Codeforces Round #481 (Div. 3)

A. Remove Duplicates

1 second, 256 megabytes

Petya has an array a consisting of n integers. He wants to remove duplicate (equal) elements.

Petya wants to leave only the rightmost entry (occurrence) for each element of the array. The relative order of the remaining unique elements should not be changed.

Input
The first line contains a single integer n ($1 \le n \le 50$) — the number of elements in Petya's array.

The following line contains a sequence $a_1, a_2, \dots, a_n$ ($1 \le a_i \le 1\,000$) — the Petya's array.

Output
In the first line print integer x — the number of elements which will be left in Petya's array after he removed the duplicates.

In the second line print x integers separated with a space — Petya's array after he removed the duplicates. For each unique element only the rightmost entry should be left.

input
```
6
1 5 5 1 6 1
```
output
```
3
5 6 1
```

input
```
5
2 4 2 4 4
```
output
```
2
2 4
```

input
```
5
6 6 6 6 6
```
output
```
1
6
```

In the first example you should remove two integers 1, which are in the positions 1 and 4. Also you should remove the integer 5, which is in the position 2.

In the second example you should remove integer 2, which is in the position 1, and two integers 4, which are in the positions 2 and 4.

In the third example you should remove four integers 6, which are in the positions 1, 2, 3 and 4.

B. File Name

1 second, 256 megabytes

You can not just take the file and send it. When Polycarp trying to send a file in the social network "Codehorses", he encountered an unexpected problem. If the name of the file contains three or more "x" (lowercase Latin letters "x") in a row, the system considers that the file content does not correspond to the social network topic. In this case, the file is not sent and an error message is displayed.

Determine the minimum number of characters to remove from the file name so after that the name does not contain "xxx" as a substring. Print 0 if the file name does not initially contain a forbidden substring "xxx".

You can delete characters in arbitrary positions (not necessarily consecutive). If you delete a character, then the length of a string is reduced by 1. For example, if you delete the character in the position 2 from the string "exxxii", then the resulting string is "exxii".

Input
The first line contains integer n ($3 \le n \le 100$) — the length of the file name.

The second line contains a string of length n consisting of lowercase Latin letters only — the file name.

Output
Print the minimum number of characters to remove from the file name so after that the name does not contain "xxx" as a substring. If initially the file name dost not contain a forbidden substring "xxx", print 0.

input
```
6
xxxiii
```
output
```
1
```

input
```
5
xxoxx
```
output
```
0
```

input
```
10
xxxxxxxxxx
```
output
```
8
```

In the first example Polycarp tried to send a file with name contains number 33, written in Roman numerals. But he can not just send the file, because it name contains three letters "x" in a row. To send the file he needs to remove any one of this letters.

C. Letters

4 seconds, 256 megabytes

There are n dormitories in Berland State University, they are numbered with integers from 1 to n. Each dormitory consists of rooms, there are a_i rooms in i-th dormitory. The rooms in i-th dormitory are numbered from 1 to a_i.

A postman delivers letters. Sometimes there is no specific dormitory and room number in it on an envelope. Instead of it only a room number among all rooms of all n dormitories is written on an envelope. In this case, assume that all the rooms are numbered from 1 to $a_1 + a_2 + \dots + a_n$ and the rooms of the first dormitory go first, the rooms of the second dormitory go after them and so on.

For example, in case $n = 2$, $a_1 = 3$ and $a_2 = 5$ an envelope can have any integer from 1 to 8 written on it. If the number 7 is written on an envelope, it means that the letter should be delivered to the room number 4 of the second dormitory.

For each of m letters by the room number among all n dormitories, determine the particular dormitory and the room number in a dormitory where this letter should be delivered.

Input
The first line contains two integers n and m ($1 \le n, m \le 2 \cdot 10^5$) — the number of dormitories and the number of letters.

The second line contains a sequence $a_1, a_2, \dots, a_n$ $(1 \le a_i \le 10^{10})$, where a_i equals to the number of rooms in the i-th dormitory. The third line contains a sequence $b_1, b_2, \dots, b_m$ $(1 \le b_j \le a_1 + a_2 + \dots + a_n)$, where b_j equals to the room number (among all rooms of all dormitories) for the j-th letter. All b_j are given in **increasing** order.

Output

Print m lines. For each letter print two integers f and k — the dormitory number f $(1 \le f \le n)$ and the room number k in this dormitory $(1 \le k \le a_f)$ to deliver the letter.

input
```
3 6
10 15 12
1 9 12 23 26 37
```
output
```
1 1
1 9
2 2
2 13
3 1
3 12
```

input
```
2 3
5 10000000000
5 6 9999999999
```
output
```
1 5
2 1
2 9999999994
```

In the first example letters should be delivered in the following order:

- the first letter in room 1 of the first dormitory
- the second letter in room 9 of the first dormitory
- the third letter in room 2 of the second dormitory
- the fourth letter in room 13 of the second dormitory
- the fifth letter in room 1 of the third dormitory
- the sixth letter in room 12 of the third dormitory

D. Almost Arithmetic Progression

1 second, 256 megabytes

Polycarp likes arithmetic progressions. A sequence $[a_1, a_2, \dots, a_n]$ is called an arithmetic progression if for each i $(1 \le i < n)$ the value $a_{i+1} - a_i$ is the same. For example, the sequences $[42]$, $[5, 5, 5]$, $[2, 11, 20, 29]$ and $[3, 2, 1, 0]$ are arithmetic progressions, but $[1, 0, 1]$, $[1, 3, 9]$ and $[2, 3, 1]$ are not.

It follows from the definition that any sequence of length one or two is an arithmetic progression.

Polycarp found some sequence of positive integers $[b_1, b_2, \dots, b_n]$. He agrees to change each element by at most one. In the other words, for each element there are exactly three options: an element can be decreased by 1, an element can be increased by 1, an element can be left unchanged.

Determine a minimum possible number of elements in b which can be changed (by exactly one), so that the sequence b becomes an arithmetic progression, or report that it is impossible.

It is possible that the resulting sequence contains element equals 0.

Input

The first line contains a single integer n $(1 \le n \le 100\,000)$ — the number of elements in b.

The second line contains a sequence $b_1, b_2, \dots, b_n$ $(1 \le b_i \le 10^9)$.

Output

If it is impossible to make an arithmetic progression with described operations, print `-1`. In the other case, print non-negative integer — the minimum number of elements to change to make the given sequence becomes an arithmetic progression. The only allowed operation is to add/to subtract one from an element (can't use operation twice to the same position).

input
```
4
24 21 14 10
```
output
```
3
```

input
```
2
500 500
```
output
```
0
```

input
```
3
14 5 1
```
output
```
-1
```

input
```
5
1 3 6 9 12
```
output
```
1
```

In the first example Polycarp should increase the first number on 1, decrease the second number on 1, increase the third number on 1, and the fourth number should left unchanged. So, after Polycarp changed three elements by one, his sequence became equals to $[25, 20, 15, 10]$, which is an arithmetic progression.

In the second example Polycarp should not change anything, because his sequence is an arithmetic progression.

In the third example it is impossible to make an arithmetic progression.

In the fourth example Polycarp should change only the first element, he should decrease it on one. After that his sequence will looks like $[0, 3, 6, 9, 12]$, which is an arithmetic progression.

E. Bus Video System

1 second, 256 megabytes

The busses in Berland are equipped with a video surveillance system. The system records information about changes in the number of passengers in a bus after stops.

If x is the number of passengers in a bus just before the current bus stop and y is the number of passengers in the bus just after current bus stop, the system records the number $y - x$. So the system records show how number of passengers changed.

The test run was made for single bus and n bus stops. Thus, the system recorded the sequence of integers $a_1, a_2, \dots, a_n$ (exactly one number for each bus stop), where a_i is the record for the bus stop i. The bus stops are numbered from 1 to n in chronological order.

Determine the number of possible ways how many people could be in the bus before the first bus stop, if the bus has a capacity equals to w (that is, at any time in the bus there should be from 0 to w passengers inclusive).

Input

The first line contains two integers n and w $(1 \le n \le 1\,000, 1 \le w \le 10^9)$ — the number of bus stops and the capacity of the bus.

The second line contains a sequence $a_1, a_2, \dots, a_n$ $(-10^6 \le a_i \le 10^6)$, where a_i equals to the number, which has been recorded by the video system after the i-th bus stop.

Output

Print the number of possible ways how many people could be in the bus before the first bus stop, if the bus has a capacity equals to w. If the situation is contradictory (i.e. for any initial number of passengers there will be a contradiction), print `0`.

input
```
3 5
2 1 -3
```
output
```
3
```

input
```
2 4
-1 1
```
output
```
4
```

input
```
4 10
2 4 1 2
```
output
```
2
```

In the first example initially in the bus could be 0, 1 or 2 passengers.

In the second example initially in the bus could be 1, 2, 3 or 4 passengers.

In the third example initially in the bus could be 0 or 1 passenger.

F. Mentors

3 seconds, 256 megabytes

In BerSoft n programmers work, the programmer i is characterized by a skill r_i.

A programmer a can be a mentor of a programmer b if and only if the skill of the programmer a is strictly greater than the skill of the programmer b $(r_a > r_b)$ and programmers a and b are not in a quarrel.

You are given the skills of each programmers and a list of k pairs of the programmers, which are in a quarrel (pairs are unordered). For each programmer i, find the number of programmers, for which the programmer i can be a mentor.

Input

The first line contains two integers n and k $(2 \le n \le 2 \cdot 10^5$, $0 \le k \le \min(2 \cdot 10^5, \frac{n \cdot (n-1)}{2}))$ — total number of programmers and number of pairs of programmers which are in a quarrel.

The second line contains a sequence of integers $r_1, r_2, \dots, r_n$ $(1 \le r_i \le 10^9)$, where r_i equals to the skill of the i-th programmer.

Each of the following k lines contains two distinct integers x, y $(1 \le x, y \le n, x \ne y)$ — pair of programmers in a quarrel. The pairs are unordered, it means that if x is in a quarrel with y then y is in a quarrel with x. Guaranteed, that for each pair (x, y) there are no other pairs (x, y) and (y, x) in the input.

Output

Print n integers, the i-th number should be equal to the number of programmers, for which the i-th programmer can be a mentor. Programmers are numbered in the same order that their skills are given in the input.

input
```
4 2
10 4 10 15
1 2
4 3
```
output
```
0 0 1 2
```

input
```
10 4
5 4 1 5 4 3 7 1 2 5
4 6
2 1
10 8
3 5
```
output
```
5 4 0 5 3 3 9 0 2 5
```

In the first example, the first programmer can not be mentor of any other (because only the second programmer has a skill, lower than first programmer skill, but they are in a quarrel). The second programmer can not be mentor of any other programmer, because his skill is minimal among others. The third programmer can be a mentor of the second programmer. The fourth programmer can be a mentor of the first and of the second programmers. He can not be a mentor of the third programmer, because they are in a quarrel.

G. Petya's Exams

1 second, 256 megabytes

Petya studies at university. The current academic year finishes with n special days. Petya needs to pass m exams in those special days. The special days in this problem are numbered from 1 to n.

There are three values about each exam:

- s_i — the day, when questions for the i-th exam will be published,
- d_i — the day of the i-th exam $(s_i < d_i)$,
- c_i — number of days Petya needs to prepare for the i-th exam. For the i-th exam Petya should prepare in days between s_i and $d_i - 1$, inclusive.

There are three types of activities for Petya in each day: to spend a day doing nothing (taking a rest), to spend a day passing exactly one exam or to spend a day preparing for exactly one exam. So he can't pass/prepare for multiple exams in a day. He can't mix his activities in a day. If he is preparing for the i-th exam in day j, then $s_i \le j < d_i$.

It is allowed to have breaks in a preparation to an exam and to alternate preparations for different exams in consecutive days. So preparation for an exam is not required to be done in consecutive days.

Find the schedule for Petya to prepare for all exams and pass them, or report that it is impossible.

Input

The first line contains two integers n and m $(2 \le n \le 100, 1 \le m \le n)$ — the number of days and the number of exams.

Each of the following m lines contains three integers s_i, d_i, c_i $(1 \le s_i < d_i \le n, 1 \le c_i \le n)$ — the day, when questions for the i-th exam will be given, the day of the i-th exam, number of days Petya needs to prepare for the i-th exam.

Guaranteed, that all the exams will be in different days. Questions for different exams can be given in the same day. It is possible that, in the day of some exam, the questions for other exams are given.

Output

If Petya can not prepare and pass all the exams, print `-1`. In case of positive answer, print n integers, where the j-th number is:

- $(m + 1)$, if the j-th day is a day of some exam (recall that in each day no more than one exam is conducted),

- zero, if in the j-th day Petya will have a rest,
- i $(1 \le i \le m)$, if Petya will prepare for the i-th exam in the day j (the total number of days Petya prepares for each exam should be **strictly** equal to the number of days needed to prepare for it).

  Assume that the exams are numbered in order of appearing in the input, starting from 1.

  If there are multiple schedules, print any of them.

input
```
5 2
1 3 1
1 5 1
```
output
```
1 2 3 0 3
```

input
```
3 2
1 3 1
1 2 1
```
output
```
-1
```

input
```
10 3
4 7 2
1 10 3
8 9 1
```
output
```
2 2 2 1 1 0 4 3 4 4
```

In the first example Petya can, for example, prepare for exam 1 in the first day, prepare for exam 2 in the second day, pass exam 1 in the third day, relax in the fourth day, and pass exam 2 in the fifth day. So, he can prepare and pass all exams.

In the second example, there are three days and two exams. So, Petya can prepare in only one day (because in two other days he should pass exams). Then Petya can not prepare and pass all exams.