Quest of Passion 1.0 - Hours of Code

https://toph.co/c/seed-to-big-quest-passion-1-0-hour-code

Toph

Schedule

The contest will run for **3h0m0s**.

Authors

The authors of this contest are AST_TheCoder, Bruteforcekid, crevious509, Tanmoy_Datta, and TwiiN_N.

Rules

This contest is formatted as per the official rules of ICPC Regional Programming Contests.

You can use C++11 GCC 7.4, C++14 GCC 8.3, C++17 GCC 9.2, C11 GCC 9.2, Free Pascal 3.0, Java 1.8, Kotlin 1.1, Python 2.7, Python 3.7, and Python 3.8 in this contest.

Be fair, be honest. Plagiarism will result in disqualification. Judges' decisions will be final.

Notes

There are 8 challenges in this contest.

Please make sure this booklet contains all of the pages.

If you find any discrepencies between the printed copy and the problem statements in Toph Arena, please rely on the later.

A. Thanks , Seed to BIG

Seed To BIG is Presenting one of the biggest Online Fiesta of the year, **“Kazi and kazi Tea, Quest of Passion 1.0”** . Due to pandemic, students are passing time at home by scrolling down social media or doing nothing which will value their time. This is the programming event.

Authors and Judges worked very hard to make the round great. So, I think we should be thankful to them. To express your gratitude, print **“Thanks, Seed to BIG”** without quotes.

If you want to judge the perfection of the contest before thanking them, you can. But, before doing this, think about the leaderboard once. :-)

Input

There’s no input.

Output

Described in statement.

B. Imran and His Sum

Imran and Jakiya are two friends. They always do some contest to prove their ability in front of one another.

This time Jakiya want to test the ability of problem solving of Imran. So she gave an integer N, and then build an array A = [1, 2, 3, …, N]. For each subarray of length less than or equal to K, compute the product of the subarray's elements. Output the sum of these products, modulo M.

Imran has fear that he can't do it . Let's see if you can help imran to solve the problem and print the output.

Input

The first line contains three integers N, K and M.

Output

Print the answer on the first line.

Samples

Input	Output
5 3 100	45

The answer before begin truncated modulo 100 was 145. For K = 3 and N= 5 all subarray are (1),(2),(3),(4),(5),(1,2),(2,3),(3,4),(4,5),(1,2,3),(2,3,4),(3,4,5) .The product of the subarray's elements and then the sum of these products$=

```
1+2+3+4+5+(1*2)+(2*3)+(3*4)+(4*5)+(1*2*3)+(2*3*4)+(3*4*5) = 145 $
```

Output = 145 % 100 = 45

Input	Output
10 5 666013	68893

Constraints and notes

- $1 \leq N \leq 10^{18}$
- $1 \leq K \leq min(600, N)$
- $1 \leq M \leq 10^{18}$

C. AST_TheCoder, the Prime Minister

AST_TheCoder is the new prime minister of programming country which has **n** cities connected by **m** roads. To ensure the safety, all the roads are directed between cities in the country. But, PM noticed that, for some pair of cities **u & v**, there's not any path **u to v** or **v to u** or both. Now, he wants to improve the country road connection, such that, for every pair of cities **u & v**, it's possible to reach **v from u** and **u from v**. And he wants to do it in minimum possible cost. He can perform two types of operations.

One: He can change the direction of an existing road for free.
Two: He can build a road between two cities for cost 1.

As, AST_TheCoder is not good in logical thinking and also you are the greatest coder in the country, he comes to you to compute the minimum cost to reach his target.

Note: A road can connect same city. And also more than one road can connect same pair of cities.

Input

First line contains two integers **n (1 ≤ n ≤ 10^5)** and **m (0 ≤ m ≤ 10^5)**, number of cities and number of roads respectively. Each line of next **m** lines contains two integer **u & v (1 ≤ u, v ≤ n)**, which means a directed road from **u to v**.

Output

Print the minimum cost to change the road map, such that, for every pair of cities **u & v**, it is possible to go to **v from u** and **u from v**.

Samples

Input	Output
3 2 1 2 1 3	1

Input	Output
7 0	7

D. Mission: Help an Old Friend in the Land of Tea

The twin brothers from the land of tea have set up n shops on the road from the village hidden in leaves to the village hidden by sand. And they asked Lady Tsunade to build watchtowers on the road to protect their shops from Gaara's bloodthirst. Every watchtower has a constant range in both directions of the tower. Every watchtower has a building cost of **k** and a range cost of **j**, so the total cost of building a tower is **k+j**. When Tsunade builds a tower she can set its range (the value of **j** for that tower). Value of **k** is constant for every tower but the value of **j** for each tower can vary.

A tower with **j = 0** can only cover the shop that is at the same position as the tower. A tower with **j = x** can cover all shops within **x** *units* to its left and **x** *units* to its right **including the point the tower is built on.**

Tsunade has to build watchtowers on the road so that all the shops have watchtower coverage. As lady Tsunade has to save up money for her gambling, she also wants to minimize the total cost of building the towers. As her subordinate, you have to help her find the optimal positions, **p** to place the towers and the value of **j** for each tower.

A tower can also be built on the same position as a shop.

Input

First line of the input will contain one integer **t**, the number of test cases. The first line of each test case will contain a number **n**, number of shops on the road and a number **k**, building cost of each tower. Next line will contain **n** integers denoting the position, **p** of every shop (the distance of the shops from leaf village).

Constraints:

- 1<=t<=100
- 0<=n<=5 * 10^3
- 0<=k<=10^6

- 0<=p<=10^6

It is guaranteed that the value of total cost will fit inside a 32bit signed integer. It is guaranteed that the sum of **n** over all test cases does not exceed 10^3.

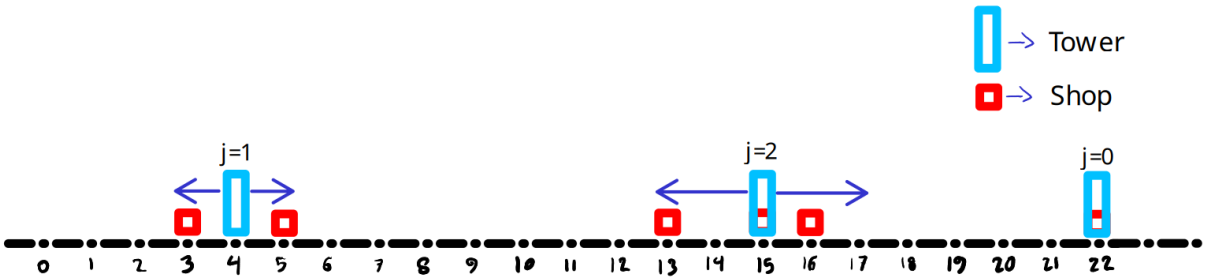
Output

For each test case, first line should contain two integers, the number of towers to build and the total cost, followed by position,**p** and range,**j** of all towers in separate lines.

There may be multiple optimal answers for the same total cost, output any one of them.

Samples

Input	Output
1 6 2 3 5 13 15 16 22	3 9 4 1 15 2 22 0



The tower at position **4** has a range of **1** and covers **1** *unit* in both directions.

The tower at position **15** is built at the same position as the shop at position **15**. It covers the shops at position **13**,**15** and **16** with range of **2**.

The tower at position **22** has a range of **0** and only covers the shop that is at the same position as the tower.

Input	Output
1 5 5 3 3 11 11 15	1 11 9 6

E. AST_TheCoder, the Security Guard

This is an interactive problem.
AST_TheCoder is the new security guard of programming country which has a big entrance gate. You want to enter the programming country. But you have to tell the security key to the guard. **The security key is nothing but the sum of all values of a secret array of size n.** The secret array is only known by the guard. AST_TheCoder won't say you the array $a_1, a_2, a_3, a_4, ……., a_n$, but he will answer at most **43 queries** of you.

Query type: Print **"1 x"** without quotes, where $1 \leq x \leq 43$. AST_TheCoder will replace the values with $\lfloor a_i / 2^{x-1} \rfloor$ **XOR** $(2^x - 1)$ where $1 \leq i \leq n$. Then, he will tell you how many odd numbers the replaced array has now, and he will return to the main array again.

Answer: Print **"0 sum"** without quotes, when you got the security key.

Note:
It is certain that every value of the array will be non-negetive and less than 8,796,093,022,208.
'⌊ ⌋' means the floor value of a fractional number and XOR means Bitwise Exclusive OR.

Input

Take n $(1 \leq n \leq 10^5)$ as input, the length of the array.
Then you will get an integer **odd_cnt** for every query. **odd_cnt** is the number of odd values in the replaced array said above. System will return a string **"enter"** without quotes as input, when you give the right answer, **"go_back"** otherwise.

Output

For query, print **"1 x"** without quotes, where $1 \leq x \leq 43$. System will give the answer described above.
For answer, print **"0 sum"** without quotes. **sum** is the summation of all the values in the secret array.

Note:
If you do more than **43 queries** or print a wrong security code, you will get **wrong**

answer as verdict.
Don't forget to flush the output after every query and answer. For example, you can use *fflush(stdout)* in C & C++, *System.out.flush()* in Java, *flush(output)* in Pascal and *stdout.flush()* in Python etc.

Sample

Input
3
2
0
enter

Output
1 1
1 3
0 13

Explanation
The secret array is [4, 5, 4]. After the first query, the replaced array will be [5, 4, 5] which has 2 odd numbers. And, after the second query, the replaced array will be [6, 6, 6] which has not any odd numbers.

F. Alice and Bob Very Sus!

There is one imposter between Alice and Bob. Everyone is suspecting only the two of them.

This is not a typical Alice and Bob scenario where they play optimally and control the result. Here they have no impact on the outcome. They are neither going to vote anyone. Everyone else votes either Alice or Bob.

You will be given **Q** queries. There are two types of queries.
1. $1\ x$: Add a voter having favourite number x.
2. $2\ k\ L\ R$: Report who will be ejected if everyone added up until now votes. The first k (in the order they were added) voters will skip. It is guaranteed that atleast k voters have been added up until now. Also voters having their favourite number out of the range $[L, R]$ will skip. The rest will vote according to one simple rule: **If a voter's favourite number is odd, he/she will vote for Alice. Voters having even favorite number will vote for Bob.** The person with the **most vote will be kicked out**.

Input

The first line contains a single integer Q
Each of the next Q lines have one of the two types of queries.

Output

After each query of type 2, print one of these followed by a newline:
if Bob was ejected, print "**Bob was ejected!**"

if Alice was ejected, print "**Alice was ejected!**"
if it's a tie, print "**No one was ejected!**"

Samples

Input	Output
10 1 5 1 4 1 6 1 7 1 9 2 0 1 10 2 1 1 10 2 1 2 8 1 3 2 1 2 8	Alice was ejected! No one was ejected! Bob was ejected! No one was ejected!

These are the explanation for the first 3 queries of type 2:
1. No one skips vote. And everyone votes (for everyone's favourite number x). 3 of them have odd favourite numbers, while 2 have even. So Alice gets kicked out.
2. 1st voter skips voting. Among the others, there are equal number of voters with odd and even favorite numbers on the given range. So no one is ejected.
3. 1st voter skips voting. Among the others, there are 2 voters whose favourite number is even and is in the range $[2, 8]$, while there is only 1 voter with whose favorite number is odd. So Bob gets ejected.

G. Happy Shopping

Mr. and Mrs. Smith are a very happy couple. But Mrs. Smith likes online shopping very much and it gives her happiness. For this reason, sometimes she orders some unnecessary extra stuff. On the other hand, Mr. Smith doesn't like to spend too much money. Now at the beginning of the month, the Smith family needs to buy **n** necessary things with some fixed amount of minimum quantity on each item. For this, Mrs. Smith is planning to buy those things from an online shop. As Mr. Smith knows that his wife will buy something extra and will spend unnecessary money, he gave her some fixed amount of money **m** and he fixed some maximum number x_i on each of the n items which are needed to be bought from the online shop so that Mrs smith can't buy i^{th} product greater than x_i quantity, and she can't spend more than **m** money.

Every product available on the online shop has some happiness factor for Mrs. Smith. So that different product gives her different happiness. For example, if a product has a happiness factor y_i then if she buys this product then her happiness will be increased by y_i. Now Mrs. Smith wants to find the maximum amount of happiness she can get from online shopping so that Mr. Smith's conditions will not break and she will get all the necessary things on the list in minimum quantity. If it is impossible for her to buy each of the necessary things with a minimum quantity then she will not buy anything. Can you help her with this problem?

Input

1st line of input contains two integer n and m which are the number of elements and amount of money.

2nd line of input contains n space separated integer where i^{th} integer represent price of i^{th} item.

3rd line of input contains n space separated integer where i^{th} integer represent happiness factor of i^{th} item.

4th line of input contains n space separated integer where i^{th} integer represent minimum number of quantity of i^{th} item.

5th line of input contains n space separated integer where i^{th} integer represent maximum number of quantity of i^{th} item.

- $1 \le n \le 100$
- $1 \le m \le 1000$
- $1 \le price_i \le 100$
- $1 \le happiness_i \le 1000$
- $1 \le min\_quantity_i \le 50$
- $1 \le max\_quantity_i \le 50$

Output

The output contains a single integer, denoting the maximum amount of happiness Mrs Smith can get from online shopping.

Samples

Input	Output
3 30 2 4 3 2 1 3 2 1 1 4 3 1	14

Input	Output
2 5 2 1 5 4 2 3 3 5	0

H. Journey

Tashfin lives at (0, 0) in a 2D coordinate system. He is setting off a journey. But the further from home he is, the more scared he gets. So the entire journey he wants to stay as close to his home as possible.

Tashfin follows a specific set of instructions for his journey. There are 4 types of instructions, **U D L R**, which respectively change his position from (x, y) to $(x, y+1), (x, y-1), (x-1, y), (x+1, y)$.

Given a string S of length N, he follows S as his journey manual. Tashfin must perform all of the N moves to complete his journey. Before the start of his journey, you can rearrange the string S in any way you want.

There's one **condition** though. After each move, he can't go to the opposite direction in the next move. For example, if he follows **U** on some move, he can't have a **D** in the exact next move. Specifically, the rearranged string must not contain any of "**LR**", "**RL**", "**UD**", and "**DU**" as substring.

Tashfin is at (0, 0) at first. Your task is to rearrange the string S so that the maximum distance from home is minimized. More formally, you have to minimize the maximum $abs(x) + abs(y)$ over all positions (x,y) that he reaches on his journey. It might be that he can't perform all the N moves, report when it happens.

Note that, he doesn't have to return to his home after N moves. Also, you have to minimize his maximum distance over the N moves on his journey, not necessarily the distance of his final position.

Input

The first line contains a single integer T, the number of journeys X is going to take. Each of the next T lines contain an integer N and a string S, separated by space.

Constraints

S contains only the letters **'U', 'D', 'L'**, and **'R'**
Sum of N over all journeys

Output

For each journey, print the rearranged string. If there are multiple solutions, print any. If the journey is not possible, print "**Impossible**" without quotes.

Samples

Input	Output
4 2 DD 3 UDL 8 UUDDLLRR 3 UUD	DD ULD LURDDLUR Impossible