 9 ****#**** *..#.#..* ****.**** *$#.#.#$* ********* 5 11 *#********** *$*...*...* *$*.*.*.*.* *...*...*.* **********.* 9 9 *#**#**#* *#**#**#* *#**#**#* *#**.**#* *#*#.#*#* *$##*##$* *#******#* *.#.#.#.* *********	4 0 9

K Destination Unknown

You are agent B100. A pair of prominently dressed circus artists is traveling over the roads of the city and your mission is to find out where they are headed. All we know is that they started at point s and that they are heading for one of several possible destinations. They are in quite a hurry, though, so we are sure they will not take a detour to their destination.

Alas, prominently dressed as they may be, the duo is nowhere to be seen. Fortunately, you have an exceptional sense of smell. More specifically: your nose will never let you down. You can actually smell they have traveled along the road between intersections g and h.

Where is the elusive duo headed? Or are we still not sure?



A visual representation of the second sample. The duo is travelling from the gray circle to one of the two black circles, and you smelled them on the dashed line, so they could be heading to 6.

Input

On the first line one positive number: the number of test cases, at most 100. After that per test case:

- One line with three space-seperated integers n, m and t ($2 \leq n \leq 2\,000$, $1 \leq m \leq 50\,000$ and $1 \leq t \leq 100$): the number of intersections in the city, the number of individual roads between those intersections, and the number of possible destinations respectively.
- One line with three space-seperated integers s, g and h ($1 \leq s, g, h \leq n$): the intersection the duo started from and the two intersections between which the duo has traveled, with $g \neq h$.
- m lines with three space-separated integers a, b and d ($1 \leq a < b \leq n$ and $1 \leq d \leq 1\,000$), indicating that there is a bidirectional road between intersections a and b of length d.
- t lines with one integer x ($1 \leq x \leq n$): the possible destinations. All possible destinations are distinct and they are all different from s.

There is at most one road between a pair of intersections. One of the m lines describes the road between g and h. This road is guaranteed to be on a shortest path to at least one of the possible destinations.

Output

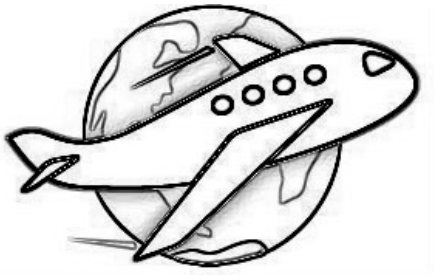
Per test case:

- One line with one or more space-separated integers, indicating the destinations that the duo can still be headed for, in increasing order.

Sample in- and output

Input	Output
2 5 4 2 1 2 3 1 2 6 2 3 2 3 4 4 3 5 3 5 4 6 9 2 2 3 1 1 2 1 1 3 3 2 4 4 2 5 5 3 4 3 3 6 2 4 5 4 4 6 3 5 6 7 5 6	4 5 6

L Flying Safely



Due to budget cuts, even spies have to use commercial airlines nowadays to travel between cities in the world. Although this mode of travel can be very convenient for a spy, it also raises a problem: the spy has to trust the pilot to make sure he is not in danger during the flight. And even worse, sometimes there is no direct flight between some pairs of cities, so that the spy has to take multiple flights to get to the desired location, and thus has to trust multiple pilots!

To limit the trust issues you are asked for help. Given the flight schedule, figure out the smallest set of pilots that need to be trusted, such that the spy can safely travel between all cities.

Input

On the first line one positive number: the number of test cases, at most 100. After that per test case:

- one line with two space-separated integers n ($2 \leq n \leq 1\,000$) and m ($1 \leq m \leq 10\,000$): the number of cities and the number of pilots, respectively.
- m lines with two space-separated integers a and b ($1 \leq a, b \leq n, a \neq b$): a pilot flying his plane back and forth between city a and b.

It is possible to go from any city to any other city using one or more flights. In other words: the graph is connected.

Output

Per test case:

- one line with and integer: the minimum number of pilots that need to be trusted such that it is possible to travel between each pair of cities.

Sample in- and output

Input	Output
2 3 3 1 2 2 3 1 3 5 4 2 1 2 3 4 3 4 5	2 4