MIST-National Collegiate Programming Contest, 2020

22nd February 2020

You get 16 Pages, 11 Problems & 300 Minutes

Sponsored By:

Problem A

Find the Word

You are given a string **S** of lowercase English letters. Let, **X** and **Y** be two substrings of S such that:

$$\mathbf{X} = S_A S_{A+1} \ldots\ldots S_{B-1} S_B \text{ and } \mathbf{Y} = S_C S_{C+1} \ldots\ldots S_{D-1} S_D$$

Given the values of **A**, **B**, **C** and **D** as input, you have to answer how many times **Y** occurs as a **substring** of **X**.

Input

The first line of the input contains an integer **T** (**1 ≤ T ≤ 5**) denoting the number of test cases.
Each test case is described as follows :

- The first line contains **S (1 ≤ |S| ≤ 10^5)** - the given string.
- The second line contains **Q (1 ≤ Q ≤ 10^5)** - the number of queries for this string.
- Next **Q** lines each describes a query **A B C D (1 ≤ A ≤ B ≤ |S|, 1 ≤ C ≤ D ≤ |S|)**

Output

For each query, output the number of times **Y** occurs as a **substring** in **X**.

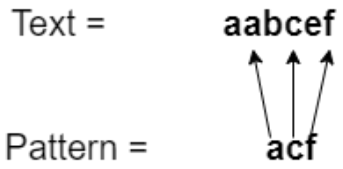
Sample Input	Output for Sample Input
3 ababab 3 2 4 4 5 3 6 1 4 5 6 2 3 abcac 2 1 3 4 5 1 3 5 5 aaaaa 1 1 5 1 2	1 1 0 0 1 4

Problem B

Search The Files

Kaishya was fed up with all the IDEs he uses. So, he wanted to build a great IDE which he calls **AIDE (Artificially Intelligent Development Environment)**. He started coding to build this legendary **AIDE**. But at one stage, he stopped. He was trying to implement a feature to search files from the root of the project folder. But he was not sure which algorithm to implement for that. Kaishya is familiar with a lot of string matching algorithms, but the disadvantage of those is - he will need an exact keyword to find out a file. So what he needs is an algorithm that will do the approximate matching.

After Googling for a few hours, Kaishya came to know about a new term called "**Fuzzy Search**". This algorithm tries to find if the search **pattern** is a subsequence of the main **text**. For example, **acf** is a subsequence of **aabcef** but **acb** or **aba** are not.



Though Kaishya learned this algorithm, he doesn't have much time to implement this as he will invest his precious time in developing other features of AIDE. From somewhere he got information about you guys. So, he is asking for your help. You will be given **N** file paths and a search **pattern**. A file path is a string representing where the associated file is located. Each file path is assigned an unique id between **1** to **N**. You need to find all the file paths which will contain the **pattern** according to the fuzzy search algorithm.

Input

The first line of the input will contain a single integer **T (1 ≤ T ≤ 5)** which denotes the number of test cases. Each test case starts with a line containing a single integer **N (1 ≤ N ≤ 10^5)**. Each of the next **N** lines will contain one **file path (1 ≤ |File path| ≤ 10^5)** as a string. Id of the i^{th} file path is i. After that, there will be one line that contains the **pattern (1 ≤ |pattern| ≤ 10^5)**. The file paths and the pattern will only consist of lowercase letters and one special character "*/*" (ASCII 47). You should assume that the sum of length of all the strings over all test cases in a single input file will never cross the limit **10^6**.

Output

For each test case, the first line of the output should be the case number in the format "**Case X:**" (without quotes) where **X** is the case number. The next line of the output should contain the ids of the files in increasing order, separated by a single space that contains the pattern according to the fuzzy search algorithm. Print "**No files found!**" (without quotes) if you can't find any such file path.

Sample Input	Output for Sample Input
1 3 /root/documents/a /root/downloads/b /tmp/downloadedfile rs	Case 1: 1 2

Problem C

Bracket Colouring

Do you know what Araspa loves more than a bracket sequence? **A Balanced Bracket Sequence**. Today she is constructing them. She will make a balanced Bracket sequence of **2 * N** length. To make it more interesting she will use different types of brackets. She has an **infinite** number of brackets of **infinitely many** types.

A balanced bracket sequence is a string consisting of only brackets defined formally below:

- The empty string is a balanced bracket sequence.
- if **S** is a balanced bracket sequence, then so is "Opening Bracket of any **i^{th}** type" + **S** + "Closing bracket of **i^{th}** type".
- if **S** and **T** are balanced bracket sequences, then so is **S + T**.

Here, **'+'** means concatenation of string.

Now she is asking you to find the distinct number of balanced bracket sequences of **2 * N** length you can make**.** It should be infinity right?! How can you calculate infinity**?**

To make your life easier she will only permit you to use **X** types of brackets of her own choosing and you will have to use all of them **at least once** in every distinct sequence you make i.e if she allows you to use only these 3 types of brackets: **()**, **{}** and **[]**; then you can make "{}([])", but you can't make "({()})" because you didn't use 3rd type of bracket here.

But to make your life a little bit easy-medium she will not ask you to do this only for a single **X.** Rather she will give you **K** and ask you to find ***DBS()*** value for all the integers from **1 to K**. The definition of ***DBS()*** function is given below:

DBS(X) = Number of distinct balanced bracket sequences you can make using all of the **X** types of brackets at least once**.** As the number of ways can be very large so we will store them using modulo **998244353**.

Now Araspa knows that printing an array is no joke so you will have to print **Func()** which is defined below:

```
Func() :
      Ans = 0
      For(X = 1 to K)
            Ans = Ans ^ DBS(X)
      Return Ans
```

Here **'^'** means Bitwise XOR operation

See the Explanation for more clarification.

Input

In the first line, there will be a single integer **T (1 ≤ T ≤ 20)**, denoting the number of test cases. Each of the next **T** lines will contain two space separated integers: **N** and **K (1 ≤ K ≤ N ≤ 10^5)**

Output

For each test case, print the case number and then print a single integer answer to that test case as described in the statement. Please note that the final result may be greater than **998244353.**

Sample Input	Output for Sample Input
6 2 2 99190 98831 100000 10 100000 100 100000 10000 100000 100000	Case 1: 6 Case 2: 107518997 Case 3: 412801477 Case 4: 442870344 Case 5: 531237973 Case 6: 168710547

Explanation :

In the first test case**,**
For **X** = 1, you can make the following 2 types of brackets :
()()
(())

For **X** = 2, you can make the following 4 types of brackets :
({})
{()}
(){}
{}()
So answer = 2 ^ 4 = 6

Problem D

Water in Tree

You are given an undirected tree of **N** nodes, where nodes are labeled from **1** to **N**, with the root in node **1**. Each node of this tree can store exactly one unit of water. Now, you need to process **Q** queries on this tree. There can be two types of queries. Here are the definitions:

Type 1: *1 U X* means - pour **1** unit of water on node **U** for **X** times. When a node receives **1** unit of water, it will try to pour the water down to its first empty child. If every child is storing water, then the node will store the water itself. If the node itself isn't empty, the water overflows. Note that, this process is true for every node that receives any water. No other water is poured until one unit of water is finished propagating down. On the tree below, fig-1 and fig-2 reflects the effects of query 1 1 3 and then query 1 2 12. Note that the shaded nodes are storing water.

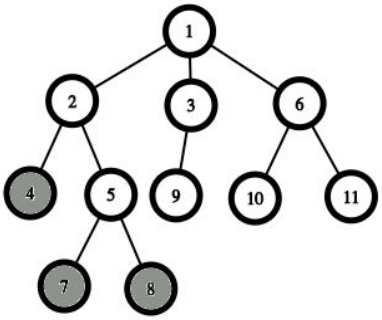


fig-1

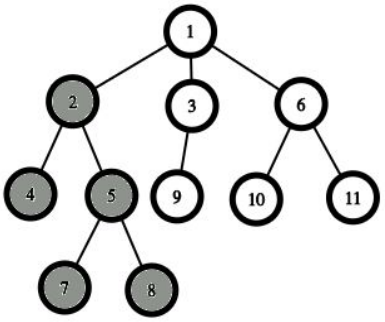


fig-2

Type 2: *2 U* means - you have to answer if node **U** has water in it or not at this particular moment.

Note that, the order of children of each node U is determined by the ascending order of their node label.

Input

The input contains **T (1 ≤ T ≤ 100)**, the number of test cases. The first line of each case contains a single integer **N (1 ≤ N ≤ 10^5)**, number of nodes in this tree. Each of the next **N - 1** lines contains two integers **U** and **V (1 ≤ U, V ≤ N)** - which means there is an edge between node **U** and node **V**. It is guaranteed that the given input is a tree. Next line contains a single integer **Q (1 ≤ Q ≤ 10^5)**, number of queries. Each line of the next **Q** lines contains either **1 U X** (if type 1) or **2 U** (if type 2), where **1 ≤ U ≤ N** and **1 ≤ X ≤ 10^8**. It is guaranteed that sum of **N** and sum of **Q** is at most **400,000** separately, in each input file.

Output

For each test case, the first line should print the test case number in the format showing in the sample output. Then for each query of type 2, you have to print **1** (if node **U** has water) or **0** (if it doesn't).

Sample Input	Output for Sample Input
1 11 1 2 1 3 1 8 2 4 2 5 5 6 5 7 3 11 8 9 8 10 8 1 1 1 1 2 2 2 5 1 1 1 2 5 1 1 4 2 3 2 10	Case 1: 0 1 1 0

Problem E

K-Divisors

The ***positive divisor function*** is defined as a function that counts the number of positive divisors of an integer **N**, including **1** and **N**.

If we define the *positive divisor function* as **D(N)**, then, for example:
D(24) = 8 (Because **24** has **8** divisors and they are **1, 2, 3, 4, 6, 8, 12, 24**)

Calculating **D(N)** is a classical problem and there are many efficient algorithms for that. But what if you are asked to find something different? Given a range and an integer **K**, can you find out for how many **N** in the given range, **D(N)** equals **K**?

Input

In the very first line, you'll have an integer called **T**. This is the number of test cases that shall follow. Every test case contains three integers, **L**, **R**, and **K**. **L** and **R** represent the range and are inclusive.

Constraints

- **$1 \leq T < 31$**
- **$1 \leq L \leq R < 2^{31}$**
- **$1 \leq K < 2^{31}$**

Output

For every test case, you must print the case number, followed by the count of numbers with exactly **K** divisors in the range.

Sample Input	Output for Sample Input
3 10 10 4 2 13 2 100 10000 100	Case 1: 1 Case 2: 6 Case 3: 0

Problem F

Jumping Frog Redefined

There was a river beside a road in Babylon. The river is divided into **M+1** blocks numbered from **0** to **M.** Some of these blocks in the river contain one stone each. At the further end of the river, there lives a frog. In order to search for food, the frog crosses the river by jumping on blocks. The frog can only perform jumps on blocks that have a stone.

The frog lives on the **0th** block, and it wants to reach the **Mth** block jumping through these blocks containing stones. Both the **0th** and **Mth** block will always contain stones. If the frog jumps from block **x** to block **y,** then he has to perform **|x - y|** unit jump. The frog gets tired if he performs a long jump. So he prefers small jumps, rather than long jumps.

Among all the **M+1** blocks, there are only **N+2** (including **0th** and **Mth** block) blocks that contain stones. And the frog memorized those block numbers. So the frog decided to put some extra stones in some of the blocks of the river to minimize his maximum jump length. Now, the frog is curious to know his lowest possible maximum jump length, if he puts **at most S** extra stones in some of the blocks of the river in **Q** different queries. Can you help him?

Input

The First line contains an integer **T(1 ≤ T ≤ 14)** denoting the number of test cases. Each test case starts with two space separated integers **N, M (1 ≤ N ≤ 10^5), (N < M ≤ 10^6)**. The next line contains **N** space separated integers A_1 A_2 A_3 ... A_N **(1 ≤ A_i < M)** denoting the blocks having a stone. All the block numbers are unique. The following line contains an integer **Q (1 ≤ Q ≤ 2 * 10^5)** denoting the number of queries. The next line contains **Q** space separated integers **S (0 ≤ S ≤ M)**, denoting the maximum number of extra stones. It is guaranteed that the sum of **N** and **Q** of over all test cases will be less than **3 * 10^6**.

Output

For each test case print the case number, and then for each query print the lowest possible maximum jump length. Please see the sample for details.

Sample Input	Output for Sample Input
1 3 30 5 10 29 3 1 2 30	Case 1: 10 7 1

Warning: Dataset is huge. Please use faster I/O methods.

Problem G

Age of Perceptrons

For research purposes, you are going to collect cancer data from the National Institute of Cancer Research & Hospital (NICRH) for **N** days. Each day they will provide you with a 2D data point that represents a tumor location in an X-Ray and its label - malignant or benign. Data will be given in the format **(X, Y, L)** where **(X, Y)** is the data point and **L** is its label. If **L = 0,** then it's malignant cancer and if **L = 1,** then it's benign. For each day, you want to figure out if the malignant and benign data points up to that day are **linearly separable**.

Two sets of 2D points are **linearly separable** if there exists a straight line so that no points are on the line and no pair of points from different sets are on the same side of the line. And also, if one of the sets is empty, then the sets are linearly separable.

Input

The first line of the input contains an integer **T** (**1 ≤ T ≤ 25**), which denotes the number of test cases. The first line of each test case contains a single integer **N** (**$1 \le N \le 10^5$**), the number of days. **N** lines will follow, i^{th} of which will contain three integers X_i, Y_i and L_i (**$-10^9 \le X_i, Y_i \le 10^9$, $0 \le L_i \le 1$**) - the data point and label for i^{th} day. It is guaranteed that all the points in a test case are distinct. It is guaranteed that the sum of **N** of over all test cases will be less than or equal **$3 * 10^5$**.

Output

For each test case, the first line should contain the case number in the format "**Case X:**" (without the quotes) where **X** is the case number. After that, for each day, print a single line containing "**YES**" or "**NO**". Print "**YES**" if the dataset up to that day is linearly separable or "**NO**" otherwise. Please see the sample for details.

Sample Input	Output for Sample Input
2 2 0 0 0 0 1 1 4 0 0 1 1 1 1 1 0 0 0 1 0	Case 1: YES YES Case 2: YES YES YES NO

Problem H

The Nostalgia of a Deja Vu

In this problem, all you have to do is generate an array, having **N non-negative** integers, which satisfies the given **Q** constraints. Every constraint is of the following kind - "The bitwise **XOR** of all the numbers from the **L**-th position to the **R**-th position of the array has to be equal to **X**". You need to find the lexicographically smallest array which satisfies all the constraints or claim that such an array does not exist.

Input

The first line of the input contains an integer **T**, denoting the number of test cases. Then the description of the **T** test cases follows. The first line of every test case will contain two integers **N** and **Q**, denoting the length of the array and the number of constraints respectively. Each of the following **Q** lines will contain a constraint of the following form: **L R X**.

Constraints

- **$1 \le T \le 15$**
- **$1 \le N, Q \le 5 * 10^4$**
- **$1 \le L \le R \le N$**
- **$1 \le X \le 2 * 10^9$**

Output

For each test case, print a line containing the case number and if such an array can be constructed then print the lexicographically smallest array which satisfies all the given constraints. Otherwise, print "**Impossible**".

Sample Input	Output for Sample Input
1 7 2 2 5 2 6 7 3	Case 1: 0 0 0 0 2 0 3

An array **C** of size **N** is called lexicographically smaller than another array **D** of equal size, if and only if there exists an **i ≤ N** such that:

C[p] == D[p] for all **1 ≤ p < i** and **C[i] < D[i]**

Warning: Dataset is huge. Please use faster I/O methods.

Problem I

Save Mozzarella!

Joker is back again! He has planned to destroy the country Mozzarella.
Mozzarella is a country of **N** cities and **N** roads. Initially, people can go to any city from any city via one or multiple roads. All roads are bi-directional. There might be multiple roads between the same pair of cities. Even both endpoints of a road can be the same.
We will say that Mozarella is **disconnected** if there exists a pair of cities, **u** and **v** such that **u is not reachable from the city v.**
To destroy Mozzarella, **Joker's ultimate goal is to make Mozzarella disconnected**. So he picks **K roads randomly** and puts bombs on those roads.

Batman, as usual, got the news. But he only knows that exactly **K roads** have been selected but doesn't know which roads got selected. And he has no time to check every road, so he also picks **K roads randomly**. For each of the **K** roads Batman has selected, he checks if this road contains any bomb. If it has any, then he disposes of the bomb.

A road will be destroyed if Joker puts a bomb in it and Batman doesn't dispose of it.

You have to find the **probability of Joker's success, S**. In other words, the probability that Mozarella got disconnected. It can be shown that **S** can be represented by **P/Q** where **P** and **Q** are co-prime integers and **$Q \neq 0 \bmod (10^9+7)$**. Find the value of **$(PQ^{-1}) \bmod (10^9+7)$**.

Input

The first line of the input contains a single integer **T ($1 \leq T \leq 200$)** denoting the number of test cases. The description of **T** test cases follows.
The first line of each test case contains two integers **N** and **K ($1 \leq K \leq N \leq 100000$)**. Each of the next **N** lines contains two space-separated integers **x ($1 \leq x \leq N$)** and **y ($1 \leq y \leq N$)** denote that cities **x** and **y** are connected by a road. Note that, the summation of all **N** in a single test file will be at most **1000000.**

Output

For each test case, print a single line containing the **probability of Joker's success, S**. For more details, please see the sample I/O.

Sample Input	Output for Sample Input
2 3 3 1 2 2 3 1 3 4 1 1 2 2 3 1 3 3 4	0 687500005

Problem J

Evil Judges

Not Competitive Programming Contest (NCPC) 2020 is going on. But this time it's going to be held in a brand new format. The format is given below:

- Scoring for a problem will be dynamic. The score for a problem will be an integer and can range from **1** to **1000000000**. The function for dynamic scoring is unknown to the contestants.
- There is no time penalty for wrong submissions.
- The time penalty will be the exact time of the latest accepted solution.
- Contestants will be sorted according to their total score, a larger score will come first. If their total scores match, then they will be sorted according to their time penalty, a smaller time penalty will come first. If they also match, they will share the position. Technically, If **X** has a larger score than **Y** or they both have the same score but **X** has a smaller time penalty, then **X** performed better than **Y**. So, the rank of a contestant is **1 + [number of contestants having better performance]**.

Rank	Name	Total Score	Time Penalty	A(3)	B(9)	C(10)	D(10)
1	ptr	23	120	120	-	23	57
2	trst	22	111	7	18	-	111
3	rng	20	107	-	-	43	107
3	bmry	20	107	-	-	89	107
3	mnvmr	20	107	-	-	107	73
6	rdsh	19	107	-	80	107	-
7	bnq	19	315	-	315	-	212

rng, **bmry**, **mnvmr** share the same rank **3** because they have the same score and time penalty thus the exact same performance. **rdsh** gets rank **6** because he has **5** contestants having better performance than his.

The contest is over. Each contestant solved some problems (probably zero, probably all) having some time penalty. The standings are not revealed yet thus no one knows the scores of the problems.

But you know the judges are evil. If a contestant bribes enough to them, they may manipulate the score for each problem in such a way that the rank of that contestant would be best possible.

In the standings above, the scoring for each problem gives the best possible position for **ptr** which is the championship. But if the judges give **11** points instead of **9** for problem **B**, then **trst** would get **24** points and will become champion in that contest. In this scenario, **rdsh** and **bnq** will also have **21** points instead of **19** and will become **3rd** and **4th** respectively. Remember, if the judges set the score for a particular problem, it's the same for all the contestants.

So, judges can set scores for the problems according to their wish. Thus different scoring distribution will generate different standings. Let's call the best possible rank for a particular contestant, **evil-rank**. As the

contestants don't know the function for dynamic scoring, judges can set the score for each problem at their wish so that the rank of the contestant is best possible. So mathematically, if there are **M** problems, then a total of $\mathbf{1000000000^M}$ different scoring distribution is possible. For a particular contestant, among all these standings, the best rank for that contestant is the evil-rank for him.

You are given the list of solved problems and the time penalty for each contestant. You have to find the evil-rank of each contestant if he bribes the evil judges individually. This means when you are calculating the evil-rank for a particular contestant, only he bribes the judges and other contestants do not.

Input

Input starts with an integer, **T**, denoting the number of test cases. For each test case, the next line contains two integers **N** and **M,** denoting the number of contestants and the number of problems. The next **N** lines contain the details for each contestant. Each line will be in this format:

[type][problems][space][penalty]

[type] will be either '**!**' or '**?**' (without the single quotations). '**!**' denotes that the contestant has solved the following problems and '**?**' denotes that the contestant has solved all the problems except the following problems.
[problems] will contain a string containing only the first **M** upper-case Latin letters. Each character of the string will be unique and this can be empty.
[space] denotes simply space character ' ' (**ASCII 32**).
[penalty] will be an integer between **0** to **1000000**.

Check the sample input for more clarification.

Output

For each test, print the case number and space-separated evil-rank of each contestant in the given order. Check the sample output for more clarification.

Constraints

1 ≤ T ≤ 50
1 ≤ N ≤ 100000
1 ≤ M ≤ 20
1 ≤ Sum of N over all test cases ≤ 1000000
1 ≤ Sum of M over all test cases ≤ 200
0 ≤ Time Penalty ≤ 1000000

Sample Input	Output for Sample Input
2 4 2 !A 49 ?A 49 !B 99 ?B 99 7 4 !ACD 120 !ABD 111 !CD 107 ?AB 107 !CD 107 !BC 107 ?AC 315	Case 1: 1 1 2 2 Case 2: 1 1 2 2 2 1 2

Problem K

Respectfully Giving Away

These days, it's very hard to get respect. People only respect a few people. Like, among the contestants, people only respect the ICPC World Finalists. This behavior is so infectious that some of the companies are giving different salaries to World Finalists. This often becomes the reason for heartache for many who took part in contests for so long but couldn't manage to get into the World Finals.

You as a contestant, want to analyze this behavior. Given a company's salary for its newly recruited **N** employees, you want to figure out if they are biased towards World Finalists. You know among the **N** newly recruited employees exactly one person **P** is an Ex-World-Finalist. And every company will give **P** the highest amount of salary. If there is at least one other newly recruited employee whose salary is equal to **P**'s salary, then the company is not biased, otherwise, it is.

Input

The first line will contain **T (1 < T < 10)**, the number of test cases. Each case will start with **N (1 < N < 101),** the number of newly recruited employees for that company. A line with **N** integers will follow. Each of these integers will be between **50** to **100** which represents the salary of each employee in thousands.

Output

For each case print one line with either **"Yes"** if the company is biased, or **"No"** if it is not.

Sample Input	Output for Sample Input
2 3 60 60 75 3 60 75 75	Yes No