Brain Craft Ltd. - SUB Inter University Programming Contest 2019

Hosted by
State University of Bangladesh

9th March 2019
You get 17 Pages
10 Problems &
300 Minutes

Platform Support:

Problem Set By:

Contest Rules for SUB IUPC 2019

1. Solutions to the problems submitted for judging are called runs. Each run is judged as accepted or rejected by the judge, and the team is notified of the result. Only source code should be submitted, not the executables or any other files.
2. **Contest will be held on CodeMarshal online judge. Only this website will be accessible from contestant's pc. Any attempt to access other websites or the Internet will result in disqualification. Any attempt to tamper with the online judge will also result in disqualification.**
3. A contestant may submit a clarification request to the judges only through **CodeMarshal's** clarification system. If the judges agree that an ambiguity or error exists, a clarification will be issued to all contestants. Judges may decide not to answer a clarification at all in which case that particular clarification request will be marked as **IGNORED** in the CodeMarshal clarification page.
4. **If teams believe that there is something wrong with the judge data they are strongly advised to use the CodeMarshal clarification system to communicate with the judges rather than meeting them in person after the contest.**
5. Contestants are not to converse with anyone except members of their own team and personnel designated by the organizing committee while seated at the team desk. They may not even talk with their team members when they are walking around the contest floor to have food or any other purposes.
6. While the contest is scheduled for a particular time length **(five hours)**, the contest director has the authority to alter the length of the contest in the event of any unforeseen difficulties. Should the contest duration be altered, every attempt will be made to notify contestants in a timely and uniform manner.
7. A team may be disqualified for any activity that jeopardizes the contest such as dislodging extension cords, unauthorized modification of contest materials, distracting behavior or communicating with other teams. The judges can also recommend penalizing a team with additional penalty minutes for their distracting behavior.
8. 9-12 problems will be posed. So far as possible, problems will avoid dependence on detailed knowledge of a particular applications area or particular contest language.
9. **Rank-list will be frozen in the final hour of the contest. During this period, teams will only get verdict of their own submissions.**
10. Contestants cannot leave the contest arena during the contest without explicit permission from the judges. The contestants are not allowed to communicate with any other contestants (even contestants of the same team) or coaches when they are outside the contest arena.
11. Teams can bring any number of pages of printed materials with them. But they are not allowed to bring any electronic devices like calculator, CD, DVD, Pen-drive, IPOD, MP3/MP4 players, watches(smart, digital, analog) etc. **Teams CANNOT bring their own keyboard, mouse etc.**
12. With the help of the volunteers and judges, the contestants can have printouts of their codes for debugging purposes. Passing printout of codes to other teams is strictly prohibited and may cause disqualifications of teams involved.
13. Teams should inform the volunteers/judges if they don't get any verdict/reply within 10 minutes of submission/clarification. Teams should also notify the volunteers if they cannot log into CodeMarshal. These sort of complaints will not be entertained after the contest.
14. Codes must not use any system command or use multi-threading. Contestants must not attempt to access any other computers other than their own in the network. Violating these rules may result in disqualification.
15. **Teams using Java should be extra careful about TL and ML, as problems are usually not tested with Java.**
16. **Each team will be given the same machine in the same location during mock and main contest. That's why, teams are strongly advised to attend the mock. Any issues during mock should be notified to the judges via the clarification system.**
17. **Any member of a team, if late, may not be allowed to enter the contest arena after 30 minutes from the starting of the contest.**
18. **The decision of the judges is final.**

IDEs:

1. CodeBlocks
2. Geany
3. IntelliJ Idea
4. Netbeans

Compilers (In CodeMarshal):

1. C, C++: GCC 8.2
    a. C compile command: `gcc -O2 -static filename.c -lm`
    b. C++ compile command: `g++ -O2 -static -std=c++11 filename.cpp`
2. Java: JDK 1.8 OpenJDK

Java compile command: `javac -J-Xmx2048M filename.java`

A

Problem A

Input: Standard Input
Output: Standard Output

A Permutation with Mr. Peabody and Sherman

Mr. Peabody was teaching Sherman about permutations. A permutation of length **N** is a sequence of **N** numbers consisting of **1** to **N** exactly once. The order of the numbers is accountable. So, you can choose a permutation of length **N** in **N!** different ways.

Now, Mr. Peabody asked Sherman another question. He asked Sherman to pick a permutation of length **N**. Then asked him to calculate the number of displacement of that permutation.

If the **x**-th index *(1 based indexing)* of a permutation does not contain the number **x** then it is considered as a displacement. Here is an example. This is a **5** length permutation **{5, 2, 1, 4, 3}**. The first place has number **5**, so this is a displacement. But the second place has the number **2** which is not a displacement. Like this way the third and fifth places have displacements but the fourth place does not. So, this permutation has overall **3** displacements.

If Sherman picks a permutation of length **N** randomly, what is the expected number of displacements of that permutation? In simple words, you have to find average number of displacements from all the permutation of length **N**. For **N** = **2**, there are two permutations, **{1, 2}** has **0** displacement and **{2, 1}** has **2** displacements. So, there is **1** displacement on average.

Input

Input starts with an integer **T**, denoting the number of test cases. For each test case there will be an integer **N**, denoting the length of the permutation.

Output

For each test case print a single line in this format **Case X: A/B**. Here **X** denotes the test case number. **A/B** denotes the expected number of displacements where **A/B** is a fraction. **A** denotes the numerator and **B** denotes the denominator where **gcd(A, B)** is **1**.

Constraints

- **1 ≤ T ≤ 22**
- **1 ≤ N ≤ 22**

Sample Input

```
1
2
```

Output for Sample Input

```
Case 1: 1/1
```

B

Problem B

Input: Standard Input
Output: Standard Output

Satisfy the Constraints

In this problem, all you have to do is generate an array having **N** positive integers which satisfies **Q** constraints.

Every constraint is of the following kind - "The greatest common divisor of all the numbers from the **L**-th position to the **R**-th position of the array has to be a multiple of **X**".

You need to find the lexicographically smallest array which satisfies all the constraints.

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 follow. 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

- $1 \le T \le 100$
- $1 \le N \le 10^5$
- $1 \le Q \le 10^5$
- $1 \le L \le R \le N$
- $1 \le X \le 10^6$
- Both, sum of **N** and sum of **Q** over all test cases $\le 4 \times 10^5$

Output

For each test case, print a line containing the case number and the lexicographically smallest array which satisfies all the given constraints. As the numbers of the array can be very large, output the numbers modulo **1000000007**.

Sample Input

```
1
7 2
2 5 2
5 7 3
```

Output for Sample Input

```
Case 1: 1 2 2 2 6 3 3
```

An array **C** having size **N**, is called lexicographically smaller than another array **D**, having the same size, iff there exists an **'i'** $\le$ **N** such that:
C[p] == D[p] for all $1 \le p < i$ and **C[i] < D[i]**

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

C

Problem C

Input: Standard Input
Output: Standard Output

Bittersweets of Our Community

The history of ICPC style programming contests in Bangladesh started in the late 1990s. Since then, this particular event has created a large community in our country. Through programming contest, we have made friends from different institutions and localities across our national border. Sometimes we cannot explain our friendship to people outside this community. However, we ourselves feel the strong bond. Often it feels easier to open up to a fellow contestant than to a blood relative.

Usually after graduation, an experienced contestant becomes a problem setter to be in touch with the community. Also after graduation, people get married. So contestants and problem setters get wedding invitations of contestant alumni every now and then. Nobody wants to miss such events. But our hearts break down when we find a Programming Contest and a wedding invitation scheduled for the same day. We want to attend both events. So, we will surely attend the wedding if it is not on the day of contest. Even if it is the day before the contest or the day after contest, we are ready to travel long distance. But, if both the event dates are same, we will attend the programming contest for sure (as this is the glue that holds us together) and attend the wedding only if it is an invitation for dinner in the same city where the contest is going to be hosted. Otherwise, we will just curse the alumnus who scheduled her/his wedding on such a day (yes, we will not criticize the contest schedule).

You are given a list of contest schedules for this year and an upcoming wedding invitation (also in this year). You have to figure out whether the programming contest community can attend the wedding.

Input

There will be **T** $(1 \le T \le 5)$ test cases. Each case will start with a line having four integers D_w $(1 \le D_w \le 31)$, M_w $(1 \le M_w \le 12)$, C_w $(1 \le C_w \le 10)$ and R_w $(0 \le R_w \le 1)$ denoting the Day, Month, City and timing of the wedding invitation. Value of R_w will be **1** if the invitation is for dinner, **0** otherwise. The next line will contain an integer **P** $(1 \le P \le 10)$ denoting the number of scheduled programming contests. Then **P** lines will follow each having three integers D_i $(1 \le D_i \le 31)$, M_i $(1 \le M_i \le 12)$ and C_i $(1 \le C_i \le 10)$ denoting the Day, Month and City of the *i-th* contest. You can assume, all dates are valid according to the calender of year 2019 and no two contests are on the same day.

Output

For each case, print the case number first in the format "**Case X:** " without the quotes. Then print **Yes** or **No** depending on whether we can attend the wedding invitation or not, followed by a new line. See sample I/O for more clarity.

Sample Input

```
2
22 2 3 1
2
23 2 5
9 3 5
23 2 5 0
3
23 2 5
9 3 5
6 4 5
```

Output for Sample Input

```
Case 1: Yes
Case 2: No
```

D

Problem D

Input: Standard Input
Output: Standard Output

Smartest Subarray

You will be given an array **A** consisting of **N** integers. You have to find the maximum smartness score among all the subarrays of **A** consisting of **at least two elements**. Subarrays are contiguous segments of an array. For example, the subarray[u, v] is a contiguous segment of an array that starts at index u and ends at index v where $\mathbf{1 \le u \le v \le N}$.

For any **subarray[u, v]**, the smartness score of that subarray = **(P - Q)** × **L** where,

- **P** = Largest integer in the subarray **A**[u, v]
- **Q** = Second largest integer in the subarray **A**[u, v]
- **L** = Length of the subarray, which is basically (v - u + 1)

For example, let's consider the array in our first sample input where **A** = [6, 4, 3, 2, 7, 7]. For the subarray **A**[2, 5], the largest and the second largest elements are respectively 7 and 4. So the smartness score of this subarray is (7 - 4) × 4 = 12. But if we consider the subarray **A**[3, 6], then both the largest and the second largest elements will be 7. So the smartness score of this subarray will be (7 - 7) × 4 = 0. So 12 is the maximum score we can get considering all the subarrays of **A**.

Input

The first line of the input contains a single integer **T**, which denotes the number of test cases. Each case consists of two lines. The first line contains an integer **N**. And the second line contains **N** space separated integers of the array **A**.

Output

For each test case, output a single line in the format "**Case X: D**" without the quotes. Here, **X** is the case number and **D** is the desired answer denoting the maximum smartness score you can achieve.

Constraints

- $\mathbf{1 \le T \le 20}$
- $\mathbf{2 \le N \le 10^6}$
- $\mathbf{1 \le A_i \le 10^9}$
- The sum of **N** over all the test cases in an input file $\mathbf{\le 10^6}$

Sample Input	Output for Sample Input

2
6
6 4 3 2 7 7
6
6 7 3 2 7 2
	Case 1: 12
	Case 2: 16

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

E

Problem E

Input: Standard Input
Output: Standard Output

Carry My Stuff

You all know that US-based company BezosCorp™ is the world's largest online shop. However, their shipping charge to Bangladesh is very high, and the process is highly inconvenient. CarryMyStuff Inc. tries to solve this problem. They have a network of travelers who travel from the US to Bangladesh regularly. CarryMyStuff distributes orders placed by customers among these travelers.

Yes, there are customs issues. Border control does not allow a traveler to carry more than one product of the same type i.e. they will not allow someone to carry more than one laptop. But he or she can carry at most one item of several types of products. For example, a traveler is allowed to carry one laptop, one cellphone, and one wrist watch. (Maybe the law is slightly different than this in reality, but assume it to be true for this problem.)

Festival season is coming. Lots of orders will be placed. And the customers are unwilling to wait for a long time for their orders to arrive. Using the power of machine learning and telepathy, you have acquired a list of near future orders and a list of near future travelers' schedule. However, you are not allowed to assign an order to a traveler who is supposed to travel on an earlier date than the day when the order will be placed. Your task is to make an arrangement in order to minimize the maximum waiting time (in days) among all orders. For simplicity, assume that the traveler reaches Bangladesh the same day he departs from US. And CarryMyStuff delivers the product to the customer on that same day too. This means that waiting time is the difference between order placement day and traveler's departure day. Also a traveler can carry products ordered on the same day. In that case the waiting time will be zero days.

Input

The first line of input will contain an integer **TC**, denoting the number of test cases. Each case starts with a line containing three integers **P**, **N**, **T**; which respectively indicate number of different type of products, number of orders, number of travelers. Each of the following **N** lines contain two integers **$\textbf{productType}_i$** and **$\textbf{orderDay}_i$**, denoting the type of product and the day - on which the **i**th order will be placed. Each of the following **T** lines contains an integer **$\textbf{travelDay}_i$**, denoting the day when the **i**th traveler will departs for Bangladesh.

Constraints

- **$1 \le TC \le 5 \times 10^4$**
- **$1 \le \text{productType}_i \le P$**
- **$1 \le \text{orderDay}_i \le 10^9$**
- **$1 \le \text{travelDay}_i \le 10^9$**
- **$1 \le$ sum of P among all test cases $\le 5 \times 10^4$**
- **$1 \le$ sum of N among all test cases $\le 5 \times 10^4$**
- **$0 \le$ sum of T among all test cases $\le 5 \times 10^4$**

Output

For each test case, firstly print the case number. After that, if no arrangements can be made, then print **-1**. Otherwise, print an integer indicating the minimized maximum waiting time (in days) among all orders.

Sample Input

```
4
2 2 2
1 1
2 75
76
88
1 1 1
1 10
2
1 1 1
1 1
1
1 2 1
1 1
1 2
2
```

Output for Sample Input

```
Case 1: 75
Case 2: -1
Case 3: 0
Case 4: -1
```

Explanation

Case 1: Order placed in day 1 has to wait for 75 days, as there is a traveler traveling on day 76. Assigning the products to the traveler flying out on day 88, will result in waiting time longer than 75 days.

Case 2: Only order is placed on day 10. And there are no flights on or after that day.

Case 3: Traveler can carry products ordered on the same day.

Case 4: Only traveler, flying out on day 2, can carry at most one product of type 1. Hence the other order of type 1 cannot be delivered in any arrangement.

F

Problem F

Input: Standard Input
Output: Standard Output

Cut the Rope

Cut the rope is an interesting simulation game where you are given a sequence of numbers from **1** to **N**, arranged in increasing order from left to right. There can be two types of operations and you need to deal with them. The operations are described below:

1. The query operation, where given **L** and **R**, a genie asks you to find the *bitwise XOR* of all the numbers in the list from the **L-th** position to the **R-th** position. **L** and **R** are both inclusive and follow 1-based indexing.

2. Cut the rope! Where given three positive integers **L, R** and **K**, delete all the numbers from the following indices **J**, where **L ≤ J ≤ R** and **(J - L) % K ≠ 0**, where **%** is the modulo operator.
   In other words, delete all the numbers in the segment **[L:R]**, except for the first number and every subsequent **K-th** number in the segment.

After the numbers are removed by an operation of the second type, the remaining numbers are shifted such that there are no gaps between them, preserving their relative order, and a new sequence is formed for the next set of operations. That is, if the size of the sequence before the operation was **M**, and **C** numbers were deleted, then the size of the new sequence is (**M - C)**. The next operations will follow indexing scheme for the new sequence, i.e, from **1** to (**M - C)** up until some numbers are deleted again. When this happens a new sequence is formed again and the same process is repeated and so on.

Please note that any particular delete operation is atomic. This means that all the numbers are deleted at the same time, so the new sequence is formed only after all of them are removed. So the indexing does not change until the whole operation completes.

Input

The first line contains an integer **T**, denoting the number of test cases. Each test case starts with two integers **N** and **Q**, denoting the initial size of the sequence and the number of operations respectively. Each operation is given in either one of the following two formats:

1 L R, denoting an operation of the first type, where you need to find the bitwise XOR of all the numbers from the **L-th** position to the **R-th** position.

2 L R K, denoting an operation of the second type, where you need to cut the rope using the procedure described as above.

It is guaranteed that **L ≤ R**. Also **L** and **R** will be valid positions in the current sequence, i.e, if the size of the current sequence is **M**, then **1 ≤ L ≤ R ≤ M**. There will be at least one number in the sequence at any given moment, and the list will not be empty even after all operations are completed. There will be at least one operation of the first type.

Constraints

- **$1 \le T \le 10$**
- **$1 \le N \le 10^{18}$**
- **$1 \le Q \le 10^{5}$**
- **$1 \le L \le R \le M$, where M is the size of the current sequence**
- **$1 \le K \le 10^{18}$**
- **Sum of Q over all the test cases $\le 10^{5}$**

Output

For each test case, you need to print the case number in one line of the format: "**Case X:**".
Then for each operation of the first type, calculate and output the *bitwise XOR* of all the numbers from **[L:R]** in a single line. Refer to the sample i/o section for better understanding.

Sample Input

```
2
16 5
2 3 10 3
1 3 7
2 1 11 4
2 2 3 100
1 1 2
12345678 2
2 12345 67890 1000
1 1 3
```

Output for Sample Input

```
Case 1:
11
8
Case 2:
0
```

Explanation For Case 1

Initial Sequence: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16

After the first operation (2 3 10 3),
Old sequence: 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
New sequence: 1 2 3 6 9 11 12 13 14 15 16

After the second operation(1 3 7),
Sequence: 1 2 3 6 9 11 12 13 14 15 16
Answer: **11**

After the third operation (2 1 11 4),
Old sequence: 1 2 3 6 9 11 12 13 14 15 16
New sequence: 1 9 14

After the fourth operation (2 2 3 100),
Old sequence: 1 9 14
New sequence: 1 9

After the fifth operation (1 1 2),
Sequence: 1 9
Answer: **8**

G

Problem G

Input: Standard Input
Output: Standard Output

Precision Eroor

Solving problem is an art. There are so many online judges and onsite contests now a days. Problem setters are always busy with generating new and creative ideas for setting a problem, writing solutions and preparing input/output data for testing different solutions. When a solution is submitted, a comparison is done between the output file generated from the submitted solution and the output file generated from judges solution. When both output files match exactly, the submitted solution is awarded an accepted verdict. But, for comparing output files with floating point values, exact match is not always a proper way because of some limitations of modern machines. So, judges consider an Absolute Error or a Relative Error for comparing output files with floating point values. Specially for the solutions that deal with precision.
Let's consider judge's output is **A** and submitted solution's output is **B**, where A and B are real numbers in decimal number system.
then absolute error $E_{absolute} = |A - B|$ and relative error, $E_{relative} = \frac{|A-B|}{|A|}$
This is the night before programming contest, and a lot of discussions are going on among the judges, whether they would consider the absolute error or the relative error. While discussing, the setter of the problem that deals with precision came up with a different idea!

So, the setter decided to do manual judge.

For given two values, **A**$\geq$**1** and **B**$\geq$**1** he will do manual judging and will give accepted verdict if any of the following happens:

1) Numerical value of **A** and **B** are same, that is **A = B**.
2) Both **A** and **B** has same number of digits before the decimal point, and first **K** digits of **A** match with first **K** digit of **B**.

If a solution is not **accepted**, it is considered as **rejected**.

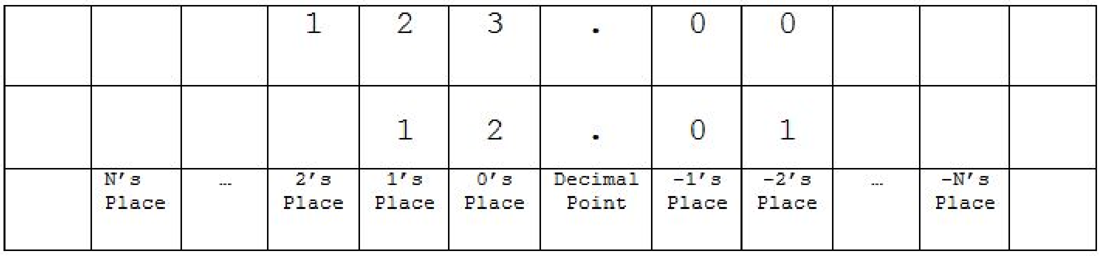
Example 1: A=10.00 and B = 10,
Here the numerical value of A and B are same. So, the verdict will be accepted.

Example 2: A = 526.10 and B = 526.11111

			5	2	6	.	1	0			
			5	2	6	.	1	1			
	N's Place	…	2's Place	1's Place	0's Place	Decimal Point	-1's Place	-2's Place	…	-N's Place	

Here, starting from 2's place, first four digit of both outputs match. Note that, we ignored leading zeros. So, for K$\leq$4, answer will be accepted and for K$>$4 answer will be rejected.

Example 3: A = 123.00 and B = 11.01
Here we can clearly observe that A has first digit at 2’s place, but B doesn’t have any value at 2’s place. So, the verdict will be rejected.



			1	2	3	.	0	0			
				1	2	.	0	1			
	N’s Place	…	2’s Place	1’s Place	0’s Place	Decimal Point	-1’s Place	-2’s Place	…	-N’s Place	

You are given **K** , **A** and **B**. You have to determine whether the verdict will be **accepted** or **rejected**.

Input

Input starts with an integer **T** denoting the number of test cases in a single line. Each of the next **T** test cases will contain **K** , **A** and **B.**

Constraints

- **$1 \leq T \leq 1000$**
- **$1 \leq K \leq 10000$**, K is an integer.
- **$A \geq 1$**
- **$B \geq 1$**
- **A** and **B** does not contain any leading zeros. But, they may contain trailing zeros at the end after the decimal point. They both are real numbers, each consisting of at most **100** digits.

Output

For each test case, output a single line in the format “**Case X: D**” without the quotes. Here, **X** is the case number and **D** is the desired answer.

Sample Input

```
4
2 1 1.0
3 12.45999 12.4600
3 12.45 12.3
10 12.45 12.4500
```

Output for Sample Input

```
Case 1: accepted
Case 2: accepted
Case 3: rejected
Case 4: accepted
```

You may safely assume that judges did not lose any precision during making this problem.

H

Problem H

Input: Standard Input
Output: Standard Output

Horrible Pass

Doboland consists of **N** islands. They are numbered from **1** to **N**. There are some unidirectional bridges between some pairs of islands. Any island **B** is a neighbour of island **A** if there is a unidirectional bridge from island **A** to island **B**. No island is connected to itself by a bridge. Admiral general Larry is a crazy dictator of Doboland. He imposed several freaky rules for the inhabitants of Doboland. People are not allowed to use ship or flight for going from one island to another island. Even, they are not allowed to use those bridges to leave an island if they don't have appropriate leaving pass.

If anyone wants to leave an island, he/she has to collect a leaving pass from the passport office of that island. Every island has its own passport office. A leaving pass collected from an island contains a list of its neighbour islands (that list may not contain all of the neighbour islands). After getting the leaving pass, the pass holder can go to one of the islands mentioned in the leaving pass. Neither he/she is allowed to go to any other island out of that list nor to collect a new leaving pass. Leaving passes are one time usable and it expires immediately after leaving the current island. But the pass holder can stay in that island as long as he/she wants, even for the lifetime.

To create a permitted island list, all the passport offices follow the same strategy. They put all the neighbour islands in the list if the number of neighbour islands is less than or equal to **P**. Otherwise they randomly choose **P** neighbour islands for that list. All the neighbor islands have equal probability of being chosen for the list.

Boko lives in island **H**. He wants to visit all the neighbour islands of island **H** exactly once. He has promised that, he won't visit an island twice (even he won't come back to island **H** any more after leaving it once) and won't enter any island that is not a neighbour of island **H**. In any time of his journey, if he finds that he is not able to keep this promise, he will choose to stay forever in the island where he is now. He has the complete information about all the islands and bridges of Doboland. Since he has the complete map of the Doboland, he will always choose the optimal next island from the island list of leaving pass to maximize the probability of completing his journey (visiting all the neighbour islands of the $\mathbf{H}^{\mathbf{th}}$ island).

Calculate the maximum probability of completing Boko's journey if he behaves optimally.

Input

Input starts with an integer **T (1 ≤ T ≤ 20)** denoting the number of test cases followed by a blank line.

The first line of each test case contains three integers **N (2 ≤ N ≤ 14), H (1 ≤ H ≤ N)** and **P (1 ≤ P < N)** as described above. Following N lines describe the neighbour lists of N islands (i^{th} line describes the neighbour list of i^{th} island). Each neighbour list description starts with an integer $\mathbf{L_i}$ (**1 ≤** $\mathbf{L_i}$ **< N**) denoting number of neighbours of i^{th} island followed by $\mathbf{L_i}$ space separated distinct integers denoting neighbour islands. Consecutive test cases are separated by a blank line.

Output

For each case, print the case number first, then print desired maximum probability. Absolute errors less than **10^{-6}** will be ignored.

Sample Input

```
3

3 1 1
2 2 3
2 1 3
2 1 2

3 1 2
2 2 3
2 1 3
2 1 2

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

Output for Sample Input

```
Case 1: 0.5000000000
Case 2: 1.0000000000
Case 3: 0.2962962963
```

I

Problem I

Input: Standard Input
Output: Standard Output

Fencing The T-Rex

A genetically modified T-Rex escaped from the facility last year. Luckily we were able to track it, and now we must have to confine it from human civilization.

The goal is to fence the T-Rex very tactically. You are here to help designing a prison facility for it bounded by electric fences. There are **N** points considered as the source of electricity. You can connect any two points with electric wire in a straight line, and create an electric fence.

Now, as you know the T-Rex is extremely clever, there are some restrictions you must have to maintain. Obviously the facility should be a polygon with nonzero area enclosed by the fences, otherwise it will definitely escape through. Next, it must be convex, because the T-Rex knows how to break through a concave facility! Also for obvious reasons you have to maximize the area of the facility.

After building the fences you have to build another special cage within it to keep the T-Rex. The T-Rex, if escaped, will always run to the fence closest to it. So, for building this special cage with the facility you need to calculate a particular risk metric. For that, you have to find the strictly interior point of the convex facility furthest from the fences, and determine its distance from the closest fence. That means, fix the interior point in such a way that, the minimum distance from any fence to it will be maximized.

Input

The first line will contain an integer **T** (**1 ≤ T ≤ 100**), the number of test cases.
Each test case will contain an integer **N** (**3 ≤ N ≤ 8000**) in the first line. Next **N** lines will contain two space separated integers denoting the coordinates of the points of electric source. There might be duplication of points. **Absolute value of all coordinates are not greater than 150000**.

Output

Print the required distance rounded to nearest integer along with test case. If it is not possible to build the facility, print **-1** along with test case. Please see sample for detail.

Sample Input

```
2
6
0 30
100 10
240 30
240 70
90 90
0 70
5
0 0
10 2
9 4
5 5
0 1
```

Output for Sample Input

```
Case 1: 39
Case 2: 2
```

J

Problem J

Input: Standard Input
Output: Standard Output

Evil Corps

So in this problem, you are a programmer working for Evil Corps, the multi-billion dollar company everyone is in love with. Your latest product has just been launched. Now your manager has come to you with a bunch of data related to the population of different regions and the profit your new product has made in those regions. Formally, there are **N** different regions and for the *i-th* region, you have one data point in the form of $\mathbf{x_i}$ and $\mathbf{y_i}$. Here **x** denotes the population of a region and **y** denotes the profit your product has made in that region. Now there are **Q** more regions where your company wants to expand its business and they want to start by launching your product in those regions first. Now the manager has given you the population of each of the **Q** new regions. You have to predict the profit your product should make in those regions.

But correlating population of a place with the profit of your product seems too simplistic. There can be a thousand more factors in play here! But it's Evil Corps. You just follow your manager's commands here most of the time. And now, someone from your team comes up with the following solution.

You can construct polynomials which pass through the points of the dataset given by the manager. Formally, in this solution, you have to come up with a polynomial **p** with the property,

$\mathbf{p(x_i) = y_i \quad for\ 1 \le i \le N}$

And then another math geek of your team found out that the polynomial with the lowest possible degree which has the above property is actually unique! Of course, your manager then demanded that you have to find out the polynomial with the lowest possible degree. After that, you can just fit the population of a new region into that polynomial and evaluate it to get the predicted profit of that region. Your manager is happy with this solution. So all that is left to do, is to solve the problem within reasonable time and memory.

Input

There are **T** test cases. For each case, in the first line, there is one integer, **N**. **N** lines will follow. The *i-th* line here will contain two integers $\mathbf{x_i}$ and $\mathbf{y_i}$. After that, the next line will contain one integer, **Q**. Then there will be **Q** lines where each line will contain one integer representing the population of a new region.

Constraints

- **$1 \le T \le 10$**
- **$1 \le N \le 5000$**
- **$1 \le Q \le 5000$**
- **$0 \le x_i$, population of any region $\le 10^5$**
- **$-10^5 \le y_i \le 10^5$**
- **All x_i are distinct**
- **Sum of N over all test cases ≤ 5000**
- **Sum of Q over all test cases ≤ 5000**

Output

For each case, print one line stating the case number. Then **Q** lines should follow where the *i-th* line should contain the predicted profit for the *i-th* region. The final result can be expressed as **A/B** where **A** and **B** are **non-negative coprime integers** and **B != 0**. Since the results can be very huge, print the value of **($A*B^{-1}$) modulo 100019**. Here **B^{-1}** denotes the modular inverse of **B modulo 100019**, that is, (**$B*B^{-1}$ modulo 100019) = 1**. You may assume that there will be a **unique** modular inverse of **B** modulo **100019**.

Sample Input

```
3
3
3 -7
4 6
1 -15
2
2
5
4
3 15
8 40
129 645
5 25
2
0
11
10
35616 59012
22527 95426
96547 21687
96265 36408
57902 24004
1914 69996
67632 14333
31754 60214
12320 18205
67074 24148
5
43371
59083
85377
2229
84845
```

Output for Sample Input

```
Case 1:
100005
25
Case 2:
0
55
Case 3:
74651
64073
38972
94709
75614
```

Explanation

In the first test case, the polynomial p looks like this,
$p(x) = 3x^2 - 8x - 10$

so if we evaluate p(5) here, we would get,
$p(5) = 3 \times 5^2 - 8 \times 5 - 10 = 25$