RUET CSE FEST 2k22
Inter University Programming Contest

Technical Partner:

04 June, 2022
You get 15 Pages, 11 Problems & 300 Minutes

RUET
CSE FEST 2k22

Problem A

Maiden Over

Bangladesh Association of Problem Setters

We all know about cricket, a game which is played between two teams. Each team bowls and bats alternatively. The team that scores more runs, is declared the winner. The aim of the batsmen is to score runs while the aim of bowlers is to minimize the run of the opposite team.

The person who bowls to the batters is known as a bowler. A bowler bowls six balls consecutively to finish an over. After finishing an over another bowler starts to bowl from the other end of the pitch. If a bowler doesn't give any run in a particular over, then, that over is considered a maiden over.

Mr. X is one of the deadliest bowlers nowadays. He doesn't get wickets because batsmen are very aware of his bowl and play carefully. In this problem, you are given how many runs were scored in each bowl of Mr. X. You have to determine how many maiden overs were bowled.

You may safely assume that Mr. X does not bowl any no ball or wide ball and for batters, there are no other ways to score runs without hitting a ball.

Input

The first line will contain **T (1 ≤ T ≤ 1000)**, the number of test cases. The following **T** test cases will start with an integer **O (1 ≤ O ≤ 10),** denoting the number of overs bowled by Mr. X. Each of the next **O** lines will consist of **6** integers describing the runs scored in each ball.

Output

For each case, print one line with **"Case <x>: <y>"**, where **x** is the case number and **y** is the number of maiden overs bowled. Please see the sample output for more details.

Sample Input	Output for Sample Input
2 1 0 0 0 0 0 0 1 1 0 0 0 0 0	Case 1: 1 Case 2: 0

Problem B

All About Constraints

In this problem, all you have to do is generate an array, having **N positive** 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 the **SUM** of all the numbers from the **L**-th position to the **R**-th position". You can only use numbers from 2^0 to 2^{63} 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** $(1 \le T \le 10)$, 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**, $(1 \le N, Q \le 10^5)$ 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** $(1 \le L \le R \le N)$.

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**". Please make sure two consecutive elements in the array are separated by a single space. And there are no extra spaces after the last element. Please see the sample output for more details.

Sample Input	Output for Sample Input
2 6 2 2 3 6 6 100000 1 1 100000	Case 1: 1 1 2 1 1 1 Case 2: Impossible

Note: An array **C** of size **N** is called lexicographically smaller than another array **D** of equal size, if and only if there exists an $i \le N$ such that: **C[p] == D[p]** for all $1 \le p < i$ and **C[i] < D[i]**

Problem C

Some Game

Alfonso and Bethany are playing a game. In this game, there are **P** boxes numbered from **1** to **P**. The i-th box has **A[i]** balls initially. All of the balls in the game are distinct. The boxes are similar, i.e. there is no way to distinguish one box from the other.

At first, the players are given a target integer **T** (the significance of **T** is explained towards the end). The game consists of **P** sequential moves. Alfonso gives the first move and then Bethany gives the 2nd move, and so on. So if **P** is odd, Alfonso gets one extra move, but Bethany is okay with it.

In the i-th move of the game, a player simply decides if they want to destroy the i-th box or not. After **P** moves, **K** boxes will remain where **K** is the number of moves where a player had decided not to destroy a box. Let **N** be the total number of balls in these **K** boxes. Then the players take out these **N** balls from these **K** boxes and calculate the following value for **R**:

> ***R*** = *(The number of ways to put **N distinct** balls in a linearly ordered way in **K** **indistinguishable** boxes so that none of the boxes remain empty) mod* 10^9+7

See the notes section for an example of how **R** is calculated.

Alfonso's target is to give moves in such a way so that ***abs(T - R)*** is **maximized**. Bethany's target is to **minimize *abs(T - R)*.** *abs(x)* denotes the absolute value of x.

Note that we care about the value of **R** after doing the mod operations.

If both play optimally, what will be the value of ***abs(T - R)***?

Input

The first line will contain a single integer **Q**, denoting the number of test cases. The first line of each test case will have two integers **P**, denoting the number of boxes, and **T**, the target integer. The next line will contain **P** integers where the i-th integer denotes **A[i],** the number of balls in the i-th box.

Constraints

- **$1 \le Q \le 5$**
- **$1 \le P \le 30$**
- **$1 \le A[i] \le 10^6$**
- **$1 \le T \le 10^9$**

T and all A[i] are generated using a pseudo-random generator. The generator is used to make the inputs random and ensure inputs are not adversarially generated against any particular solution.

Output

Print the case number in a single line followed by the value of **abs(T - R)**. Please see the sample for details.

Sample Input	Output for Sample Input
3 1 100 10 2 19490 1 2 5 123456789 5555 1000000 1 342 1653	Case 1: 3628700 Case 2: 19488 Case 3: 687865730

Notes

Suppose, we have 3 balls A, B, and C and 2 boxes.

- {A} {BC} and {A} {CB} are two different ways. (As the linear order of balls inside a box matters)
- But {BC} {A} and {A} {BC} are the same ways (As the boxes are similar)
- {} {ABC} is not a valid way since no box can be empty.

For N = 3, K = 2, the value of R is (6 % (10^9+7)) = 6

.

Explanation of sample 1:

In this game, Alfonso has one move and Bethany has 0 moves.

- Alfonso can destroy the first box making R = 1 and getting ans = abs(100 - 1) = 99.
- Alfonso can decide to keep the first box making R = (10! % (10^9+7)) = 3628800 and getting answer = abs(100 - 3628800) = 3628700. Since Alfonso wants to maximize the final answer, he will decide to keep the box.

CSE FEST 2k22

Problem D

Maxxxxxximum Spanning Tree

You are given a complete graph of size **n** where the nodes are labeled from 1 to **n**. The weight of the edge between any two nodes **i** and node **j** is equal to the gcd(**i**, **j**). Find the cost of the **maximum** spanning tree for this graph.

Input

The first line contains an integer **T**, denoting the number of test cases. Then **T** test cases follow. Each test case contains a single integer **n** in a separate line.

Constraints

- $1 \leq T \leq 10$
- $1 \leq n \leq 1000000$

Output

For each test, print one line in the format "**Case X: Y**", where **X** is the case number and **Y** is the cost of the maximum spanning tree for that case.

Sample Input	Output for Sample Input
2 **1** **4**	**Case 1: 0** **Case 2: 4**

RUET CSE FEST 2k22

Problem E

Number of Zeros

Bangladesh Association of Problem Setters

Hermione is once again trying to solve the following problem in arithmancy class:

Let us consider an array of **N** elements $V = [v_1, v_2, ..., v_n]$. Now we do the following operations repeatedly on V till there is only one element in **V**:

1. $W = [v_1 \times v_2 , v_2 \times v_3, ..., v_{n-1} \times v_n]$
2. $V = W$

Now if initially $v_k = k!$ and then we continue this operation till there is only one element in **V**, what would be the number of trailing zeros in that number in base **B**?

As a friend of Hermione, your task is to solve this problem by writing a program.

Input

The first line will contain a single integer **T (1 ≤ T ≤ 200),** the number of test cases. Then T cases follow. Each test case will have two integers **N (1 ≤ N ≤ 10^5)** and **B (2 ≤ B < 10^5)** in a separate line. **B** will always be a prime number.

Output

For each case, print the case number in a single line followed by the number of zeros. Please see the sample for details. As the number of trailing zeros can be too large, print the **number modulo 1,000,000,007 (10^9 + 7)**.

Sample Input	Output for Sample Input
3 5 5 3 3 4 5	Case 1: 1 Case 2: 1 Case 3: 0

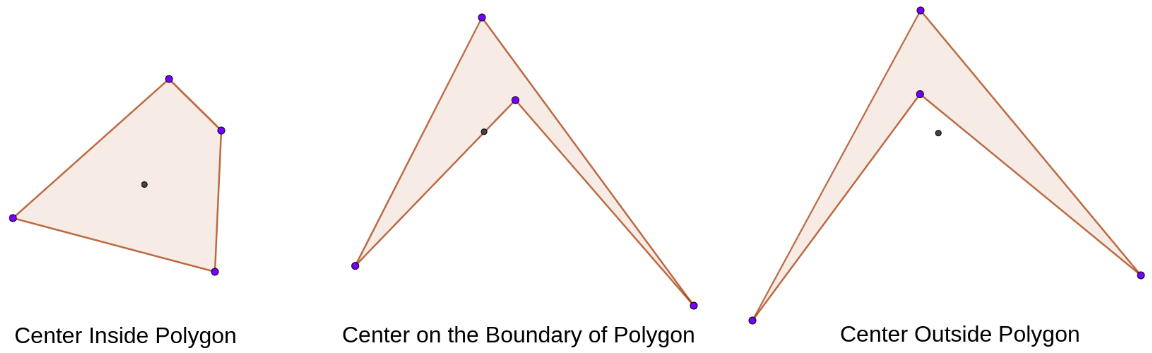
Explanation of Sample Input 2: Given **N = 3, V = [1, 2, 6]**. Then after 1st iteration, **V = [2, 12]** and after 2nd iteration, **V = [24]**. If we write it in base **3**, it is **220,** where there is 1 trailing 0.

Problem F

Unbalanced Polygon

The center of a polygon is the average of its vertices. Formally, the center of the polygon with **n** vertices $(x_1, y_1), (x_2, y_2), (x_3, y_3) \ldots\ldots (x_n, y_n)$ is $\left(\frac{1}{n}\sum_{i=1}^{n} x_i, \frac{1}{n}\sum_{i=1}^{n} y_i\right)$

A polygon is called unbalanced if it is simple and the center of the polygon lies strictly outside the polygon.



Center Inside Polygon — Balanced | Center on the Boundary of Polygon — Balanced | Center Outside Polygon — Unbalanced

Alice has a convex polygon with **n** vertices, $P = P_1P_2P_3 \cdots P_n$ For each vertex $\mathbf{P_i}$, Alice tries to move the point $\mathbf{P_i}$ to a new point **S** anywhere **inside** the polygon **P**, such that the resulting polygon is unbalanced. For each vertex $\mathbf{P_i}$, help Alice find out the area all such points cover.

Formally, Let $T(i)$ be the set of all points **S** inside **P**, such that $P = P_1P_2 \cdots P_{i-1}\, S\, P_{i+1} \cdots P_n$ is an unbalanced Polygon. Find $Area(T(i))$ for all $1 \le i \le n$. It can be proven that the area is finite.

Input

The first line contains an integer **T** $(1 \le T \le 50)$ denoting the number of test cases.

The first line of each case contains an integer **n** $(3 \le n \le 5000)$ the number of vertices of the polygon **P**.

Each of the next **n** lines contains two integers x_i and y_i $(-10^5 \le x_i, y_i \le 10^5)$ - the coordinates of $\mathbf{P_i}$.

The vertices of the polygon are given in counter-clockwise order. It is guaranteed that the vertices form a convex polygon.

Output

For each test case, output **n** space-separated real numbers. The **i**-th number should be **Area(T(i))**, the answer for the **i**-th point. Your answer will be considered correct if its relative or absolute error doesn't exceed 10^{-4}.

Sample Input	Output for Sample Input
2 3 0 0 1 1 1 2 4 0 0 1 0 1 1 0 1	0.0 0.0 0.0 0.166667 0.166667 0.166667 0.166667

Problem G

Polymorphism

Polymorphism is one of the four main pillars of Object Oriented Programming. It means "many forms", and it occurs when we have many classes that are related to each other by inheritance. **Inheritance** lets us inherit attributes and methods from another class.

In this problem, we will only focus on creating classes and objects by following these properties. We are designing a system that will support two types of operation, either we will add a new class in the system, or ask whether we can create an object at this state or not. Details are described below.

Creating classes:
We already have an existing class in our system called **Main**. All the other classes will be inherited from one of the already declared classes. Class names are case-sensitive and unique. A plus sign ("+") will be used while creating a new class. It will follow this format

```
+ class SubClassName extends SuperClassName
```

Here, **`SubClassName`** is the newly created class and **`SuperClassName`** is the parent class it inherits from. And **`extends`** is the keyword that denotes the inheritance. That Superclass has to be created before declaring this new Subclass. We assure you that all the classes in the input are created in a valid way.

Creating objects:
We can create objects from the existing classes in two ways. This will contain a question ('?') sign in the beginning

```
a)    ? ClassName objectName = new ClassName()
b)    ? AncestorClassName objectName = new ClassName()
```

To create an object, specify the class name, followed by the object name, and use the keyword **`new`**. In the second method, there has to be a chain of inheritance from **`AncestorClassName`** to **`ClassName`**. Object names are case-sensitive and unique as well.

Here's an example for better understanding. Suppose we have created these classes.

```
+ class Animal extends Main
+ class Dog extends Animal
+ class Cat extends Animal
+ class Puppy extends Dog
```

Now some valid object creations are:

```
Animal animal = new Animal()
Main abcd = new Animal().
```

`Main xyz = new Puppy()` - This is valid because *Main* is an ancestor class of *Puppy* class.
`Animal catAnimal = new Cat()`

Some invalid object creations are:
`Animal object = new Main()` - This is invalid, because *Animal* is not an ancestor class of *Main*.
`Main myObject = new MyClass()` - This is invalid because we don't have any class named *MyClass*.

Now in this problem, you have to determine whether each object creation is valid or not.

Input

Input starts with an integer **T,** number of test cases. The first line of each test case starts with integer **N**, which indicates how many lines to follow. Then each of the next **N** lines starts with either a plus sign (+) which means a new class is created or a question sign (?) which asks a query whether this object creation is valid or not.

Constraints

- **1 ≤ T ≤ 100**
- **1 ≤ N ≤ 100000**
- **1 ≤ Length of each class name ≤ 10**
- **1 ≤ Length of each object name ≤ 10**

Summation of all **N** in a test file will be less than or equal to **200000**.
Class and object names consist of lower-case, upper-case characters, and numeric digits.
All names and keywords are case-sensitive and unique.

Output

Print the case number in a single line. Then for every query, you need to print "yes" only if it is a valid way of creating an object or print "no" otherwise. Please see the sample for details.

Sample Input	Output for Sample Input
1 7 + class Abc extends Main ? Abc obj = new Abc() ? Main obj2 = new Abc() ? Abc obj3 = new Main() + class Xyz extends Abc ? Main obj4 = new Xyz() ? Abc obj5 = new Xyz()	Case 1: yes yes no yes yes

Note: Dataset is huge. Please use faster input/output methods.

RUET
CSE FEST 2k22

Problem H

Digit Shifts

Bangladesh Association of Problem Setters

You will be given a **non-negative integer X**. You need to perform **Q** queries on that integer. Each query will consist of a single decimal digit **D**. After every query, you need to move all occurrences of digit **D** in the integer to the end, while keeping the relative position of every other digits intact.

For example, suppose **X = 123123,** and suppose **Q = 3**.

1. For the first query, **D = 1**, then after the digits are shifted **X = 232311**.
2. For the second query, **D = 2**, then after the digits are shifted **X = 331122**.
3. For the third query, **D = 3**, then after the digits are shifted **X = 112233**.

After every query, you need to output the value of the integer **X**. Since it can be really large, output it modulo **1000000007** (**10^9 + 7)**.

Please note that if at any point after a query, **X** contains leading zeros, then the leading zeros should be **discarded**. Therefore, if **X = 2022** and if **D = 2**, then after the query, **X** will become **222**.

Input

The first line will contain a single integer **T (1 ≤ T ≤ 20)**. Each starts with a single line, which will contain the **integer X**. Then, in the next line there is a single integer **Q (1 ≤ Q ≤ 10^5)**, denoting the number of queries. Each of the next **Q** lines denotes a query, containining a decimal digit **D** **(0 ≤ D ≤ 9)**. You can safely assume that **X** won't contain leading zeros initially and that **X** will never have more than **10^5** digits.

Output

For each test case, first print the case number in a single line like "**Case V:**". Then for each query, output the value of **X** modulo **1000000007**. Refer to the sample I/O for more clarity.

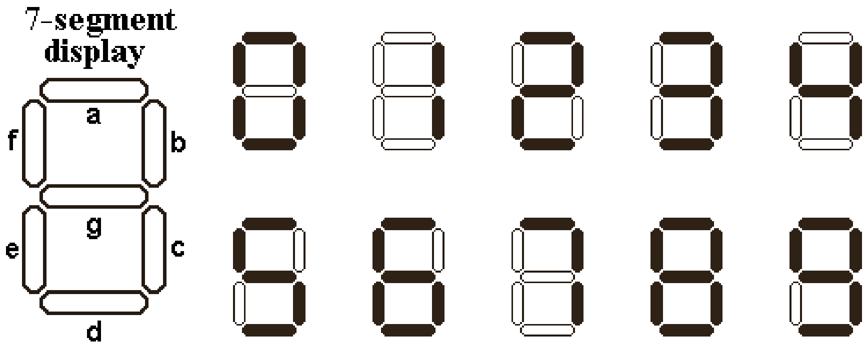
Sample Input	Output for Sample Input
1 123123 3 1 2 3	Case 1: 232311 331122 112233

Note: Dataset is huge. Please use faster input/output methods.

Problem I

Seven Segment Display

You are given an integer. The integer is written using sticks in the same format as 7 segment display. The figure below shows the digits (0 - 9) in seven segment display format.



In the given integer, you can't add or remove any sticks. But you can move some (possibly none) sticks. In a single move, you can take one stick from one position and put it to a different position which is empty. The move can happen from one digit to another digit. It can also happen between the same digit as well. The length of the starting number and the final number must be the same. What is the maximum integer you can make by using at most **K** moves?

Input

The first line will contain a single integer **T (1 ≤ T ≤ 20)**. Each test case will have a two integers **S, Q (1 ≤ Length of S, Q ≤ 200)**, where **S** is the given integer in seven segment display and **Q** is the number of queries. **Q** lines will follow. **S** will be a non-negative integer, with possible leading zeroes. Each line will contain an integer **K** (**1 ≤ K ≤ 200)** which denotes the number of moves. For each **K**, you will need to produce the maximum integer you can make from **S** by using at most **K** moves.

Output

Print the case number in a single line followed by query answers in a single line. Please see the sample for details.

Sample Input

```
3
123 3
1
2
3
011 2
1
2
111 2
1
100
```

Output for Sample Input

```
Case 1:
133
724
797
Case 2:
911
911
Case 3:
111
111
```

RUET CSE FEST 2k22

Problem J

K Subsequence Problem

Bangladesh Association of Problem Setters

There is an array of **N** integers. How many subsequences of it have at least **K** distinct numbers?

Input

The first line of the input contains **T(1 ≤ T ≤ 20)**, the number of test cases. The following **T** tests each contains two lines. The first line has two numbers **N** and **K(1 ≤ K ≤ N ≤ 100000)**. The next line contains **N** integers in range **[1..N]**.

Output

For each test print one line in the format "**Case X: Y**", where **X** is the case number and **Y** is the number of subsequences. Since the result can be very large, print it modulo **998244353**.

Sample Input	Output for Sample Input
3 3 2 1 2 3 3 1 1 1 2 3 2 1 1 2	Case 1: 4 Case 2: 7 Case 3: 3

Problem K

Change the usernames

One of the most popular online judges of Bangladesh, CodeMania (also known as CM) has recently introduced a new feature - now users can change their username if the new username they want is not being used by anyone else.
For example, let's say the user with a username **"theredwarrior"** wants to change it to **"thegreenwarrior"**. If no one is using it then the user should be able to change it to the new one. As soon as the change happens, the old username "theredwarrior" will become available for anyone to use. Users can request to change their username as many times as they want. But for a particular request, if the new username isn't available, then that operation will fail.

Suddenly the developers of CM are in a situation where they need to know what username an user initially had before the new feature was introduced. From the example above, the developers would like to know what username **"thegreenwarrior"** had before the new feature was in place. In that case, the answer should be **"theredwarrior"**.
The developers are in hurry and unfortunately, they are unable to find any optimized solution for it. So, they are asking you to help them out.

Input

The first line will contain **T ($1 \le T \le 10$)**, the number of test cases. Each case will start with **N ($1 \le N \le 100000$),** the number of instructions the developers give to you. Each of the instructions can be one of the following two types:

> **1 X Y:** This is the change instruction, username **X** wants to change to username **Y**. **X** is a username which someone is using at this time. Y is the new username that the user wants. **X** and **Y** will always be different.
> **2 X:** For the user who currently has username **X** the developers want to know what the username user initially had before the new feature took place.

In all the instructions, the given usernames will consist of only lowercase letters, and the length of the usernames will be at most **20**. The input file size will never exceed **10MB**.

Output

For each of the instructions of type 2, print the initial username the user had before the new feature took place. If the username was released by any earlier instructions provided by the developers and no one is currently using it, then just print **"Not in use!"** (without the quotes).

Sample Input

```
1
6
1 theredwarrior thegreenwarrior
1 theyellowwarrior thegreenwarrior
2 thegreenwarrior
2 theyellowwarrior
2 theredwarrior
2 theorangewarrior
```

Output for Sample Input

```
theredwarrior
theyellowwarrior
Not in use!
theorangewarrior
```

Explanation of Sample Input:

- **1 theredwarrior thegreenwarrior** (the username of **theredwarrior** is changed to **thegreenwarrior**)
- **1 theyellowwarrior thegreenwarrior** (instruction failed, **thegreenwarrior** already in use)
- **2 thegreenwarrior** (Initial name of **thegreenwarrior** was **theredwarrior**)
- **2 theyellowwarrior** (Initial name of **theyellowwarrior** is **theyellowwarrior.** In past instructions we can see **theyellowwarrior** wanted to change the user name which was unsuccessful.)
- **2 theredwarrior** (No one is currently using it)
- **2 theorangewarrior** (**theorangewarrior** was never changed, so initial username of **theorangewarrior** is **theorangewarrior**)