A. Plus One on the Subset

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Polycarp got an array of integers $a[1 \dots n]$ as a gift. Now he wants to perform a certain number of operations (possibly zero) so that all elements of the array become the same (that is, to become $a_1 = a_2 = \dots = a_n$).

- In one operation, he can take some indices in the array and increase the elements of the array at those indices by 1.

For example, let $a = [4, 2, 1, 6, 2]$. He can perform the following operation: select indices 1, 2, and 4 and increase elements of the array in those indices by 1. As a result, in one operation, he can get a new state of the array $a = [5, 3, 1, 7, 2]$.

What is the minimum number of operations it can take so that all elements of the array become equal to each other (that is, to become $a_1 = a_2 = \dots = a_n$)?

Input

The first line of the input contains a single integer t ($1 \le t \le 10^4$) — the number of test cases in the test.

The following are descriptions of the input test cases.

The first line of the description of each test case contains one integer n ($1 \le n \le 50$) — the array a.

The second line of the description of each test case contains n integers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le 10^9$) — elements of the array a.

Output

For each test case, print one integer — the minimum number of operations to make all elements of the array a equal.

Example

input

```
3
6
3 4 2 4 1 2
3
1000 1002 998
2
12 11
```

output

```
3
4
1
```

Note

First test case:

- $a = [3, 4, 2, 4, 1, 2]$ take a_3, a_5 and perform an operation plus one on them, as a result we get $a = [3, 4, 3, 4, 2, 2]$.
- $a = [3, 4, 3, 4, 2, 2]$ we take a_1, a_5, a_6 and perform an operation on them plus one, as a result we get $a = [4, 4, 3, 4, 3, 3]$.
- $a = [4, 4, 3, 4, 3, 3]$ we take a_3, a_5, a_6 and perform an operation on them plus one, as a result we get $a = [4, 4, 4, 4, 4, 4]$.

There are other sequences of 3 operations, after the application of which all elements become equal.

Second test case:

- $a = [1000, 1002, 998]$ 2 times we take a_1, a_3 and perform an operation plus one on them, as a result we get $a = [1002, 1002, 1000]$.
- $a = [1002, 1002, 1000]$ also take a_3 2 times and perform an operation plus one on it, as a result we get $a = [1002, 1002, 1002]$.

Third test case:

- $a = [12, 11]$ take a_2 and perform an operation plus one on it, as a result we get $a = [12, 12]$.

B. Make AP

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Polycarp has 3 positive integers a, b and c. He can perform the following operation **exactly once**.

- Choose a **positive** integer m and multiply **exactly one** of the integers a, b or c by m.

Can Polycarp make it so that after performing the operation, the sequence of three numbers a, b, c (**in this order**) forms an arithmetic progression? Note that you **cannot change** the order of a, b and c.

Formally, a sequence $x_1, x_2, \dots, x_n$ is called an arithmetic progression (AP) if there exists a number d (called "common difference") such that $x_{i+1} = x_i + d$ for all i from 1 to $n-1$. In this problem, $n = 3$.

For example, the following sequences are AP: $[5, 10, 15]$, $[3, 2, 1]$, $[1, 1, 1]$, and $[13, 10, 7]$. The following sequences are not AP: $[1, 2, 4]$, $[0, 1, 0]$ and $[1, 3, 2]$.

You need to answer t independent test cases.

Input

The first line contains the number t ($1 \le t \le 10^4$) — the number of test cases.

Each of the following t lines contains 3 integers a, b, c ($1 \le a, b, c \le 10^8$).

Output

For each test case print "YES" (without quotes) if Polycarp can choose a **positive** integer m and multiply **exactly one** of the integers a, b or c by m to make $[a, b, c]$ be an arithmetic progression. Print "NO" (without quotes) otherwise.

You can print YES and NO in any (upper or lower) case (for example, the strings yEs, yes, Yes and YES will be recognized as a positive answer).

Example

input

```
11
10 5 30
30 5 10
1 2 3
1 6 3
2 6 3
1 1 1
1 1 2
1 1 3
1 100000000 1
2 1 1
1 2 2
```

output

```
YES
YES
YES
YES
NO
YES
NO
YES
YES
NO
YES
```

Note

In the first and second test cases, you can choose the number $m = 4$ and multiply the second number ($b = 5$) by 4.

In the first test case the resulting sequence will be $[10, 20, 30]$. This is an AP with a difference $d = 10$.

In the second test case the resulting sequence will be $[30, 20, 10]$. This is an AP with a difference $d = -10$.

In the third test case, you can choose $m = 1$ and multiply any number by 1. The resulting sequence will be $[1, 2, 3]$. This is an AP with a difference $d = 1$.

In the fourth test case, you can choose $m = 9$ and multiply the first number ($a = 1$) by 9. The resulting sequence will be $[9, 6, 3]$. This is an AP with a difference $d = -3$.

In the fifth test case, it is impossible to make an AP.

C. Division by Two and Permutation

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given an array a consisting of n positive integers. You can perform operations on it.

In one operation you can replace any element of the array a_i with $\lfloor \frac{a_i}{2} \rfloor$, that is, by an integer part of dividing a_i by 2 (rounding down).

See if you can apply the operation some number of times (possible 0) to make the array a become a permutation of numbers from 1 to n —that is, so that it contains all numbers from 1 to n, each exactly once.

For example, if $a = [1, 8, 25, 2]$, $n = 4$, then the answer is yes. You could do the following:

1. Replace 8 with $\lfloor \frac{8}{2} \rfloor = 4$, then $a = [1, 4, 25, 2]$.
2. Replace 25 with $\lfloor \frac{25}{2} \rfloor = 12$, then $a = [1, 4, 12, 2]$.
3. Replace 12 with $\lfloor \frac{12}{2} \rfloor = 6$, then $a = [1, 4, 6, 2]$.
4. Replace 6 with $\lfloor \frac{6}{2} \rfloor = 3$, then $a = [1, 4, 3, 2]$.

Input

The first line of input data contains an integer t ($1 \le t \le 10^4$) —the number of test cases.

Each test case contains exactly two lines. The first one contains an integer n ($1 \le n \le 50$), the second one contains integers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le 10^9$).

Output

For each test case, output on a separate line:

- YES if you can make the array a become a permutation of numbers from 1 to n,
- NO otherwise.

You can output YES and NO in any case (for example, strings yEs, yes, Yes and YES will be recognized as a positive response).

Example

input

```
6
4
1 8 25 2
2
1 1
9
9 8 3 4 2 7 1 5 6
3
8 2 1
4
24 7 16 7
5
22 6 22 4 22
```

output

```
YES
NO
YES
NO
NO
YES
```

Note

The first test case is explained in the text of the problem statement.

In the second test case, it is not possible to get a permutation.

D. Palindromes Coloring

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You have a string s consisting of lowercase Latin alphabet letters.

You can color some letters in colors from 1 to k. It is not necessary to paint all the letters. But for each color, there must be a letter painted in that color.

Then you can swap any two symbols painted in the same color as many times as you want.

After that, k strings will be created, i-th of them will contain all the characters colored in the color i, written in the order of their sequence in the string s.

Your task is to color the characters of the string so that all the resulting k strings are palindromes, and the length of the shortest of these k strings is as **large** as possible.

Read the note for the first test case of the example if you need a clarification.

Recall that a string is a palindrome if it reads the same way both from left to right and from right to left. For example, the strings abacaba, cccc, z and dxd are palindromes, but the strings abab and aaabaa — are not.

Input

The first line of input data contains a single integer t ($1 \le t \le 10^4$) — the number of input data sets in the test.

The descriptions of the input data sets follow.

The first line of the description of each input data set contains two integers n and k ($1 \le k \le n \le 2 \cdot 10^5$) — the length of the string and the number of colors in which its letters can be painted. The second line of the description of each input data set contains a string s of length n consisting of lowercase letters of the Latin alphabet.

It is guaranteed that the sum of n over all test cases does not exceed $2 \cdot 10^5$.

Output

For each set of input data, output a single integer — the maximum length of the shortest palindrome string that can be obtained.

Example

input

```
10
8 2
bxyaxzay
6 3
aaaaaa
6 1
abcdef
6 6
abcdef
3 2
dxd
11 2
abcabcabcac
6 6
sipkic
7 2
eatoohd
3 1
llw
6 2
bfvfbv
```

output

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

Note

- In the first test case, s="bxyaxzay", $k = 2$. We use indices in the string from 1 to 8. The following coloring will work: **bxyaxzay** (the letter z remained uncolored). After painting:
  - swap two red characters (with the indices 1 and 4), we get **axybxzay**;
  - swap two blue characters (with the indices 5 and 8), we get **axybyzax**.

  Now, for each of the two colors we write out the corresponding characters from left to right, we get two strings **aba** and **xyyx**. Both of them are palindromes, the length of the shortest is 3. It can be shown that the greatest length of the shortest palindrome cannot be achieved.
- In the second set of input data, the following coloring is suitable: $[1, 1, 2, 2, 3, 3]$. There is no need to swap characters. Both received strings are equal to aa, they are palindromes and their length is 2.
- In the third set of input data, you can color any character and take it into a string.
- In the fourth set of input data, you can color the ith character in the color i.
- In the fifth set of input data can be colored in each of the colors of one character.
- In the sixth set of input data, the following coloring is suitable: $[1, 1, 1, 1, 1, 2, 2, 2, 2, 2, 0]$. Rearrange the characters so as to get the palindromes abcba and acbca.

E. Masha-forgetful

time limit per test: 3 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Masha meets a new friend and learns his phone number — s. She wants to remember it as soon as possible. The phone number — is a string of length m that consists of digits from 0 to 9. The phone number may start with `0`.

Masha already knows n phone numbers (all numbers have the same length m). It will be easier for her to remember a new number if the s is represented as segments of numbers she already knows. Each such segment must be of length **at least** 2, otherwise there will be too many segments and Masha will get confused.

For example, Masha needs to remember the number: $s =$ '12345678' and she already knows $n = 4$ numbers: '12340219', '20215601', '56782022', '12300678'. You can represent s as a 3 segment: '1234' of number one, '56' of number two, and '78' of number three. There are other ways to represent s.

Masha asks you for help, she asks you to break the string s into segments of length 2 or more of the numbers she already knows. If there are several possible answers, print **any** of them.

Input

The first line of input data contains an integer t ($1 \leq t \leq 10^4$) —the number of test cases.

Before each test case there is a blank line. Then there is a line containing integers n and m ($1 \leq n, m \leq 10^3$) —the number of phone numbers that Masha knows and the number of digits in each phone number. Then follow n line, i-th of which describes the i-th number that Masha knows. The next line contains the phone number of her new friend s.

Among the given $n + 1$ phones, there may be duplicates (identical phones).

It is guaranteed that the sum of $n \cdot m$ (n multiplied by m) values over all input test cases does not exceed 10^6.

Output

You need to print the answers to t test cases. The first line of the answer should contain one number k, corresponding to the number of segments into which you split the phone number s. Print -1 if you cannot get such a split.

If the answer is yes, then follow k lines containing triples of numbers l, r, i. Such triplets mean that the next $r - l + 1$ digits of number s are equal to a segment (substring) with boundaries $[l, r]$ of the phone under number i. Both the phones and the digits in them are numbered from 1. Note that $r - l + 1 \geq 2$ for all k lines.

Example

input

```
5

4 8
12340219
20215601
56782022
12300678
12345678

2 3
134
126
123

1 4
1210
1221

4 3
251
064
859
957
054

4 7
7968636
9486033
4614224
5454197
9482268
```

output

```
3
1 4 1
5 6 2
3 4 3
-1
2
1 2 1
2 3 1
-1
3
1 3 2
5 6 3
3 4 1
```

Note

The example from the statement.

In the second case, it is impossible to represent by segments of known numbers of length 2 or more.

In the third case, you can get the segments '12' and '21' from the first phone number.

F. Interacdive Problem

time limit per test: 1 second
memory limit per test: 256 megabytes
input: standard input
output: standard output

This problem is interactive.

We decided to play a game with you and guess the number x ($1 \le x < n$), where you know the number n.

You can make queries like this:

- `+ c`: this command assigns $x = x + c$ ($1 \le c < n$) and then returns you the value $\lfloor \frac{x}{n} \rfloor$ (x divide by n and round down).

You win if you guess the current number with no more than 10 queries.

Interaction

The interaction begins by reading an integer n ($2 < n \le 1000$), which is written in the input data on its own line.

Then you can make no more than 10 queries. To make a query, print on a separate line:

- `+ c`: this command will assign $x = x + c$ ($1 \le c < n$) and then print $\lfloor \frac{x}{n} \rfloor$ (divide x by n and round down) on a separate line.

Print the answer, like the queries, on a separate line. The answer doesn't count in number of queries. To output it, use the following format:

- `! x`: the current value of x.

After that, your program should exit.

You have to use a `flush` operation right after printing each line. For example, in C++ you should use the function `fflush(stdout)`, in Java — `System.out.flush()`, in Pascal — `flush(output)` and in Python — `sys.stdout.flush()`.

Note that the interactor is not responsive.

To make a hack, use the following format: a single line must contain two numbers x and n, separated by a space.

Examples

input

```
3

1
```

output

```
+ 1

! 3
```

input

```
5

0

0

1
```

output

```
+ 1

+ 1

+ 1

! 5
```

input

```
10

0

0

1

2
```

output

```

+ 2
+ 2

+ 3

+ 8

! 20
```

Note

In the first sample initially $x = 2$. After the first query $x = 3$, $\lfloor \frac{x}{n} \rfloor = 1$.

In the second sample also initially $x = 2$. After the first query $x = 3$, $\lfloor \frac{x}{n} \rfloor = 0$. After the second query $x = 4$, $\lfloor \frac{x}{n} \rfloor = 0$. After the third query $x = 5$, $\lfloor \frac{x}{n} \rfloor = 1$.

G. MinOr Tree

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Recently, Vlad has been carried away by spanning trees, so his friends, without hesitation, gave him a connected weighted undirected graph of n vertices and m edges for his birthday.

Vlad defined the *ority* of a spanning tree as the bitwise OR of all its weights, and now he is interested in what is the minimum possible *ority* that can be achieved by choosing a certain spanning tree. A spanning tree is a connected subgraph of a given graph that does not contain cycles.

In other words, you want to keep $n-1$ edges so that the graph remains connected and the bitwise OR weights of the edges are as small as possible. You have to find the minimum bitwise OR itself.

Input

The first line of the input contains an integer t ($1 \le t \le 10^4$) — the number of test cases in the input.

An empty line is written in front of each test case.

This is followed by two numbers n and m ($3 \le n \le 2 \cdot 10^5, n-1 \le m \le 2 \cdot 10^5$) — the number of vertices and edges of the graph, respectively.

The next m lines contain the description of the edges. Line i contains three numbers v_i, u_i and w_i ($1 \le v_i, u_i \le n$, $1 \le w_i \le 10^9$, $v_i \ne u_i$) — the vertices that the edge connects and its weight.

It is guaranteed that the sum m and the sum n over all test cases does not exceed $2 \cdot 10^5$ and each test case contains a connected graph.

Output

Print t lines, each of which contains the answer to the corresponding set of input data — the minimum possible spanning tree *ority*.

Example

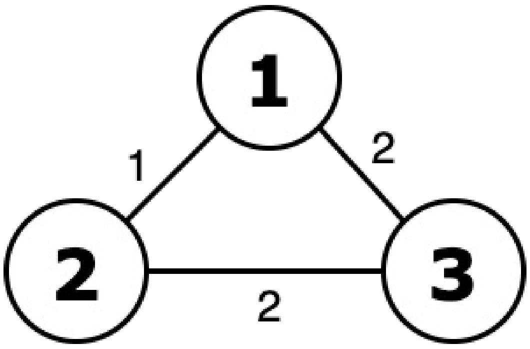
input

```
3

3 3
1 2 1
2 3 2
1 3 2

5 7
4 2 7
2 5 8
3 4 2
3 2 1
2 4 2
4 1 2
1 2 2

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

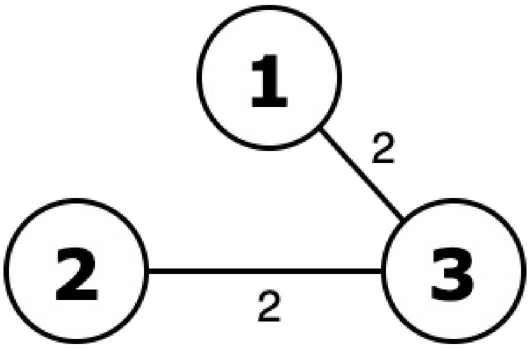
output

```
2
10
3
```

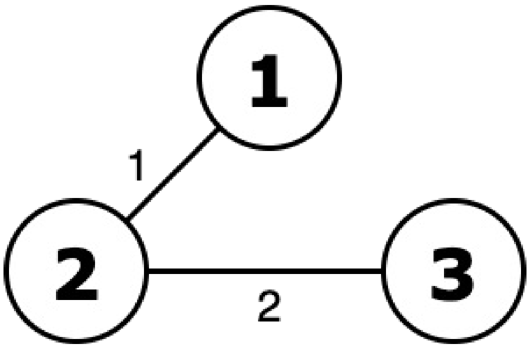
Note



Graph from the first test case.



Ority of this tree equals to 2 `or` $2 = 2$ and it's minimal.



Without excluding edge with weight 1 *ority* is 1 `or` $2 = 3$.