The 2017 ACM ASIA Region Programming Contest

(Kolkata-Kanpur) Site

Onsite Contest Problems
(11 Problems)

24th December 2017

ACM Asia Regional (Kolkata-Kanpur Site) Programming Contest

December 24, 2017

Instructions

There are **Eleven (11)** problems for each team to be completed in **five hours**. Standard Inputand Output files are to be used for each problem. If you test your program using CodeChefplatform, it will automatically redirect input from the sample input file to your program. Outputmust correspond exactly to the provided sample output format, including (mis)spelling andspacing. Multiple spaces will not be used in any of the judges' output, except where explicitlystated.

Your solution to any problem should be submitted for judging using the CodeChef platform. Onceyou have submitted the solution, it will reach the CodeChef judge and the result will be shown.The judgment may be "Correct Answer", meaning that your submission was judged to becorrect. Otherwise you will get a message indicating the problem with your program. Forexample, the message may be "Wrong Answer", "Compilation Error", "Runtime Error", "TimeLimit Exceeded" etc.

You can submit only one source file.

You can use any of the standard library functions that your chosen programming language provides. In addition, you can use the math library in C/C++. You cannot use any other librarythat requires an extra flag to be passed to the compiler command. If you do this, the judges willprobably find a code "compilation error" in your program. Your program is not permitted toinvoke any external programs. *Violation of this rule may lead to disqualification from the contest.*

Programming style is not considered in this contest. You are free to code in whatever style youprefer. Documentation is not required. The judges will only test whether the inputoutputbehavior of your program is correct or not. The regional contest director and judges areempowered to adjust for or adjudicate unforeseen events and conditions. Their decisions arefinal.

Teams are ranked according to the most problems solved. Teams who solve the same numberof problems are ranked by least total time. The total time is the sum of the time consumed foreach problem solved. The time consumed for a solved problem is the time elapsed from thebeginning of the contest to the submittal of the accepted run plus 20 penalty minutes for everyrejected run for that problem regardless of submittal time. There is no time consumed for aproblem that is not solved.

Teams are allowed to bring 25 pages of printed materials stamped by the authority in the contest area. Further, the team is not allowed to discuss/ talk with any other team by any meanswhatsoever, during the contest period. *Any such attempt, if it is detected, may lead to immediatedisqualification of all the teams involved.*

Problem code: **ZUBAPCNT**
Problem name: **A - Appearance Count**

You will be given **m** strings. For each of those strings, you need to count the total number of appearances of that string as substrings in all possible strings of length **n** containing only lower case English letters.

A string may appear in a string multiple times. Also, these appearances may overlap. All these must be counted separately. For example, *aa* appears thrice in the string *aaacaa*: ***aa****acaa*, *a****aa****caa* and *aaac****aa***.

Input

- The first line contains one integer, **T**, the number of test cases. The description of each test case follows:
- The first line of each test case will contain two integers **n** and **m**.
- The i^{th} of the next **m** lines will have one string in each line. All the strings will consist only of lower case English letters.

Output

- For each test case, print **"Case x:"** (without quotes. **x** is the test case number, 1-indexed) in the first line.
- Then print **m** lines. The i^{th} line should contain the number of appearances of the i^{th} string in all possible strings of length **n**. As the numbers can be very large, print the answers modulo $\mathbf{10^9+7}$

Constraints

- $1 \leq \mathbf{T} \leq 100$
- $1 \leq \mathbf{n} \leq 100000$
- $1 \leq \mathbf{m} \leq 1000$
- $1 \leq$ Total length of all strings in one test case $\leq 5 * 10^5$
- $1 \leq$ Total length of all strings in one test file $\leq 5 * 10^6$

Example

Input:

```
3
2 1
aa
2 1
d
12 3
cdmn
qweewef
qs
```

Output:

```
Case 1:
1
Case 2:
52
Case 3:
443568031
71288256
41317270
```

Explanation:

Testcase 1: *aa* is the only string of length 2 which contains *aa* as a substring. And it occurs only once. Hence the answer is 1.

Problem code: **ZUBGOLD**
Problem name: **B - Get The Gold**

A man in his death-bed told his lazy son that throughout his life he had worked very hard. With the money he saved he has bought gold. This gold is hidden in a forest which is represented by a 2 dimensional plane in 3 dimensional space. There are two special trees **A** and **B** in the forest. The son needs to find these trees, and a special stone **S**. He must walk from **S** to **A** along the plane representing the forest, and then take a right turn and walk the same distance (ie. walk perpendicular to **SA**), to get to a point **C**. He must then come back to **S**. Next, he should walk from **S** to **B**, take a left turn and then walk that distance again to reach the point **D**. Half way between **C** and **D** is the place where the gold is hidden.

The forest is a plane defined by the points **S(x_0,y_0,z_0)**, **A(x_1,y_1,z_1)** and **B(x_2,y_2,z_2)**. **S, A**, and **B** will not be collinear. You need to find the coordinates of the gold. Each input set consists of the coordinates of the points **S, A** and **B** in separate lines. It is guaranteed that the origin **O**(0,0,0) will always be above the forest surface. That is, you standing on the forest surface, if you look up, (0, 0, 0) will be in the sky.

Input

- The first line contains one integer, **T**, the number of test cases. The description of each test case follows:
- Three lines will be printed for each test case.
- The first line will contain three space-separated numbers **x_0, y_0, z_0**.
- The second line will contain three space-separated numbers **x_1, y_1, z_1**.
- The third line will contain three space-separated numbers **x_2, y_2, z_2**.
- An empty line will be printed after each test case.
- All the numbers will have two digits after the decimal point.

Output

For each test case, print three real numbers, **x y z**, in a new line where **(x,y,z)** are the coordinates of the gold. Your answer will be considered correct if the absolute error is less than 10^{-6}.

Constraints

- $1 \leq \mathbf{T} \leq 100000$
- $-10^6 \leq \mathbf{x_0, y_0, z_0, x_1, y_1, z_1, x_2, y_2, z_2} \leq 10^6$

Example

Input:

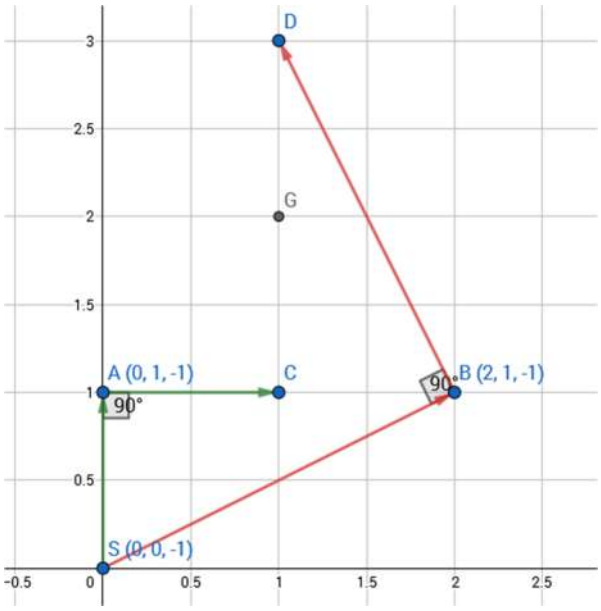
```
2
0.00 0.00 -1.00
0.00 1.00 -1.00
2.00 1.00 -1.00

-8.00 4.65 5.72
-6.14 -3.43 -3.87
-5.50 1.56 8.60
```

Output:

```
1.00000000 2.00000000 -1.00000000
-2.29963939 -6.31303751 4.33640147
```

Explanation

Test case 1: The figure below shows the scenario:



The plane represented by the three input points is the plane z = -1. We have **S**(1.00, 1.00, -1.00). Going from **S** to **A**, taking a right turn, and going the same distance as **SA** will take you to the point **C**(1.00, 1.00, -1.00). Going from **S** to **B**, taking a left turn, and going the same distance as **SB** will take you to the point **D**(1.00, 3.00, -1.00). The gold is at the midpoint of segment **CD**, which is (1.00, 2.00, -1.00).

Problem code: **ZUBRIDER**
Problem name: **C - Club of Riders**

At a shopping mall, every now and then, a naughty kid steals some chocolates from different chocolate shops and runs away. The guards in the shopping mall were unable to catch the kid as he has got a fast scooter.

Each scooter has a performance value. Also, this kind of scooter needs a skilled rider to perform well. The total performance of a rider is measured by the value of scooter performance multiplied by rider's skill. A rider **X** can catch a rider **Y** only if **X**'s total performance is more than **Y**'s.The shopping mall manager has bought one scooter for each guard. However, as he has bought them from an wholesale offer, their performance values need not be the same.

Each guard gets a scooter and of course, a scooter can be assigned to only one guard. Now, the manager wants to assign the scooters to the guards. Everyday the guards are on duty in different places and that's why each has to be able to catch the kid and only then an assignment will be valid.

So in short, given the skill of that kid, his scooter's performance value, each guard's skill, and each scooter's performance value, find out the number of valid assignments possible. In a valid assignment, each guard will be able to catch the kid. As the result can be very large, find the mod $10^9 + 7$ of the result.

Input

- The first line of the input contains an integer **T** denoting the number of test cases. The description of each test case follows.
- The first line of each test case contains two integers $\mathbf{K_p}$ and $\mathbf{K_s}$ denoting the kid's performance value and his scooter's performance value respectively.
- The next line contains a single integer **N** denoting the number of guards.
- The third line contains **N** integers $\mathbf{G_1, G_2, G_3, \dots, G_N}$ denoting the performance values of the guards.
- The fourth line of each test case contains **N** integers $\mathbf{S_1, S_2, S_3, \dots, S_N}$ denoting the performance values of the scooters.

Output

For each test case, print "Case i: ", and then the answer (mod $10^9 + 7$), where i is the testcase number, 1-indexed.

Constraints

- $1 \leq \mathbf{T} \leq 10$
- $1 \leq \mathbf{N, K_p, K_s, G_i, S_i} \leq 10^6$

Example

Input:

```
2
2 3
3
3 1 3
7 3 4
2 3
3
3 1 3
2 7 4
```

Output:

```
Case 1: 2
Case 2: 0
```

Explanation

Testcase 1: The valid arrangements are the following:

- [1st guard, 2nd scooter], [2nd guard, 1st scooter], [3rd guard, 3rd scooter]
- [1st guard, 3rd scooter], [2nd guard, 1st scooter], [3rd guard, 2nd scooter]

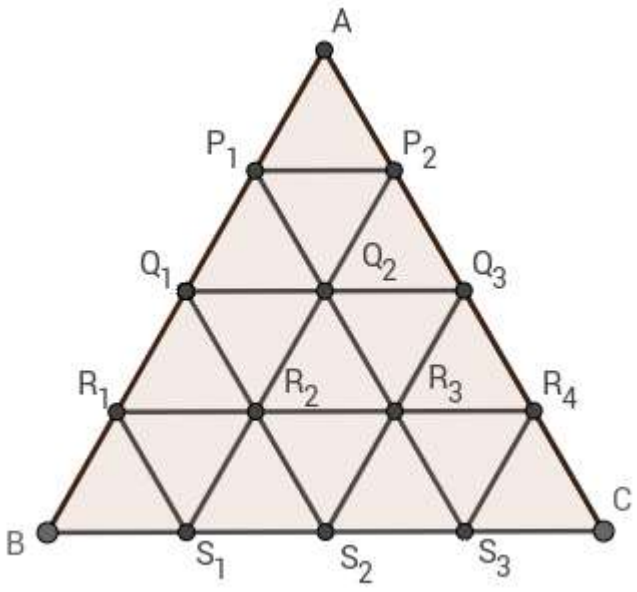
Testcase 2: There is no valid arrangement.

Problem code: **ZUBTRCNT**
Problem name: **D - Triangle Count**

You are given an equilateral triangle **ΔABC** with the side **BC** being the base. Each side of the triangle is of length **L**. There are **L-1** additional points on each of the sides dividing the sides into equal parts of unit lengths. Points on the sides of the triangle are called *major* points. Joining these points with lines parallel to the sides of **ΔABC** will produce some more equilateral triangles. The intersection points of these parallel lines are called *minor* points.

Look at the picture below. It contains

- Major points: **A, B, C, P_1, P_2, Q_1, Q_3, R_1, R_4, S_1, S_2, S_3** (note that we consider **A, B, C** as major points as well)
- Minor points: **Q_2, R_2, R_3**
- Equilateral triangles **$\Delta P_1Q_1Q_2$, $\Delta Q_2S_1S_3$**, etc



We consider an equilateral triangle to be valid if

- Each of its vertices is either a major or a minor point, *and*

- The distance from its base (the base of a triangle is the side parallel to **BC**) to **BC** is less than the distance from the other vertex of the triangle (i.e. opposite vertex that doesn't lie on the base of triangle) to **BC**.

In the figure above, $\Delta Q_2P_1P_2$ is not a valid triangle but $\Delta Q_2R_2R_3$ is a valid triangle.

You will be given **L**, the length of the original triangle **ΔABC**. You need to find out the number of valid equilateral triangles with side length exactly **K**.

Input

- The first line of the input contains an integer **T** denoting the number of test cases. The description of each testcase follows.
- Each test case has one line containing two space-separated integers: **L** and **K**.

Output

For each testcase, print "Case i: ", and then the answer, where i is the testcase number, 1-indexed.

Constraints

- $1 \leq \mathbf{T} \leq 500$
- $1 \leq \mathbf{L}, \mathbf{K} \leq 5000$

Example

Input:
```
2
4 3
4 4
```

Output:
```
Case 1: 3
Case 2: 1
```

Explanation

The figure presented in the problem description is a triangle with side length 4.

In **testcase 1**, the valid triangles are ΔAR_1R_4, ΔP_1BS_3, ΔP_2S_1C
In **testcase 2**, the only valid triangle is **ΔABC**

Problem code: **ZUBPASSR**
Problem name: **E - Password Riddle**

"What's your password, Ranju?"
"one two three four", Ranju said

Do you think it's *1234*? No, it's not. As it turns out that Ranju has made a riddle with his password. It's actually *2444*. Clever, isn't it? It's one *2* and three *4*s.

However, if we told Ranju to write down his password, he would write this - *"one two, three four."* Note that comma. It helps a lot, right? the comma separates two segments of the password.

Let's formalize things a little. The rules Ranju made to describe a password (containing digits from *0-9* only) are:

- Ranju will use only the words *zero, one, two, three, four, five, six, seven, eight, nine, odd, even*. These are called *literal*s.
- He will also use *comma*s to separate the *segment*s and a *full stop* to mark the end.
- Each segment will consist of one *clause*. A clause can be either:
  - A literal followed by a clause. *or*,
  - A literal.
- A literal matches with a digit/number as per following rules:
  - *zero, one, two, ... , nine* match with *0, 1, 2, ... , 9* respectively.
  - *odd* matches with any of the 5 odd digits/numbers - *1, 3, 5, 7, 9*
  - *even* matches with any of the *5* even digits/numbers - *0, 2, 4, 6, 8*
- For a clause which has a literal followed by another clause, we call the literal *directive literal* and the following clause *child clause*. This kind of clause matches with the occurrence of the child clause a number of times directed by the directive literal.

Let's consider the clause *two odd three*. Here the directive literal is *two* and child clause is *odd three*. The child clause also has a directive literal(*odd*) and a child clause(*three*). the clause *odd three* matches with *3*, *333*, *33333*, etc. The clause *two odd three* matches with *33* (two occurrences of *3*), *333333* (two occurrences of *333*),

etc. Note that it doesn't match with *3333* (the concatenation of *3* and *333*). The child clauses should be identical.

You will be given a password riddle and a password. Your job is to find out if it is possible to match the riddle with the password. You may assume that consecutive occurrences of the same digit in the password will **not** be described with more than one segments in the riddle. While matching the password, you are not required to use all of the segments (i.e. you can **skip** some segments. Please check the last sample test case for clarification).

Input

- The first line of the input is the number of test cases, **T**. Description of each test case is given below.
- The first line contains the riddle. It may have several segments as described above. The last literal of each segment will be followed by a comma or a full stop (without any space). Please check the sample input.
- The second line contains an integer indicating the length of the password.
- The following line contains the password. The password has digits from *0* to *9* with no white space.

Output

For each testcase, print "Case i: ", and then the answer, where i is the testcase number, 1-indexed. The answer should be **"MAY BE"** (if it is possible to match the riddle with the password) or **"NO"** (if the password doesn't match). Don't print any quotation marks.

Constraints

- $1 \leq \mathbf{T} \leq 10$
- $1 \leq$ Number of segments in a riddle ≤ 100
- $1 \leq$ Number of literals in a riddle ≤ 600000
- $1 \leq$ Length of a password ≤ 600000

Example

Input:

```
7
two odd three.
6
333333
two odd three.
4
3333
odd five, two three.
11
55555555533
one odd odd, zero six, five even, six.
9
111888886
four odd three, two two.
7
3333322
one four, one two.
2
24
one two, one five.
1
5
```

Output:

```
Case 1: MAY BE
Case 2: NO
Case 3: MAY BE
Case 4: MAY BE
Case 5: NO
Case 6: NO
Case 7: MAY BE
```

Explanation

Testcase 1 and **testcase 2** are described in the statement.

In **testcase 3**, there are nine *5*s and two *3*s which matches with the riddle.

In **testcase 4**, we can match the clause *odd odd* with three occurrences of *1*.
Hence *one odd odd* matches with *111*. *zero six* matches with zero occurrences of *6* (we can skip the segment *zero six*). *five even*matches with *88888*.
Finally, *six* matches with *6*.
In **testcase 5**, we cannot match the password with the riddle.

In **testcase 6**, the riddle doesn't match with the password. However, the riddle would match with *42*.

In **testcase 7**, we can skip the segment *one two* and match *one five* with *5*.

Problem code: **ZUBHTREE**
Problem name: **F - Huge Number of Trees**

Hermione needs to solve a tough homework problem in Arithmancy class. Professor Vector was discussing trees for some time and now he gives the problem: Say, there are **n** + 1 nodes numbered 0, 1, 2, ..., **n**. Node 0 is always the root of the tree. How many different trees can be formed using the remaining **n** nodes so that all leaves have the same *depth* and the *degree* of each non-leaf node is at least **d**? Here, *depth* of a node **u** is the number of edges we need to traverse from the root (Node 0) to reach **u**. A tree is different from another tree if there exists at least one pair of nodes **(u, v)** for which an edge between **u** and **v** is present in one tree, but absent in the other one. Hermione wants your help to write a program to solve this problem.

Note that by *degree*, we mean the total number of neighbors, not just the number of children.

Input

- The first line of the input contains a positive integer, **T** which denotes the number of test cases.
- For each case, there is one line of input containing two integers: **n** and the minimum degree of each non-leaf node, **d**.

Output

For each test case, print "Case i: ", and then the answer (mod $10^9 + 7$), where i is the testcase number, 1-indexed.

Constraints

- $1 \leq \mathbf{T} \leq 2000$
- $2 \leq \mathbf{n} \leq 500$
- $2 \leq \mathbf{d} \leq 10$

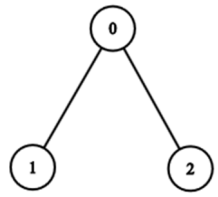
Example

Input:
```
3
2 2
3 2
4 2
```

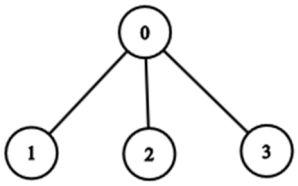
Output:
```
Case 1: 1
Case 2: 1
Case 3: 13
```

Explanation

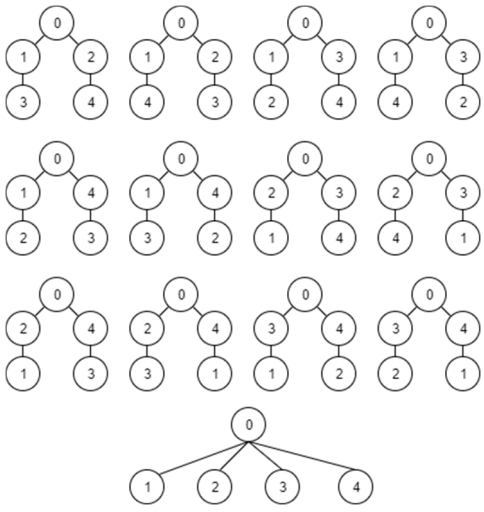
Testcase 1: The only valid tree is shown below:



Testcase 2: The only valid tree is shown below:



Testcase 3: The 13 different valid trees are as shown below:



Problem code: **ZUBPALK**
Problem name: **G - Palace of King**

A king in a distant land wants to build a new palace for himself. There are some plots of land in the country. Some of them don't have any owner, some of the plots are owned by the citizens of the country. Being a good person, the king wants to pay the appropriate amount of money for the land he uses. If he uses some portion of a plot, he needs to buy the whole plot. Also, this is a weird country where the plots owned by the citizens may overlap. For an area, the king needs to buy all the overlapping plots to use that area. To use some area not owned by any citizen, the king doesn't need to spend any money.

Formally, the country is an **M × N** grid with **L** plots owned by the citizens. Each of these plots is rectangular in shape. The $\mathbf{i^{th}}$ plot has its lower left corner at $\mathbf{(x_i, y_i)}$, length $\mathbf{l_i}$ (along with the *x*-axis), width $\mathbf{w_i}$(along with the *y*-axis), and price $\mathbf{p_i}$. The palace the king is going to build should be rectangular in shape as well. Please help him to find the rectangular place with the largest area which he can use without spending more than **C** amount of money.

(0,0) is the lower left corner of the country and **(M, N)** is the upper right corner. All the plots owned by the citizens are inside this area. The palace should be inside this area as well. The sides of the palace should be axis-parallel.

Input

- The first line of the input contains an integer **T** denoting the number of test cases. Description of each test case is given below.
- The first line of each test case contains three integers **M**, **N**, and **C**.
- The second line contains **L**.
- Each of next **L** lines contains the description of a plot owned by a citizen. $\mathbf{i^{th}}$ of these line has five integers: $\mathbf{x_i, y_i, l_i, w_i}$, and $\mathbf{p_i}$.

Output

For each test case, print "Case i: ", and then the answer (mod $10^9 + 7$), where i is the testcase number, 1-indexed. The answer should be the largest area of the rectangular shaped land that can be used without spending more than **C** amount of money.

Constraints

- $1 \leq \mathbf{T} \leq 10$
- $1 \leq \mathbf{M, N} \leq 1000$
- $0 \leq \mathbf{C} \leq 1000000000$
- $0 \leq \mathbf{x_i} \leq \mathbf{M - l_i}$
- $0 \leq \mathbf{y_i} \leq \mathbf{N - w_i}$
- $1 \leq \mathbf{l_i, w_i}$
- $1 \leq \mathbf{p_i} \leq 100000$
- $1 \leq \mathbf{L} \leq 1000$

Example

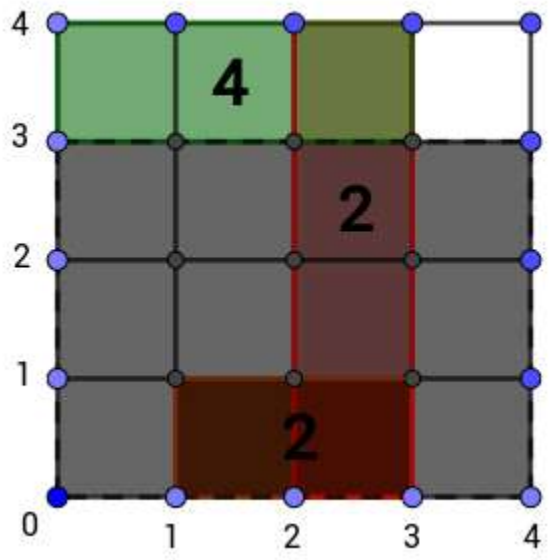
Input:

```
1
4 4 6
3
1 0 2 1 2
2 0 1 4 2
0 3 3 1 4
```

Output:

```
Case 1: 12
```

Explanation



The image above shows the lands owned by the citizens and the palace with area **12** (lower left corner at **(0,0)** and upper right corner at **(4,3)**).

Problem code: **ZUBSPOIL**
Problem name: **H - Unpredictable Array**

Value of an array is defined as the sum of the absolute differences between pairs of consecutive elements in the array. Formally, for a given array $\mathbf{A = \{A_1, A_2, A_3, \dots, A_n\}}$, value(**A**) = $|\mathbf{A_1 - A_2}| + |\mathbf{A_2 - A_3}| + |\mathbf{A_3 - A_4}| + \dots + |\mathbf{A_{n-1} - A_n}|$, where |**x**| means the absolute value of **x**.

You will be given an array **A** of **n** integers and **q** updates. Each update will have two integers **x** and **y**. For this update, you should replace *all* the occurrences of element **x** in the array with **y** and output the value of the new array.

Input

Input starts with an integer **T**, denoting the number of test cases.

The first line of each case contains two integers **n** and **q**. The next line contains **n** space separated integers $\mathbf{A_1, A_2, A_3, \dots, A_n}$ forming the initial array.

Each of next **q** lines contains two space-separated integers **x** and **y**.

Output

For each test case, print **"Case t:"** (without quotes. **t** is the test case number) in the first line. Then print **q** lines. The $\mathbf{i^{th}}$ line should contain the value of the array after the $\mathbf{i^{th}}$ update.

Constraints

- $1 \leq \mathbf{T} \leq 10$
- $1 \leq \mathbf{n, q} \leq 100000$
- $1 \leq \mathbf{A_i, x, y} \leq 100000$

Example

Input:

```
1
5 3
1 2 3 4 5
1 3
3 4
5 1
```

Output:

```
Case 1:
4
5
7
```

Explanation

After the first update,

A ={3, 2, 3, 4, 5}, Value(**A**) = |3-2| + |2-3| + |3-4| + |4-5| = 4

After the second update,

A ={4, 2, 4, 4, 5}, Value(**A**) = |4-2| + |2-4| + |4-4| + |4-5| = 5

After the third update,

A ={4, 2, 4, 4, 1}, Value(**A**) = |4-2| + |2-4| + |4-4| + |4-1| = 7

Problem code: **ZUBSECRT**
Problem name: **I - Secrets**

There are **n** people who call themselves *friends*. However, you can't trust people blindly these days! Not each of these people trusts all the other people. The good thing is, still there are some pairs of people who trust each other and consider each other as *true friend*s. Also, each person has a *best friend*. Note that a *best friend* is also a *true friend*.

These relations are symmetric. Formally, person **A** has several *true friend*s who also consider **A** as a *true friend*. Person **A** has exactly one *best friend*, **B**, and **A** is also the *best friend* of **B**.

Each person has a secret. Those **n** secrets are written on **n** pieces of papers. Those papers are sealed inside **n** envelopes. The name of the person, whose secret is written on the paper inside an envelop, is written on the envelope. These **n** envelopes are distributed among the **n** people. At each step, one person exchanges the envelope (without opening it) he/she is holding with one of his/her *true friend*s. Your job is to determine the minimum number of steps (exchanges) required to end up at an arrangement where secret about each person reaches the hand of that person or the best friend of that person, i.e. finally after these exchanges when the envelopes are opened, the secrets of a particular person is either seen by himself or his best friend.

Input

- The first line of the input is the number of test cases, **T**. Description of each test case is given below.
- The first line of each test case will contain the number of people **n**.
- The second line will contain **n** space-separated numbers $\mathbf{e_1, e_2, \ldots, e_n}$ which is a permutation of **1, 2, ... , n**. Initially, the $\mathbf{i^{th}}$ person will hold the secret for $\mathbf{e_i^{th}}$ person.
- The third line will contain **n** space-separated numbers $\mathbf{b_1, b_2, \ldots, b_n}$ which is a permutation of **1, 2, ... , n**. The $\mathbf{b_i^{th}}$ person is the best friend of the $\mathbf{i^{th}}$ person.
- The fourth line will contain a number **m**.
- Each of next **m** lines will contain two space-separated integers $\mathbf{a_i}$ and $\mathbf{b_i}$ denoting that $\mathbf{a_i}$ and $\mathbf{b_i}$ are *true friend*s. Note that *best friend*s are *true friend*s by definition. They will not be explicitly listed as *true friend*s.
- An empty line will be printed after each test case.

Output

For each test case, print "Case i: ", and then the answer, where i is the testcase number, 1-indexed. The answer should be the required number of steps. If it's not possible to make all the secrets (envelopes) reach in safe hands, the answer should be **"IMPOSSIBLE"** (without quotes)

Constraints

- $1 \leq \mathbf{T} \leq 10$
- $1 \leq \mathbf{n} \leq 10$
- **n** will be even
- $1 \leq \mathbf{m} \leq 10$
- The answer, if it exists, will be less than or equal to 18.

Example

Input:

```
2
4
2 3 4 1
3 4 1 2
1
2 3

4
2 1 4 3
3 4 1 2
0
```

Output:

```
Case 1: 4
Case 2: IMPOSSIBLE
```

Problem code: **ZUBREACH**
Problem name: **J - Reached Safely Or Not**

Everybody is worried about Rakesh as the boy does not have much knowledge about the real world. He can not go from one place to another on his own. It's high time he learned to explore the city. He is going to a relative's house situated on the other side of the city on his own. As this is his first time, he is carrying a GPS tracker of a special kind. The tracker continuously sends information to the family of Rakesh about his movement. The information is sent using the following four letters: **U, D, R,** and **L**. Those letters indicate the moves taken by Rakesh.

The city can be considered as a grid. Rakesh starts his journey from **(0, 0)** position of the grid. His relative's house is situated at **(Rx, Ry)**. Rakesh can move in four directions: up, down, right, or left indicated by **U, D, R,** and **L** respectively.

Any position of the city with *x* ordinate negative or greater than **M** is considered as dangerous. Also, any position of the city with *y* ordinate negative or greater than **N** is considered as dangerous. You will be given the total sequence of Rakesh's movement. You need to determine if Rakesh ended up being at his relative's house, at a dangerous place, or at a random place in the city.

To make things clear,

- **U** indicates a move that increases position along *y*-axis by 1
- **D** indicates a move that decreases position along *y*-axis by 1
- **R** indicates a move that increases position along *x*-axis by 1
- **L** indicates a move that decreases position along *x*-axis by 1

Note that we are interested in the position of Rakesh at the end of his journey only. He may visit some dangerous place or his relative's house at some intermediate point but that won't affect the answer.

Input

- The first line of the input contains an integer **T** denoting the number of test cases. The description of each test case follows.
- The first line of each test case contains two integers **M** and **N**.
- The second line contains two integers **Rx** and **Ry**.
- The third line contains the length of Rakesh's move sequence.
- The next line contains the move sequence containing letters **U, D, R,** and **L** only with no space.

Output

For each test case, print "Case i: ", and then the answer, where i is the testcase number, 1-indexed. The answer should be any of the following three strings:

- **"REACHED"** if Rakesh could reach his relative's house
- **"DANGER"** if Rakesh ended up being in a dangerous place
- **"SOMEWHERE"** if Rakesh ended up being in somewhere safe place in the city other than his relative's place

Don't print any quotation mark. Check the sample output.

Constraints

- $1 \leq \mathbf{T} \leq 10$
- $0 \leq \mathbf{M}, \mathbf{N} \leq 10000$
- $0 \leq \mathbf{Rx} \leq \mathbf{M}$
- $0 \leq \mathbf{Ry} \leq \mathbf{N}$
- $0 \leq$ Sum of the lengths of all sequences ≤ 10000

Example

Input:

```
2
20 20
4 5
13
LLUUUUURRRRRR
10 10
3 4
7
UDUDDRR
```

Output:

```
Case 1: REACHED
Case 2: DANGER
```

Problem code: **RAWPOWER**
Problem name: **K - Raw Power**

The University of Geo Tech, Riga has a new doctoral student on the block who calls himself Raw Power. He has **N** meetings lined up today and wants to avoid them all.

The i^{th} meeting takes place from time units $\mathbf{s_i}$ to $\mathbf{t_i}$. Raw power wants to move meetings around so that the amount of time for which all meetings are simultaneously scheduled is maximized. Once he does this, he will send an email to his advisor citing this number to justify the cancellation.

Of course moving meetings around affects a student's credibility. Specifically moving a meeting from time period [$\mathbf{s_i}$, $\mathbf{t_i}$] to [$\mathbf{s_i}$ + a, $\mathbf{t_i}$ + a] is possible for any *integer* a but reduces credibility by |a| (|x| denotes absolute value of x). Raw Power currently has a credibility of **K** and is willing to sacrifice it all. He needs your help to know the maximum possible intersection of all meeting times that he can achieve whilst keeping his credibility non-negative.

Formalizing intersection: Given a set of meetings each with a start and end time, let the maximum start time over all meetings be s and the minimum end time over all meetings be e. The intersection time of all meetings is 0 if s>=e and e-s otherwise.

Input

- The first line contains a single integer **N**, the total number of meetings scheduled.
- **N** lines follow, the i^{th} of which contains integers $\mathbf{s_i}$ and $\mathbf{t_i}$, the start time and end time of the i^{th} meeting.
- The next line contains a single integer **Q**, denoting the number of queries.
- **Q** lines follow, the i^{th} of which contains an integer **K** denoting the available credibility. Note that all the queries are indepedent.

Output

- Output **Q** lines, the i^{th} of which should contain a single integer, the answer for test case i. This value denotes the maximum attainable intersection of all meetings under the constraint of available credibility.

Constraints

- $1 \leq \mathbf{N} \leq 300000$
- $0 \leq \mathbf{s_i} \leq \mathbf{t_i} \leq 10^9$
- $1 \leq \mathbf{Q} \leq 20$
- $1 \leq \mathbf{K} \leq 10^{18}$

Example

Input:

```
4
5 10
2 12
14 25
7 19
7
100000
4
5
6
7
8
9
```

Output:

```
5
0
1
2
3
4
5
```

Explanation:

If the total credibility is 6, one way to obtain an intersection of 2 is to move the meeting [14, 25] to [8, 19] so that all meetings intersect in the time range [8, 10].