CODEFORCES β
Sponsored by Telegram

Codeforces Round #776 (Div. 3)

A. Deletions of Two Adjacent Letters

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

The string s is given, the string length is **odd** number. The string consists of lowercase letters of the Latin alphabet.

As long as the string length is greater than 1, the following operation can be performed on it: select any two adjacent letters in the string s and delete them from the string. For example, from the string "`lemma`" in one operation, you can get any of the four strings: "`mma`", "`lma`", "`lea`" or "`lem`" In particular, in one operation, the length of the string reduces by 2.

Formally, let the string s have the form $s = s_1 s_2 \dots s_n$ $(n > 1)$. During one operation, you choose an arbitrary index i $(1 \le i < n)$ and replace $s = s_1 s_2 \dots s_{i-1} s_{i+2} \dots s_n$.

For the given string s and the letter c, determine whether it is possible to make such a sequence of operations that in the end the equality $s = c$ will be true? In other words, is there such a sequence of operations that the process will end with a string of length 1, which consists of the letter c?

Input

The first line of input data contains an integer t $(1 \le t \le 10^3)$ — the number of input test cases.

The descriptions of the t cases follow. Each test case is represented by two lines:

- string s, which has an odd length from 1 to 49 inclusive and consists of lowercase letters of the Latin alphabet;
- is a string containing one letter c, where c is a lowercase letter of the Latin alphabet.

Output

For each test case in a separate line output:

- `YES`, if the string s can be converted so that $s = c$ is true;
- `NO` otherwise.

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

Example

input
```
5
abcde
c
abcde
b
x
y
aaaaaaaaaaaaaaa
a
contest
t
```
output
```
YES
NO
NO
YES
YES
```

Note

In the first test case, s="`abcde`". You need to get s="`c`". For the first operation, delete the first two letters, we get s="`cde`". In the second operation, we delete the last two letters, so we get the expected value of s="`c`".

In the third test case, s="`x`", it is required to get s="`y`". Obviously, this cannot be done.

B. DIV + MOD

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

Not so long ago, Vlad came up with an interesting function:

- $f_a(x) = \left\lfloor \frac{x}{a} \right\rfloor + x \bmod a$, where $\left\lfloor \frac{x}{a} \right\rfloor$ is $\frac{x}{a}$, rounded **down**, $x \bmod a$ — the remainder of the integer division of x by a.

For example, with $a = 3$ and $x = 11$, the value $f_3(11) = \left\lfloor \frac{11}{3} \right\rfloor + 11 \bmod 3 = 3 + 2 = 5$.

The number a is fixed and known to Vlad. Help Vlad find the maximum value of $f_a(x)$ if x can take any integer value from l to r inclusive ($l \le x \le r$).

Input

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

This is followed by t lines, each of which contains three integers l_i, r_i and a_i $(1 \le l_i \le r_i \le 10^9, 1 \le a_i \le 10^9)$ — the left and right boundaries of the segment and the fixed value of a.

Output

For each test case, output one number on a separate line — the maximum value of the function on a given segment for a given a.

Example

input
```
5
1 4 3
5 8 4
6 10 6
1 1000000000 1000000000
10 12 8
```
output
```
2
4
5
999999999
5
```

Note

In the first sample:

- $f_3(1) = \left\lfloor \frac{1}{3} \right\rfloor + 1 \bmod 3 = 0 + 1 = 1$,
- $f_3(2) = \left\lfloor \frac{2}{3} \right\rfloor + 2 \bmod 3 = 0 + 2 = 2$,
- $f_3(3) = \left\lfloor \frac{3}{3} \right\rfloor + 3 \bmod 3 = 1 + 0 = 1$,
- $f_3(4) = \left\lfloor \frac{4}{3} \right\rfloor + 4 \bmod 3 = 1 + 1 = 2$

As an answer, obviously, $f_3(2)$ and $f_3(4)$ are suitable.

C. Weight of the System of Nested Segments

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

On the number line there are m points, i-th of which has integer coordinate x_i and integer weight w_i. The coordinates of all points are different, and the points are numbered from 1 to m.

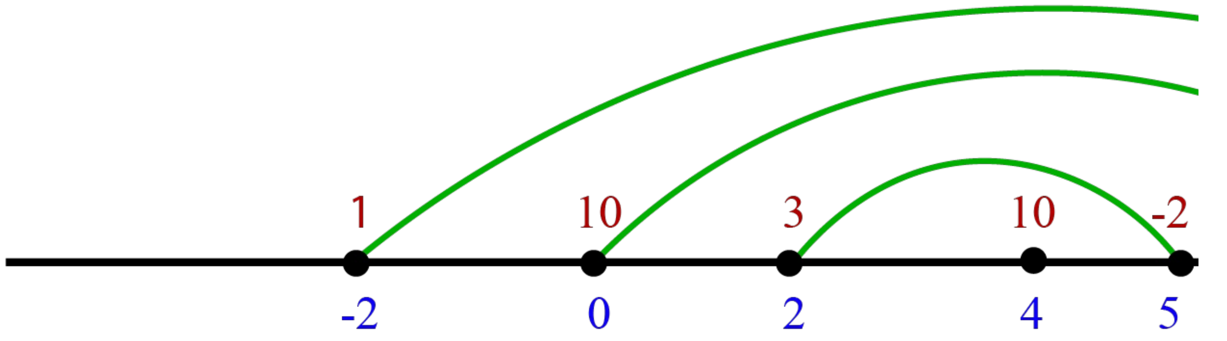
A sequence of n segments $[l_1, r_1], [l_2, r_2], \dots, [l_n, r_n]$ is called *system of nested segments* if for each pair i, j ($1 \le i < j \le n$) the condition $l_i < l_j < r_j < r_i$ is satisfied. In other words, the second segment is strictly inside the first one, the third segment is strictly inside the second one, and so on.

For a given number n, find a system of nested segments such that:

- both ends of each segment are one of m given points;
- the sum of the weights $2 \cdot n$ of the points used as ends of the segments is **minimal**.

For example, let $m = 8$. The given points are marked in the picture, their weights are marked in red, their coordinates are marked in blue. Make a system of three nested segments:

- weight of the first segment: $1 + 1 = 2$
- weight of the second segment: $10 + (-1) = 9$
- weight of the third segment: $3 + (-2) = 1$
- sum of the weights of all the segments in the system: $2 + 9 + 1 = 12$



System of three nested segments

Input

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

An empty line is written before each test case.

The first line of each test case contains two positive integers n ($1 \le n \le 10^5$) and m ($2 \cdot n \le m \le 2 \cdot 10^5$).

The next m lines contain pairs of integers x_i ($-10^9 \le x_i \le 10^9$) and w_i ($-10^4 \le w_i \le 10^4$) — coordinate and weight of point number i ($1 \le i \le m$) respectively. All x_i are different.

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

Output

For each test case, output $n + 1$ lines: in the first of them, output the weight of the composed system, and in the next n lines output exactly two numbers — the indices of the points which are the endpoints of the i-th segment ($1 \le i \le n$). The order in which you output the endpoints of a segment is not important — you can output the index of the left endpoint first and then the number of the right endpoint, or the other way around.

If there are several ways to make a system of nested segments with minimal weight, output any of them.

Example

input

```
3

3 8
0 10
-2 1
4 10
11 20
7 -1
9 1
2 3
5 -2

3 6
-1 2
1 3
3 -1
2 4
4 0
8 2

2 5
5 -1
3 -2
1 0
-2 0
-5 -3
```

output

```
12
2 6
5 1
7 8

10
1 6
5 2
3 4

-6
5 1
4 2
```

Note

The first test case coincides with the example from the condition. It can be shown that the weight of the composed system is minimal.

The second test case has only 6 points, so you need to use each of them to compose 3 segments.

D. Twist the Permutation

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

Petya got an array a of numbers from 1 to n, where $a[i] = i$.

He performed n operations sequentially. In the end, he received a new state of the a array.

At the i-th operation, Petya chose the first i elements of the array and cyclically shifted them to the right an arbitrary number of times (elements with indexes $i+1$ and more remain in their places). One cyclic shift to the right is such a transformation that the array $a = [a_1, a_2, \ldots, a_n]$ becomes equal to the array $a = [a_i, a_1, a_2, \ldots, a_{i-2}, a_{i-1}, a_{i+1}, a_{i+2}, \ldots, a_n]$.

For example, if $a = [5, 4, 2, 1, 3]$ and $i = 3$ (that is, this is the third operation), then as a result of this operation, he could get any of these three arrays:

- $a = [5, 4, 2, 1, 3]$ (makes 0 cyclic shifts, or any number that is divisible by 3);
- $a = [2, 5, 4, 1, 3]$ (makes 1 cyclic shift, or any number that has a remainder of 1 when divided by 3);
- $a = [4, 2, 5, 1, 3]$ (makes 2 cyclic shifts, or any number that has a remainder of 2 when divided by 3).

Let's look at an example. Let $n = 6$, i.e. initially $a = [1, 2, 3, 4, 5, 6]$. A possible scenario is described below.

- $i = 1$: no matter how many cyclic shifts Petya makes, the array a does not change.
- $i = 2$: let's say Petya decided to make a 1 cyclic shift, then the array will look like $a = [\mathbf{2}, \mathbf{1}, 3, 4, 5, 6]$.
- $i = 3$: let's say Petya decided to make 1 cyclic shift, then the array will look like $a = [\mathbf{3}, \mathbf{2}, \mathbf{1}, 4, 5, 6]$.
- $i = 4$: let's say Petya decided to make 2 cyclic shifts, the original array will look like $a = [\mathbf{1}, \mathbf{4}, \mathbf{3}, \mathbf{2}, 5, 6]$.
- $i = 5$: let's say Petya decided to make 0 cyclic shifts, then the array won't change.
- $i = 6$: let's say Petya decided to make 4 cyclic shifts, the array will look like $a = [\mathbf{3}, \mathbf{2}, \mathbf{5}, \mathbf{6}, \mathbf{1}, \mathbf{4}]$.

You are given a final array state a after all n operations. Determine if there is a way to perform the operation that produces this result. In this case, if an answer exists, print the numbers of cyclical shifts that occurred during each of the n operations.

Input

The first line of the input contains an integer t ($1 \le t \le 500$) — the number of test cases in the test.

The descriptions of the test cases follow.

The first line of the description of each test case contains one integer n ($2 \le n \le 2 \cdot 10^3$) — the length of the array a.

The next line contains the final state of the array a: n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le n$) are written. All a_i are distinct.

It is guaranteed that the sum of n values over all test cases does not exceed $2 \cdot 10^3$.

Output

For each test case, print the answer on a separate line.

Print -1 if the given final value a cannot be obtained by performing an arbitrary number of cyclic shifts on each operation. Otherwise, print n non-negative integers $d_1, d_2, \ldots, d_n$ ($d_i \ge 0$), where d_i means that during the i-th operation the first i elements of the array were cyclic shifted to the right d_i times.

If there are several possible answers, print the one where the total number of shifts is minimal (that is, the sum of d_i values is the smallest). If there are several such answers, print any of them.

Example

input

```
3
6
3 2 5 6 1 4
3
3 1 2
8
5 8 1 3 2 6 4 7
```

output

```
0 1 1 2 0 4
0 0 1
0 1 2 0 2 5 6 2
```

Note

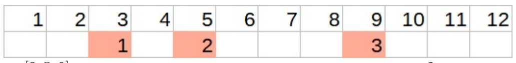
The first test case matches the example from the statement.

The second set of input data is simple. Note that the answer $[3, 2, 1]$ also gives the same permutation, but since the total number of shifts $3 + 2 + 1$ is greater than $0 + 0 + 1$, this answer is not correct.

E. Rescheduling the Exam

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

Now Dmitry has a session, and he has to pass n exams. The session starts on day 1 and lasts d days. The ith exam will take place on the day of a_i ($1 \le a_i \le d$), all a_i — are different.



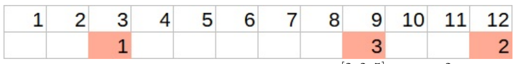
1	2	3	4	5	6	7	8	9	10	11	12
		1		2				3			

Sample, where $n = 3$, $d = 12$, $a = [3, 5, 9]$. Orange — exam days. Before the first exam Dmitry will rest 2 days, before the second he will rest 1 day and before the third he will rest 3 days.

For the session schedule, Dmitry considers a special value μ — the smallest of the rest times before the exam for all exams. For example, for the image above, $\mu = 1$. In other words, for the schedule, he counts exactly n numbers — how many days he rests between the exam $i-1$ and i (for $i = 0$ between the start of the session and the exam i). Then it finds μ — the minimum among these n numbers.

Dmitry believes that he can improve the schedule of the session. He may ask to change the date of one exam (change one arbitrary value of a_i). Help him change the date so that all a_i remain different, and the value of μ is as large as possible.

For example, for the schedule above, it is most advantageous for Dmitry to move the second exam to the very end of the session. The new schedule will take the form:



1	2	3	4	5	6	7	8	9	10	11	12
		1						3			2

Now the rest periods before exams are equal to $[2, 2, 5]$. So, $\mu = 2$.

Dmitry can leave the proposed schedule unchanged (if there is no way to move one exam so that it will lead to an improvement in the situation).

Input

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

An empty line is written in the test before each case.

The first line of each test case contains two integers n and d ($2 \le n \le 2 \cdot 10^5, 1 \le d \le 10^9$) — the number of exams and the length of the session, respectively.

The second line of each test case contains n integers a_i ($1 \le a_i \le d, a_i < a_{i+1}$), where the i-th number means the date of the i-th exam.

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

Output

For each test case, output the maximum possible value of μ if Dmitry can move any one exam to an arbitrary day. All values of a_i should remain distinct.

Example

input

```
9

3 12
3 5 9
```

```
2 5
1 5

2 100
1 2

5 15
3 6 9 12 15

3 1000000000
1 400000000 500000000

2 10
3 4

2 2
1 2

4 15
6 11 12 13

2 20
17 20
```

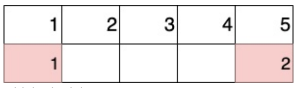
output

```
2
1
1
2
99999999
3
0
1
9
```

Note

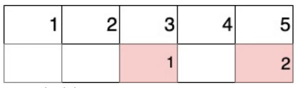
The first sample is parsed in statement.

One of the optimal schedule changes for the second sample:



1	2	3	4	5
1				2

Initial schedule.



1	2	3	4	5
		1		2

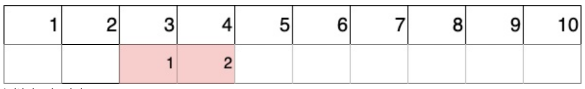
New schedule.

In the third sample, we need to move the exam from day 1 to any day from 4 to 100.

In the fourth sample, any change in the schedule will only reduce μ, so the schedule should be left as it is.

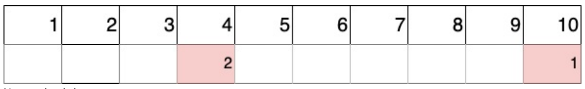
In the fifth sample, we need to move the exam from day 1 to any day from 100000000 to 300000000.

One of the optimal schedule changes for the sixth sample:



1	2	3	4	5	6	7	8	9	10
		1	2						

Initial schedule.



1	2	3	4	5	6	7	8	9	10
			2						1

New schedule.

In the seventh sample, every day is exam day, and it is impossible to rearrange the schedule.

F. Vitaly and Advanced Useless Algorithms

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

Vitaly enrolled in the course Advanced Useless Algorithms. The course consists of n tasks. Vitaly calculated that he has a_i hours to do the task i from the day he enrolled in the course. That is, the deadline before the i-th task is a_i hours. The array a is sorted in ascending order, in other words, the job numbers correspond to the order in which the assignments are turned in.

Vitaly does everything conscientiously, so he wants to complete **each** task by 100 percent, **or more**. Initially, his completion rate for each task is 0 percent.

Vitaly has m training options, each option can be used **not more than** once. The ith option is characterized by three integers: e_i, t_i and p_i. If Vitaly uses the ith option, then after t_i hours (from the current moment) he will increase the progress of the task e_i by p_i percent.

For example, let Vitaly have 3 of tasks to complete. Let the array a have the form: $a = [5, 7, 8]$. Suppose Vitaly has 5 of options: $[e_1 = 1, t_1 = 1, p_1 = 30]$, $[e_2 = 2, t_2 = 3, p_2 = 50]$, $[e_3 = 2, t_3 = 3, p_3 = 100]$, $[e_4 = 1, t_4 = 1, p_4 = 80]$, $[e_5 = 3, t_5 = 3, p_5 = 100]$.

Then, if Vitaly prepares in the following way, he will be able to complete everything in time:

- Vitaly chooses the 4-th option. Then in 1 hour, he will complete the 1-st task at 80 percent. He still has 4 hours left before the deadline for the 1-st task.
- Vitaly chooses the 3-rd option. Then in 3 hours, he will complete the 2-nd task in its entirety. He has another 1 hour left before the deadline for the 1-st task and 4 hours left before the deadline for the 3-rd task.
- Vitaly chooses the 1-st option. Then after 1 hour, he will complete the 1-st task for 110 percent, which means that he will complete the 1-st task just in time for the deadline.
- Vitaly chooses the 5-th option. He will complete the 3-rd task for 2 hours, and after another 1 hour, Vitaly will complete the 3-rd task in its entirety.

Thus, Vitaly has managed to complete the course completely and on time, using the 4 options.

Help Vitaly — print the options for Vitaly to complete the tasks in the correct order. Please note: each option can be used **not more than** once. If there are several possible answers, it is allowed to output any of them.

Input

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

The descriptions of the input test case follow.

The first line of each test case description contains two integers n and m ($1 \le n, m \le 10^5$) —the number of jobs and the number of training options, respectively.

The next line contains n numbers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le 10^9$) — the time before the deadline of job i. The array values — are non-decreasing, that is $a_1 \le a_2 \le \dots \le a_n$.

The following m lines contain triples of numbers e_i, t_i, p_i ($1 \le e_i \le n$, $1 \le t_i \le 10^9$, $1 \le p_i \le 100$) — if Vitaly chooses this option, then after t_i hours he will increase the progress of the task e_i by p_i percent. The options are numbered from 1 to m in order in the input data.

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

Output

For each test case, print on the first line the number k, meaning that for k of options, Vitaly will be able to complete each task by 100 percent or more on time. The options should not be repeated. Or print -1 if Vitaly is unable to complete all tasks in time.

If there is an answer, on the next line print k of different integers from 1 to m — the numbers of the options in the order you want. If there is more than one answer, it is allowed to print **any** of them.

Examples

input

```
5
3 5
5 7 8
1 1 30
2 3 50
2 3 100
1 1 80
3 3 100
1 5
51
1 36 91
1 8 40
1 42 83
1 3 45
1 13 40
2 9
9 20
2 8 64
2 7 64
1 20 56
2 8 76
2 20 48
1 2 89
1 3 38
2 18 66
1 7 51
3 2
7 18 33
1 5 80
3 4 37
2 5
569452312 703565975
1 928391659 66
1 915310 82
2 87017081 92
1 415310 54
2 567745964 82
```

output

```
4
1 4 3 5
3
2 4 5
4
6 7 1 2
-1
4
2 4 3 5
```

input

```
3
3 9
20 31 40
1 9 64
3 17 100
3 9 59
3 18 57
3 20 49
2 20 82
2 14 95
1 8 75
2 16 67
2 6
20 36
2 2 66
2 20 93
1 3 46
1 10 64
2 8 49
2 18 40
1 1
1000000000
1 1000000000 100
```

output

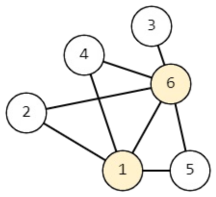
```
-1
4
3 4 1 5
1
1
```

G. Counting Shortcuts

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

Given an undirected connected graph with n vertices and m edges. The graph contains no loops (edges from a vertex to itself) and multiple edges (i.e. no more than one edge between each pair of vertices). The vertices of the graph are numbered from 1 to n.

Find the number of paths from a vertex s to t whose length differs from the shortest path from s to t by no more than 1. It is necessary to consider all suitable paths, even if they pass through the same vertex or edge more than once (i.e. they are not simple).



Graph consisting of 6 of vertices and 8 of edges

For example, let $n = 6$, $m = 8$, $s = 6$ and $t = 1$, and let the graph look like the figure above. Then the length of the shortest path from s to t is 1. Consider all paths whose length is at most $1 + 1 = 2$.

- $6 \to 1$. The length of the path is 1.
- $6 \to 4 \to 1$. Path length is 2.
- $6 \to 2 \to 1$. Path length is 2.
- $6 \to 5 \to 1$. Path length is 2.

There is a total of 4 of matching paths.

Input

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

Before each test case, there is a blank line.

The first line of test case contains two numbers n, m ($2 \le n \le 2 \cdot 10^5$, $1 \le m \le 2 \cdot 10^5$) —the number of vertices and edges in the graph.

The second line contains two numbers s and t ($1 \le s, t \le n$, $s \ne t$) —the numbers of the start and end vertices of the path.

The following m lines contain descriptions of edges: the ith line contains two integers u_i, v_i $(1 \leq u_i, v_i \leq n)$ — the numbers of vertices that connect the ith edge. It is guaranteed that the graph is connected and does not contain loops and multiple edges.

It is guaranteed that the sum of values n on all test cases of input data does not exceed $2 \cdot 10^5$. Similarly, it is guaranteed that the sum of values m on all test cases of input data does not exceed $2 \cdot 10^5$.

Output

For each test case, output a single number — the number of paths from s to t such that their length differs from the length of the shortest path by no more than 1.

Since this number may be too large, output it modulo $10^9 + 7$.

Example

input

```
4

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

6 8
6 1
1 4
1 6
1 5
1 2
5 6
4 6
6 3
2 6

5 6
1 3
3 5
5 4
3 1
4 2
2 1
1 4

8 18
5 1
2 1
3 1
4 2
5 2
6 5
7 3
8 4
6 4
8 7
1 4
4 7
1 6
6 7
3 8
8 5
4 5
4 3
8 2
```

output

```
2
4
1
11
```

Codeforces (c) Copyright 2010-2022 Mike Mirzayanov
The only programming contests Web 2.0 platform