PROBLEM A

ABNORMAL 91'S

As we all know a common method of choosing a name for teams participating in programming contests in Iran, is to choose a combination of initials of team members. For instance Jafar, MohammadKazam and Bahman would name their team "JBM". One of the first teams that ever used this method is PMP.

Abnormal 91's, have come up with a fresh idea for naming their team. Instead of using the initials, they concatenate their complete names. As an example mohammadmehdi, amirardalan and seyyedmostafa would name their team:

amirardalanmohammadmehdiseyyedmostafa

Some of 91's even go further and to make their team's name more stylish, they reversed the name of the person in middle. Therefore, the name of the example team would be:

amirardalanidhemdammahomseyyedmostafa

Note that order of team members in the naming are fixed, so names are unique for each team. Given the names of members of one team, as well as name of the team in first 91's style, you have to find the team's name in the second 91's style.

Input (Standard Input)

The first line contains an integer T ($T \leq 100$), the number of tests. Each test consists of two lines. First line contains three strings, the names of team members in some random order. These strings are separated with a single space. The names of all the team members consist of only English lowercase letters and length of each name will not exceed 50 characters. Second line of each test contains a single string, a valid name of the team in the first 91's style.

Output (Standard Output)

For each test print team's name in the second 91's style in a single line. It is guaranteed that the answer is unique.

Sample Input and Output

Sample Input	Sample Output
2 piloop mj poopi piloopipoopmj azhdar gholi gholam gholamradhzagholi	piloopppoopimj gholamazhdargholi

PROBLEM B
BUDGET

The Amirkabir annual contest is held every year to choose teams participating in Tehran regional contest. This year the responsibility of supplying the budget is on "Salim Khan". After tiresome pursuits of Salim Khan, university agreed to pay *V* Rials for the contest's expenditure. Hence, The financial manager of the university has deposited the money into Salim Khan's bank account. However, since they don't like Salim Khan, they have forced him to withdraw the money only through ATM machines.

ATM machines support two different transactions:

- **Money withdrawal**: take some money in cash out of a bank. Limited to at most *R* Rials for each account in each day.
- **Money Transfer**: transfer money to another account. No limitations on the amount of money transferred, but the bank charges *C* Rials for each transfer transaction. The transfer fee is taken from the transferor account.

Salim Khan needs *N* Rials in order to buy T-shirts for the contests and wants to withdraw the money from his bank account as soon as possible. Salim Khan has *K* bank accounts (counting the first one) and is willing to use all of them if needed. Help him to find the least number of days he needs to withdraw *N* Rials out of his bank accounts.

Input (Standard Input)

The first line of input contains a single integer T ($T \le 200$), the number of tests. Each test will be in a separate line containing 5 integers *V, N, R, C* and *K*. ($1 \le N \le V \le 10^7$, $0 \le R, C \le 10^7$, $1 \le K \le 20$) — as explained in problems statement.

Output (Standard Output)

For each test case, in a separate line, output the number of days that Salim Khan needs to withdraw the money for T-shirts. If he will never be able to buy T-shirts for the contest, print "`Contest without T-shirts!`" (Quotes for clarity).

Sample Input and Output

Sample Input	Sample Output
2 500 400 200 5 5 1000 990 80 20 10	1 13

PROBLEM C
COUNTING THE WAYS

Any ACMer likes to solve puzzles just like you. Now we have a puzzle here. Solve it if you can!

This puzzle is a new version of *word search puzzle*. We have a table of English lowercase letters with R rows and C columns. There are N strings of English lowercase letters that should be built from the table.

To build a string S from the table, starting from an arbitrary cell of the table, each time you move to an adjacent cell (i.e. up, down, left or right) given that you do not leave the table. As you move, you write down letters of the cells you pass, on a piece of paper. At the end, if the string on the paper equals to S, we say S is built from the table. Note that you might pass the same cell multiple times.

One way of building a string from the table can be represented as a sequence of ordered pairs, each denoting one cell of the table in order you passed them. Two ways are different if their representative sequences are different. You have to find for each test, the different number of ways to build if from table.

Input (Standard Input)

In the first line there is T ($T \leq 50$), the number of tests that follow. Each test begins with two integers R and C ($1 \leq R, C \leq 50$). Next R lines each contain C English lowercase letters. j-th letter of i-th line is the letter in row i and column j of the table. After that, there will be an integer N ($1 \leq N \leq 50$) which is the number of strings. Each of the next N lines contains a string of length at most 50. Each string consists of only English lowercase letters.

Output (Standard Output)

For each string in each test, print the number of ways that we can build it from the table in a single line. As this number can be very large, you should print it modulo 1,000,000,007.

Sample Input and Output

Sample Input	Sample Output
1 3 4 acmp umic tpmp 3 aut acmicpc pmp	1 4 6

PROBLEM D

DIVIDING THE SEQUENCE

"In mathematics, a geometric progression, also known as a geometric sequence, is a sequence of numbers where each term after the first is found by multiplying the previous one by a fixed non-zero number called the common ratio. For example, the sequence 2, 6, 18, 54, ... is a geometric progression with common ratio 3."

Wikipedia

Now consider the sequence *S* of positive integers. Let *K* be the smallest number, so *S* can be divided into *K* continuous subsequence $s_1, s_2 \ldots s_k$, while satisfying the following condition:

For all *i*, $1 \le i \le K$, s_i is a geometric progression with a positive **integer** common ratio.

We call such *K*, Minimum Divide Number (MDN) of S. For instance consider S to be (1, 2, 4, 12, 24, 48, 96). The sequence *S* can be divided into sets of geometric progressions in two different methods. The MDN for this sequence is 2.

$$s_1 = (1, 2), s_2 = (4, 12), s_3 = (24, 48, 96)$$

$$s_1 = (1, 2, 4), s_2 = (12, 24, 48, 96)$$

You are given a sequence of non-negative integers. You have to replace all zeros with positive integer in such a way that MDN of resulting sequence is the minimum possible.

Input (Standard Input)

The first line contains *T* ($T \le 100$), the number of test cases. Each test consists of two lines. First line contains an integer *N* ($1 \le N \le 10^5$), the length of sequence. Second line contains *N* integers a_i ($0 \le a_i \le 10^9$), the numbers in the sequence.

Output (Standard Output)

For each test, print the minimum MDN of sequence for all replacing of zeros with positive integers.

Sample Input and Output

Sample Input	Sample Output
3 4 2 0 8 0 6 2 0 0 8 0 32 8 1 1 2 4 16 32 64 5	1 2 4

PROBLEM E
EASY TASK

You are a bank teller and head of the bank asked you to do a simple job. Given the information about the entry time of customers and the duration of his job, you are to calculate at which time and which counter he will be receiving services and when he is done.

There are B counters in this bank. Two kinds of customers come to this bank: ordinary people and VIP. For each customer we know his entry time and duration of his job.

We know that N ordinary customers come to the bank. Ordinary people get a number from the machine and wait until that number is called from one of the counters. The bank tellers do not work at their best for the ordinary customers. That is why after a teller deals with an ordinary customer, he rests for some time (which might be different from teller to teller) and then calls another number.

We know that V VIP customers come to the bank. Each VIP customer only works with one of the counters which he had specified before. They also never wait in line with the ordinary customers. Assume that A is a VIP customer. He enters the bank and goes directly to his specified counter. If there are other VIP customers in that counter, he waits until they are done. When there are no more VIP customers in front of him, he will get served. Whatever else the teller was doing at that time (resting or serving an ordinary customer) is interrupted and will be resumed later in the first available time.

Note that:

- If a VIP customer comes to counter i where an ordinary person is just finishing receiving services, the ordinary customer finishes first and then it is the VIP's turn.
- If a VIP customer comes to counter i and the teller in charge of counter i has just finished his rest time, he won't call any ordinary customers and starts dealing with the VIP.
- If two counters i and j want to call the next ordinary customer together, the system will automatically assign the next ordinary customer to the counter with the smaller number and the next ordinary customer to the next counter.
- If the resting time of a teller is over and there are no ordinary customers, he waits for the next customer and calls as soon one comes in.

Input (Standard Input)

The first line begins with an integer T ($T \le 100$), the number of tests. The first line of each test contains an integer B ($1 \le B \le 20$), the number of counters. The second line contains B integers r_i ($1 \le r_i \le 1000$), the rest time for teller at the i-th counter.

In the following line there is an integer N ($1 \le N \le 100$), the number of ordinary people. The following N lines each contain two integers sn_i, ln_i ($1 \le sn_i, ln_i \le 1000$), entry time and duration of the job for the i-the ordinary customer, respectively. For each i < j, $sn_i < sn_j$.

Next line contains an integer *V* ($1 \le V \le 100$), the number of VIPs. The following *V* lines each contain three integers sv_i, lv_i, bv_i ($1 \le sv_i$, $lv_i \le 1000$, $1 \le bv_i \le B$), the entry time, duration of the job and the number of *i*-th VIP's desired counter, respectively. For each $i < j$, $sv_i < sv_j$.

Output (Standard Output)

For each test you should print *N* + *V* lines. The first *N* lines each contains three integers, *i*-th of which is start and finish time of *i*-th ordinary customer and the number counter at which he was served, respectively.

The remaining *V* lines each contains two integers, *i*-th of which is start and finish time of *i*-th VIP.

Sample Input and Output

Sample Input	Sample Output
1 2 2 1 6 1 2 9 11 10 3 40 2 42 4 43 3 7 2 1 1 5 2 1 6 1 1 7 4 2 11 1 2 15 3 2 40 3 1	1 4 1 9 20 1 12 15 2 40 42 2 43 47 1 43 46 2 2 3 5 7 7 8 7 11 11 12 15 18 40 43

PROBLEM F
FINDING PILOOP

There is bad news! Piloop has been lost. However, Poopi and Mj are trying to find him before the local contest so he can be present at the contest. Poopi knows that Piloop has been last seen in Parks City. Therefore, Poopi and MJ have analyzed the city's map thoroughly before actually going there. There are *N* parks within the city and Piloop is waiting in front of one of these parks for his friends to find him. But they do not exactly know which one. There are some of bidirectional roads in the city each connecting two different parks together. Moreover, the city is designed in a way that there is exactly one simple path between each pair of parks.

Now, Poopi and MJ have planned to use the following strategy. They both meet at some park *S* and plan for the search. In the search plan each of them chooses a path such that union of the paths covers all parks. Both paths end at park *T* where the three old friends gather together again. Traversing each road needs one unit of time and since Piloop is waiting in front of the park, Poopi and Mj do not have to search the park itself to find him.

For each starting park *S* the minimum time to complete the search is called the searching time for that park. Since Poopi and MJ are too busy with preparing the problemset of AUT local contest they are asking you to find out the park with minimum search time and calculate its search time.

Input (Standard Input)

In the first line there is an integer T ($T \le 100$), the number of tests. Each test begins with an integer N ($1 \le N \le 1000$), the number of parks in the city. Next $N-1$ lines are streets of Parks City, each of which is a pair of integers u, v ($1 \le u, v \le N$, $u \ne v$) meaning that there is a bidirectional street between park u and park v. It is guaranteed there is exactly one simple path between each pair of parks.

Output (Standard Output)

For each test print two integers in a single line separated by a single space, number of the park with the minimum searching time and it's searching time. If there are multiple parks with the minimum searching time, output the one with the smallest number.

Sample Input and Output

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

PROBLEM G

GUARDING THE COLUMNS

Persepolis is one of the historical and fascinating sites in Iran. As you know, there are numerous columns of stone in this capital of the great ancient Iran. More specifically, there are *K* columns in Persepolis that have circular, rectangular and triangular bases. These columns are built perfectly straight. In other words, if you look at them from sky you may only see the perimeter of their bases. Also, some of these columns may have intersections as depicted in the following figure.

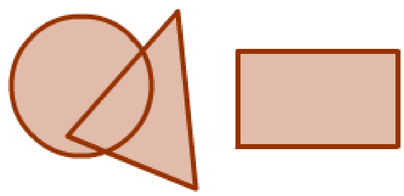

Figure 1- bases of some Persepolis columns

The men in charge of maintenance of columns want to create a safe area around columns. For this purpose they have planted a number of *N* rods (small irony columns) in the ground. They start from a specific rod, tie the rope to it and continue by tying the rope to the next rod and so until they reach the first rod again. The rope between two consecutive rods should be fully **straight**, but it can touch the columns. Consequently one safe and closed area will be formed around all columns in it.

Now the maintenance team wants to know if it is possible to create the safe surrounding using only the existing rods. If it is possible, find is the minimum length of the rope needed to create it? Note that all columns must be surrounded in one connected safe area.

Input (Standard Input)

The first line contains an integer T ($T \le 100$), the number of test cases. The first line of each test contains integer N ($3 \le N \le 300$), the number of rods. Then N lines follow, i-th of which contains two integers x_i, y_i, the coordinates of i-th rod.

In the next line there is an integer K ($1 \le K \le 300$), the number of columns. Then K lines follow, each line represent one of the columns. i-th of these lines starts with a character B. If B is equal to 'C', it means the column's base is a circle and it follows with three integers x, y, r ($1 \le r \le 10^4$) —the coordinates of center and the radius of the base of column, respectively. If B is equal to 'T' it means the column's base is a triangle and it follows with six integers x_1, y_1, x_2, y_2, x_3, y_3 that represent the coordinates of vertices of column's base. And if B is equal to 'R' it means the column's base is a rectangle and it follows with four integers x_1, y_1, x_2, y_2 the coordinates of the lower left and the upper right vertices of the base of column, respectively.

All coordinates are integers with their absolute values less than or equal to 10000. Also rods are different and do not coincide with columns. All columns have positive area.

Output (Standard Output)

For each test, if it is possible to create a safe area using existing rods, output the minimum length of the rope rounded to three digits after the decimal point. Otherwise print "`Impossible`". (Quotes for clarity)

Sample Input and Output

Sample Input	Sample Output
2 4 2 2 6 2 5 5 3 5 1 C 4 3 1 5 6 4 9 8 8 9 3 9 3 8 3 C 6 7 1 T 5 8 7 8 6 9 R 4 6 6 9	12.325 Impossible

PROBLEM H
HITTING THE TARGET

Genius little Ali is in wonderland, where there are all these amazing, fantastic and adventure games! He has played all the trilling games, but he did not enjoy any. Since Ali is genius he loves intellectual games. While he was walking around, suddenly he noticed a nice challenging game and bought a ticket for it.

In this game *N* seesaws are placed in a row and are numbered 1 to *N* from left to right. Two consecutive seesaws have a distance of 1 meter. Also there is a wooden cell 1 meter to the right of the rightmost seesaw. In this cell there is one million million … Rials, as the prize of the one who manages to break the cell. The cell has a resistance of *K*, meaning that to break it, it must be hit with a stone weighting *K* or more.

Initially there is a stone of weight w_i on the left side of *i*-th seesaw. Ali can increase or decrease the weights of these stones arbitrary as long as the weight of the stone on each seesaw remains positive. Total cost of these changes is equal to sum of absolute differences of weight changes for all seesaws.

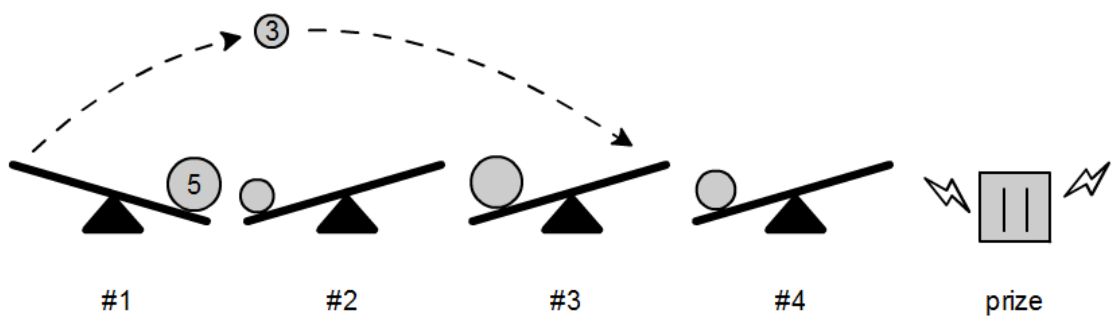

Figure 1- A sample configuration of seesaws and stones

Rule of the game is that if a stone of weight *A* drops on a seesaw with a stone of weight *B*, the seesaw will throw its stone (*A*-*B*) meters to the right (if $A < B$, the seesaw just does nothing). After all changes are made, a stone of weight W_s will drop on the first (leftmost) seesaw. The goal of the game is to change weights of the stones on the seesaws so that wooden cell will be hit with a large enough stone and breaks. Ali not only wants to win the game, but he also wants to do it with minimum cost of changes. Help him with this task.

Input (Standard Input)

The first line contains an integer *T* ($T \leq 100$), the number of tests. In the first line of each test there will be three integers *N*, W_S, *K* ($1 \leq N \leq 1000$, $1 \leq W_S, K \leq 10^9$), the number of seesaws, weight of the first dropping stone and the resistance of the wooden cell, respectively. The next line will consist of *N* weights w_i ($1 \leq w_i \leq 10^9$).

Output (Standard Output)

For each test print the minimum cost of stone changes to break the wooden cell. If Ali cannot win the prize anyway, print "`Impossible`" instead. (Quotes for clarity)

Sample Input and Output

Sample Input	Sample Output
3 5 123 1 130 23 119 20 34 3 2 2 3 2 1 4 80 1 77 82 76 1000	10 Impossible 1

PROBLEM I

INCREDIBLE COMMANDOS

This problem is about AUT Commandos Mission (ACM). AUT Commandos group has C commandos. After some successful intelligence operations, the group discovered that R convicts have plans to rob some banks of AUT City. AUT Commandos want to devise a plan to catch each one of the robbers exactly at the time when he starts the attack and right at the crime scene.

There are N banks in AUT and M bidirectional streets that each one connects two different banks. There is at most one street between two banks and each street has a certain length. AUT commandos know the map of AUT City. Also they know that i-th robber has planned to rob bank b_i at time t_i. If a commando wants to catch i-th robber he should be in bank b_i at time t_i. Catching a robber completely takes K unit of time. Don't forget AUT Commandos are skillful and professional, so they can catch a robber alone and as soon as they have caught one, they can go for another.

At time zero, i-th commando is in bank s_i. Commandos move at most one meter per unit of time (or they can stay still for some time). Now incredible AUT Commandos want to know if there is any way to catch all the robbers.

Input (Standard Input)

The first line begins with an integer T ($T \le 100$), the number of tests. The first line of each test case contains integers N, M, K ($1 \le N \le 100$, $1 \le M \le N(N\text{-}1)/2$, $0 \le K \le 10^6$)—the number of banks, the number of streets and the time needed to catch a robber, respectively. Then M lines follow, i-th of which contains three integers u_i, v_i, l_i ($1 \le u_i, v_i \le N$, $u \ne v$, $1 \le l_i \le 10^6$), meaning that there is a street between bank u_I and bank v_i with length of l_i meters.

In the following line there is an integer C ($1 \le C \le 100$), the number of commandos. Next line contains C integers s_I ($1 \le s_i \le$ N), which specify the starting banks of commandos. Next line contains an integer R ($1 \le R \le 100$), the number of robbers. The following R lines each contain two integers b_i, t_i ($1 \le b_i \le$ N, $1 \le t_i \le 10^6$), meaning that i-th robber wants to rob bank b_i at time t_i. It is guaranteed that for each $i \ne j$, either $b_i \ne b_j$ or $t_I \ne t_j$.

Output (Standard Output)

If AUT commandos can catch all of the robbers print "`They are incredible!`" Otherwise print "`Bad luck!`" (Quotes for clarity).

Sample Input and Output

Sample Input	Sample Output
2 4 3 0	They are incredible! Bad luck!

1 3 10 2 3 2 3 4 3 2 1 2 2 3 8 4 5 3 2 1 1 2 3 2 3 3 1 1 2 2 3 3 6	

PROBLEM J

JUST NO MORE COUNTERS

Once upon a time there was a teacher. He was always telling his students: "You can solve any problem using just counters." The teacher meant that it is better to find a solution using counters even if there are other solutions!

By the way, you are now asked to count the number of parallelograms in a grid of *R* rows and *C* columns (as shown in the figure). Vertices of parallelogram should be on the grid points and at least two edges of parallelogram should be parallel to the axis (i.e. vertical or horizontal).

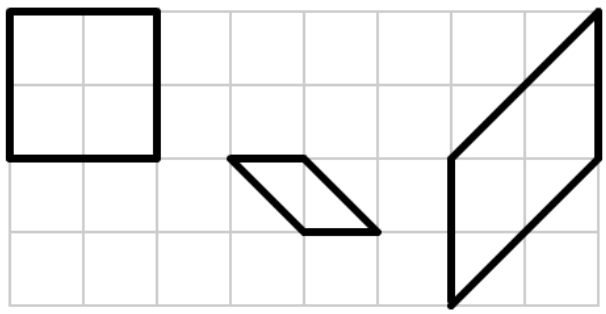

Figure 1- some sample parallelograms placed in a 4x8 grid

As you might already know, the 'execution time' is very important in a programming contest so it seems that you cannot use a counter here! Feel free to calculate the answer in any way you want but make it fast enough. As this number can be very large you should calculate it modulo 1,000,000,007.

Input (Standard Input)

The first line begin with an integer T ($T \le 10000$), the number of tests. Each test will be in a separate line contain two integers R and C ($1 \le R, C \le 10^6$), dimensions of the grid.

Output (Standard Output)

For each test print the number of parallelograms in a single line.

Sample Input and Output

Sample Input	Sample Output
3 1 1 2 3 123 456	1 54 450228282