Problem A. Ancient Keyboard

Input file: `A.IN`
Program file: `A.cpp/A.c/A.dpr/A.java`

The scientists have found an ancient device that works in a strange way. The device has a keyboard and an output tape. The keyboard has 26 keys, with symbols '`A`' through '`Z`' on them. Each key has an LED on it (like the Caps Lock key on some keyboards). Each time you press a key, the LED on it toggles (changes its state from *off* to *on* or vice versa). All LEDs are *off* initially.

To study the output written on the tape, we consider the device in discrete time steps. Suppose we are in time t. If no LED is *on*, no output is written on the tape. If there are i LEDs *on*, the ith letter of the English alphabet is written on the tape. For example, if three LEDs are *on* at a time step, a letter '`C`' is written on the tape. This process repeats at every time step.

You are asked to write a program that simulates the ancient device.

Input

The input contains multiple test cases. The first line of the input, contains t, the number of test cases that follow. Each of the following t blocks, describes a test case.

The first line of each block contains one integer n ($0 \leq n \leq 26$). After this, there are n lines, each containing one capital alphabet letter, followed by two integers a and b, ($0 \leq a < b \leq 1000$). The capital letter shows the key pressed. The number a is the first time step at which the key is pressed and the number b is the second time step at which the key is pressed. During the interval $a, a+1, \ldots, b-1$, the LED of the key is *on*. You can assume that, in each test case, these letters are distinct.

Output

For each test case, output one line containing the output string that is written on the tape.

Sample input and output

A.IN	Standard Output
2 2 X 2 6 Y 4 9 3 A 1 5 B 4 8 C 9 10	AABBAAA AAABAAAA

Problem B. Best SMS to Type

Input file: B.IN
Program file: B.cpp/B.c/B.dpr/B.java

Using SMS today is more than a pleasing hobby. As the number of messages one sends through this service grows, the need to type them fast is better felt. Sometimes, one wonders how fast a message can be typed. Changing some words to their synonyms, might help type the whole message faster, if we were able to quickly calculate the time needed for a specific message.

In the following, we assume that each message is a string of capital English letters and space character. The letters 'A' through 'Z' are assigned to keys '2' to '9', as in the following figure. To type a letter, one should press its key 1, 2, 3, or 4 times, depending on the position of the letter from left to right.

If two consecutive letters of the message are mapped to one key, one should wait for the first letter to be fixed on the screen and then use the key again to type the second one. For instance, to type the letter 'X', one should press '9' twice. If the next letter of the message is not on the same key, one can continue to type the rest of the message. Otherwise, one has to wait for some time, so that the typed 'X' is fixed, and then the next letter ('W', 'X', 'Y', or 'Z') can be typed. To type whitespace, we use the key '1'. As there is no letter mapped to the key '1', the whitespace needs no time to be fixed.

You are given the time needed to press any key, and the time one should wait for a letter to be fixed. Your program should find the minimum time needed to type a nonempty string, given the above rules.

Input

The input file contains multiple test cases. The first line of the input, contains t, the number of test cases that follow. Each of the following t blocks, describes a test case.

The first line of each block contains p and w ($1 \leq p, w \leq 1000$), which show the amount of time in milliseconds for pressing a letter and waiting for it to be fixed, respectively. The second line contains a non-empty string of length at most 1000, consisting of spaces or capital English letters. There is no leading or trailing spaces in a line.

Output

For each test case, output one line showing the time needed to type the message in milliseconds.

Sample input and output

B.IN	Standard Output
1 2 10 ABBAS SALAM	72

Problem C. Changing Phone Numbers

Input file: `C.IN`
Program file: `C.cpp/C.c/C.dpr/C.java`

You are working for OTC, the Olandican Telecommunication Company. Unfortunately, OTC does not decide wisely in assigning telephone numbers to the clients. For example, it did not make a good estimate on demand increase. As a result, it had to increase the number of digits in the local telephone numbers of Oland (the capital) from six to seven, and then again from seven to eight digits.

As usual, a telephone number consists of two parts: an area code, and the local number within that area code. For example, if `021` is the area code of the Oland city, a telephone number in that city may be `0211234567`. Note that no area code is the prefix of another area code.

The process of changing telephone numbers is not easy though. It requires updates to several databases of millions of records. Fortunately, these changes follow a limited number of rules as follows:

1. For all local numbers in a given area code, repeat the ith digit. For example, for the area code `021`, repeating the second digit causes the number `0211234567` change to `02112234567`.
2. For all numbers in a given area code, swap the ith and the $(i+1)$th digit. For example, for the area code `021`, swapping the second and the third digits causes the number `0211234567` change to `0211324567`.
3. Change a given area code. For example, changing `021` to `0211` causes the number `0211234567` change to `02111234567`.

Note that changing area codes in the third rule preserves the property that no area code is the prefix of another. You are to write a program that given the area code information, and the set of all telephone numbers, plus a given set of rules, determine the resulting telephone numbers after the changes.

Input

The first part of the input describes the set of area codes. It starts with a line containing a single integer A ($1 \le A \le 1000$), which is the number of area codes in Olandica, followed by A lines of the following form:

area-code *area-name*

where *area-code* is a string of at least one, and at most 5 digits, and *area-name* is a string of at least one, and at most 20 letters (both uppercase and lowercase). There are no two lines with the same area-code or area-name.

The second part of the input describes the rules applied by OTC. The first line of this part contains a single integer R ($0 \le R \le 10000$), the number of rules applied, followed by R lines of the following form:

year *rule-info*

where *year* is the year in which the rule is applied. You may assume the rule is applied on the first day of the year, and at most one rule is applied each year. The *rule-info* part depends on the specific rule applied. The following list shows the *rule-info* formats corresponding to the rules described in the problem statement:

- `1` *area-name* i
- `2` *area-name* i
- `3` *area-name* *new-area-code*

You can assume that input data is consistent; i.e. in the rules section, indices are not out of range, and an area code will never be prefix of another, at any fixed time. The third part of the input consists of several lines of the following form:

$year_1$ $year_2$ *number*

The query says that in $year_1$, there was a telephone number *number*, and asks for that number in $year_2$ ($year_1 \le year_2$). You must change the number according to the rules applied in the years between

$year_1 + 1$ and $year_2$ in that order. The queries end with a line containing three zeros. The years are positive integers less than 10^9.

In every line in the input containing more than one data item (number or name), the data items are separated by one or more spaces. There may be arbitrary number of leading or trailing spaces too. Each data item is at least one character. Each line of the input is at most 50 characters.

Output

The output contains one line corresponding to each query containing the changed number.

Sample input and output

C.IN	Standard Output
4 021 Oland 0511 Moland 0311 Boland 03121 Kamand 7 2002 1 Moland 2 2001 3 Moland 0515 2003 2 Boland 3 2005 1 Kamand 1 2000 1 Kamand 1 1999 1 Kamand 1 1998 3 Oland 012345 2000 2005 0311243426 2000 2005 05113837462 2000 2005 03121437478 0 0 0	0311244326 051538837462 031214437478

Problem D. Dramatic Multiplications

Input file: `D.IN`
Program file: `D.cpp/D.c/D.dpr/D.java`

Hassan, helping with his younger brother's homework, found out that when you multiply 102564 by 4, its right-most digit moves to the left, and the other digits move one position to the right; i.e. $4 \times 102564 = 410256$. We call a number that has this property when multiplied by n, an n-dramatic number. For instance, 102564 and 128205 are both 4-dramatic. Given two one-digit numbers n and k, the goal is to find the smallest n-dramatic number that its rightmost digit is k.

Input

On the first line of the input, there is an integer t, which is the number of cases that follow. Each test case, is on a line by itself, and contains two integers n and k, where $1 \leq n \leq 9$, and $1 \leq k \leq 9$.

Output

For each test case, output a single integer on a line by itself, which is the smallest n-dramatic number that its rightmost digit is k. If no such number exists, output 0 instead.

Sample input and output

D.IN	Standard Output
2 4 5 2 1	128205 0

Problem E. Entertainment

Input file: `E.IN`
Program file: `E.cpp/E.c/E.dpr/E.java`

ACM-ICPC judges, sometimes, play computer games, such as *Alphariz*. In the game of *Alphariz*, you are given a table (e.g., Table I) of alphabetic characters. At each step of the game, you choose a table entry E containing a non-blank character *ch*. Then, you recursively identify all the friends of E. The friends are the existing table entries containing the same character *ch* which are located immediately to the left, right, up, and down directions of E, and their friends in turn, until no new friends can be located. Then, you replace the characters in E and its found friends with blanks. Table II shows the result of applying the latter rule on Table I, starting at entry E in row 1 and column 2, where *ch* is '`a`'.

a	a	b	b	b
a	b	a	a	b
a	a	a	b	a
a	b	b	a	a

Table I.

		b	b	b
	b			b
			b	a
	b	b	a	a

Table II.

Then, while still in the same step, you shift all the non-blank characters to the left and the blanks to the right, as is shown in Table III. Next, you shift all non-blank characters down and the blanks are shifted up as is shown in Table IV. Note that possible full blank rows or columns are to be deleted.

b	b	b	
b	b		
b	a		
b	b	a	a

Table III.

b	b		
b	b		
b	a	b	
b	b	a	a

Table IV.

You are to write a program that given the initial table and a sequence of selected table entries, apply the above rules for each given table entry, one after another, and report the final resulting table.

Input

The input consists of several test cases. Each test case starts with the character map which is given in m lines of length n characters ($1 \le m, n \le 1000$). Characters in the initial map are all lowercase letters. After this, there is a single line containing a single integer k which is the number of selected table entries, followed by k lines, containing two integers r ($1 \le r \le m$) and c ($1 \le c \le n$) which are the row and the column numbers of the table entry, respectively. Besides, the items are always inside the current table.

Output

The output for the ith test case should start with a line in the following format:

`Test case #`i`:`

After the first line, the final map should be written with the same format as in the input.

Sample input and output

E.IN	Standard Output
aabbb abaab aaaba abbaa 1 1 2 aba bbc 2 1 3 2 2	Test case #1: bb bb bab bbaa Test case #2: a c

Problem F. Fortune at El Dorado

Input file: `F.IN`
Program file: `F.cpp/F.c/F.dpr/F.java`

In his fabulous trip to El Dorado, Kamran made his fortune. After helping the king solve a difficult math problem, the king granted a piece of royal garden to Kamran. The king wrote a letter to the gardener, so that a rectangular region of the royal garden with a specified area, be given to Kamran. You could guess that the trees in El Dorado are made from Gold!

After taking the letter to the royal gardener, Kamran found out that, unfortunately, there is no specific pattern for the location of the trees in the garden. To get the most profit, Kamran talked to the gardener and convinced him that it would not matter much, if Kamran chose the location of his rectangular share, and if he would choose a share smaller than the letter suggests. However, the gardener insisted that Kamran's subgarden should have sides parallel to the garden sides (which is itself a rectangle), and that vertices should have integer coordinates. The area of the piece of land must be positive, so that the king would not suspect anything.

Now, given the locations of the trees in the garden and the maximum allowed area of his share, Kamran should find a rectangular sub-garden with maximum number of trees inside; if a tree is on the border of the sub-garden, he can safely claim it.

Input

The only number of the first line, T, is the number of different instances to be solved. T blocks follow, which describe different independent problems.

The first line of a block, contains two integers $0 \leq F \leq 1000$, the number of trees in the field, and the specified area A. The following F lines, each describes the location of a tree, by two nonnegative integers x, y $(1 \leq x, y \leq 1000)$. No two trees have the same position.

Output

For each block in the input, write a single integer, which is the maximum number of trees that Kamran can obtain.

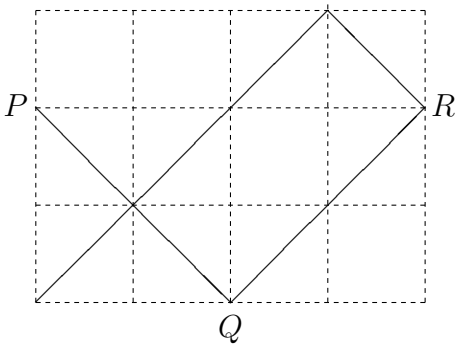
Sample input and output

F.IN	Standard Output
1 3 3 1 1 1 3 1 5	2

Problem G. Griddy Hobby

Input file: `G.IN`
Program file: `G.cpp/G.c/G.dpr/G.java`

Attending a boring weekly session, our professor started drawing on a grid in a page of his calendar. He started at a boundary grid point P; note that P is a corner of one or two grid cells. He drew a diameter of one of those cells and continued on a straight line until reaching point Q on another edge of the grid. Then he started another line from Q, perpendicular to the line PQ until hitting another edge at point R. He kept drawing lines as above, until he could not draw a new line, either because a perpendicular line would not start with a cell diameter or it would fall on an already drawn line. Then he was puzzling how he would be able to count the number of minimal rectangles he has created on the grid. At this time, the chair of the session noticed him and asked him what he was doing. "Sorry, I was designing a problem for the ACM-ICPC Tehran site," he said.



You are to write a program to, given the dimensions R and C of a grid, the coordinates x and y of a point P on one of the edges of the grid, and the direction (up, left), (down, left), (up, right), or (down, right) of the first line, help the professor to find out the number of minimal rectangles.

Input

The input file contains multiple test cases. The first line of the input, contains t, the number of test cases that follow. Each of the following t blocks, describes a test case.

The first line of each block, contains two positive integers R and C, which are the number of horizontal and vertical gridlines, respectively ($2 \leq R, C \leq 1000$). The coordinates of the starting point comes on the next line, as two positive integers y and x, ($1 \leq y \leq R$, $1 \leq x \leq C$); for the upper-left point of the grid, we have $x = y = 1$. The third line of each test case, contains a two-character code which is one of the following:

- "`DR`", the first move is in the down-right direction,
- "`DL`", the first move is in the down-left direction,
- "`UL`", the first move is in the up-left direction,
- "`UR`", the first move is in the up-right direction.

Output

For each test case, output one line containing the number of non-overlapping rectangles formed.

Sample input and output

G.IN	Standard Output
2 4 5 2 1 DR 4 5 4 1 UR	1 3

Problem H. Hotel

Input file: `H.IN`
Program file: `H.cpp/H.c/H.dpr/H.java`

Zebel, the tour coordinator, has reserved a limited number of hotel rooms for his clients. Rooms have different capacities and naturally, different prices. Zebel decides to find the least cost assignment of the tour participants to the available rooms. His strategy is to fill the rooms with appropriate collection of people to minimize the overall room cost, but he is facing some restrictions that no two people of different sex that are not married may stay in the same room, and if a room is assigned to a married couple, no other person may stay in that room. Note that it is not necessary to put a married couple in the same room. It is also possible that we do not fill a room to its capacity.

You are to write a program to help Zebel find a least cost assignment of the tour participants to the reserved hotel rooms.

Input

The only number in the first line is t, the number of test cases that follow. The first line of each test case contains four integer numbers, $0 \leq m \leq 500$ the number of male tour participants, $0 \leq f \leq 500$ the number of female tour participants, $0 \leq r \leq 500$ the number of rooms reserved by Zebel, and $c \geq 0$ which is the number of marriage relations between tour participants. Note that polygamy is not allowed in the tour; i.e. each participant is either single or has a unique mate.

The description of the reserved rooms comes on the following r lines. Each line describes a room, by two integer numbers $1 \leq b_i \leq 5$, and $1 \leq p_i \leq 1000$, which are the capacity and price of this room.

Output

For each test case in the input, output the minimum cost of assigning the rooms to the tour participants. If this is not possible, output the phrase "`Impossible`" instead.

Sample input and output

H.IN	Standard Output
2 2 1 3 1 3 5 2 10 2 4 1 1 1 0 1 4	9 Impossible

Problem I. Intercepting Missiles

Input file:	`I.IN`
Program file:	`I.cpp/I.c/I.dpr/I.java`

Our country is under enemy's attack. Hostile bombers are going to fly towards the capital and destroy everything. To defend the capital, we have a number of missiles, ready to launch and hit the enemy's bombers, before they reach the capital. Unfortunately, there are passenger planes in the sky, which we do not want to hit by our missiles.

We have been able to gather useful information regarding enemy's bombers. While they taxi over our missile defense zone, bombers travel in a fixed altitude. All bombers fly with the same speed. The same applies to airplanes, and our missiles. We know the location of each bomber and each airplane at time zero. Our missiles are placed on the ground, and their locations are also known.

You should write a program, that given the information about the bombers, and the locations of passenger planes in the sky, determines the maximum number of bombers that can be successfully hit by our missiles. Then, we pray for the rest of bombers to explode by themselves!

To simplify your task, The following are assumed:

- We consider a flat two-dimensional model of the earth. Thus, the y-coordinate of the airplanes, and attacking bombers, does not change during their movement over our defense zone.
- Each defending missile can be fired, at time zero or afterwards.
- The y-coordinate of bombers, and airplanes are distinct positive integers.
- Each bomber or airplane, has unit length, while our missiles have no length.
- If a missile hits, or just touches the edge of a target in the sky, our missile will explode, while the target keeps moving normally on its path for a while before it explodes. Assume that the hit bomber will explode after passing all defending missiles, i.e. after surpassing the x-coordinate of all our missiles. Note that during this time, it may be hit by other missiles.

Input

The input file contains multiple test cases. The first line of the input, contains t, the number of test cases that follow. Each of the following t blocks, describes a test case, and is preceded by a blank line.

The first line of each block contains three integers, m, n and k ($0 \leq m, n, k \leq 300$), which are the number of bombers, airplanes, and our missiles, respectively. The second line contains three integers, v_m, v_n, and v_k ($1 \leq v_m, v_n, v_k \leq 10,000$), which are the respective velocity of bombers, airplanes and our missiles. Airplanes and enemy's bombers, are assumed to move to the right, for simplicity, while our missile move upwards, without changing their x-coordinates.

Next come m lines, that each gives the x and y coordinate of the head of a bomber, at time zero. The planes in the sky, are described similarly, in the following n lines. The last line contains k integers, each being the x-coordinate of a defending missile which is ready to launch. The coordinates are all nonnegative integers less than $10,000$.

Output

For each test case, output one line showing the maximum number of hit bombers, without any airplane being hit. Follow the format of the sample.

Sample input and output

I.IN	Standard Output
1 2 1 3 1 1 1 0 100 1 99 2 50 100 200 300	Mission #1: 1 bomber(s) exploded

Problem J. Joy of Mobile Routing

Input file: `J.IN`
Program file: `J.cpp/J.c/J.dpr/J.java`

A computer professor, who is fond of programming, has come to Tehran to pay a visit to our regional contest. After hearing about such an event, he has tried his best to be able to participate in the contest as a visitor. He would decide to hold such contests in his university, if he finds it fun enough.

As could be predicted, he has been lost in our large and crowded city, Tehran. He does not know where to go. Getting really tired, he is standing in a city intersection when he suddenly remembers the phone number of a friend in the organizing committee, and instantly calls him. The friend understands the situation and wants to route the professor through mobile phone calls. He plans to give the professor directions at each intersection; i.e. he tells the tired professor in which direction to move, and after reaching the next intersection, there is another phone call, to get the new direction, etc. This process continues, until the professor sees his friend at the destination intersection, waiting for him. Due to poor telecommunications installment, not all city intersections are accessible through mobile network.

The professor should get to the regional contest location as soon as possible. There are a number of antennas located in some of city intersections. A city intersection is accessible through the network if an antenna can see the point; i.e. there is an imaginary straight line connecting the intersection to any part of the antenna. The line can never cross a building, but touching a boundary does not block the mobile connection. The city is composed of $R \times C$ rectangular blocks. Each block is a building having 10 meters width and length.

You should write a program to find out the shortest possible mobile route; i.e. a route that lets the professor call his friend from each intersection, until reaching the destination. The program is given R, C and height of each building and each antenna, and the location of antennas.

Input

The only number of the first line, T, $(1 \leq T \leq 20)$ is the number of different test cases to be solved. T input blocks follow, which describe different independent test cases.

The first line of a block, contains two integers R, C $(1 \leq R, C \leq 50)$, the number of rows and columns of the city map, respectively. Next R lines, each contains C non-negative integers, $0 \leq H_{ij} \leq 1000$, describing the building heights. The first entry is the leftmost building in the uppermost row. The coordinates of the starting and ending points are given next; the row coordinates come first in each line. The lower and rightmost intersection is (R, C), while the opposite intersection is $(0, 0)$.

Followed is A, $(0 \leq A \leq 100)$, the number of mobile antennas. Next A lines describe the antennas by three numbers, r, c, $(0 \leq r \leq R, 0 \leq c \leq C)$, and h, $(0 \leq h \leq 1000)$. It means that there is an antenna with height h at the intersection (r, c).

Output

For each block of the input, write a single integer, which is the least distance the professor is to travel before getting to the contest, in meters. If there is no mobile route, and the professor's friend is to choose another way for the routing, output -1 instead.

Sample input and output

J.IN	Standard Output
1 3 2 0 10 20 15 5 4 3 0 1 2 1 0 0 6	40