A. Restoring Three Numbers

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

Polycarp has guessed three positive integers a, b and c. He keeps these numbers in secret, but he writes down four numbers on a board in arbitrary order — their pairwise sums (three numbers) and sum of all three numbers (one number). So, there are four numbers on a board in random order: $a+b$, $a+c$, $b+c$ and $a+b+c$.

You have to guess three numbers a, b and c using given numbers. Print three guessed integers in any order.

Pay attention that some given numbers a, b and c can be equal (it is also possible that $a=b=c$).

Input

The only line of the input contains four positive integers x_1, x_2, x_3, x_4 ($2 \le x_i \le 10^9$) — numbers written on a board in random order. It is guaranteed that the answer exists for the given number x_1, x_2, x_3, x_4.

Output

Print such positive integers a, b and c that four numbers written on a board are values $a+b$, $a+c$, $b+c$ and $a+b+c$ written in some order. Print a, b and c in any order. If there are several answers, you can print any. It is guaranteed that the answer exists.

Examples

input
```
3 6 5 4
```
output
```
2 1 3
```

input
```
40 40 40 60
```
output
```
20 20 20
```

input
```
201 101 101 200
```
output
```
1 100 100
```

B. Make Them Equal

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

You are given a sequence $a_1, a_2, \dots, a_n$ consisting of n integers.

You can choose any non-negative integer D (i.e. $D \ge 0$), and for each a_i you can:

- add D (only once), i. e. perform $a_i := a_i + D$, or
- subtract D (only once), i. e. perform $a_i := a_i - D$, or
- leave the value of a_i unchanged.

It is possible that after an operation the value a_i becomes negative.

Your goal is to choose such **minimum non-negative integer** D and perform changes in such a way, that all a_i are equal (i.e. $a_1 = a_2 = \dots = a_n$).

Print the required D or, if it is impossible to choose such value D, print -1.

For example, for array $[2, 8]$ the value $D=3$ is minimum possible because you can obtain the array $[5, 5]$ if you will add D to 2 and subtract D from 8. And for array $[1, 4, 7, 7]$ the value $D=3$ is also minimum possible. You can add it to 1 and subtract it from 7 and obtain the array $[4, 4, 4, 4]$.

Input

The first line of the input contains one integer n ($1 \le n \le 100$) — the number of elements in a.

The second line of the input contains n integers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le 100$) — the sequence a.

Output

Print one integer — the **minimum non-negative integer** value D such that if you add this value to some a_i, subtract this value from some a_i and leave some a_i without changes, all obtained values become equal.

If it is impossible to choose such value D, print -1.

Examples

input

```
6
1 4 4 7 4 1
```

output

```
3
```

input

```
5
2 2 5 2 5
```

output

```
3
```

input

```
4
1 3 3 7
```

output

```
-1
```

input

```
2
2 8
```

output

```
3
```

C. Gourmet Cat

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

Polycarp has a cat and his cat is a real gourmet! Dependent on a day of the week he eats certain type of food:

- on Mondays, Thursdays and Sundays he eats *fish food*;
- on Tuesdays and Saturdays he eats *rabbit stew*;
- on other days of week he eats *chicken stake*.

Polycarp plans to go on a trip and already packed his backpack. His backpack contains:

- a daily rations of *fish food*;
- b daily rations of *rabbit stew*;
- c daily rations of *chicken stakes*.

Polycarp has to choose such day of the week to start his trip that his cat can eat without additional food purchases as long as possible. Print the maximum number of days the cat can eat in a trip without additional food purchases, if Polycarp chooses the day of the week to start his trip optimally.

Input

The first line of the input contains three positive integers a, b and c ($1 \le a, b, c \le 7 \cdot 10^8$) — the number of daily rations of *fish food*, *rabbit stew* and *chicken stakes* in Polycarps backpack correspondingly.

Output

Print the maximum number of days the cat can eat in a trip without additional food purchases, if Polycarp chooses the day of the week to start his trip optimally.

Examples

input

```
2 1 1
```

output

```
4
```

input

```
3 2 2
```

output

```
7
```

input

```
1 100 1
```

output

```
3
```

input

```
30 20 10
```

output

```
39
```

Note

In the first example the best day for start of the trip is Sunday. In this case, during Sunday and Monday the cat will eat *fish food*, during Tuesday — *rabbit stew* and during Wednesday — *chicken stake*. So, after four days of the trip all food will be eaten.

In the second example Polycarp can start his trip in any day of the week. In any case there are food supplies only for one week in Polycarps backpack.

In the third example Polycarp can start his trip in any day, excluding Wednesday, Saturday and Sunday. In this case, the cat will eat three different dishes in three days. Nevertheless that after three days of a trip there will be 99 portions of *rabbit stew* in a backpack, can cannot eat anything in fourth day of a trip.

D. Walking Robot

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

There is a robot staying at $X = 0$ on the Ox axis. He has to walk to $X = n$. You are controlling this robot and controlling how he goes. The robot has a battery and an accumulator with a solar panel.

The i-th segment of the path (from $X = i - 1$ to $X = i$) can be exposed to sunlight or not. The array s denotes which segments are exposed to sunlight: if segment i is exposed, then $s_i = 1$, otherwise $s_i = 0$.

The robot has one battery of capacity b and one accumulator of capacity a. For each segment, you should choose which type of energy storage robot will use to go to the next point (it can be either battery or accumulator). If the robot goes using the battery, the current charge of the battery is decreased by one (the robot can't use the battery if its charge is zero). And if the robot goes using the accumulator, the current charge of the accumulator is decreased by one (and the robot also can't use the accumulator if its charge is zero).

If the current segment is **exposed to sunlight** and the robot goes through it **using the battery**, the charge of the accumulator increases by one (of course, its charge can't become higher than it's maximum capacity).

If accumulator is used to pass some segment, its charge decreases by 1 no matter if the segment is exposed or not.

You understand that it is not always possible to walk to $X = n$. You want your robot to go as far as possible. Find the maximum number of segments of distance the robot can pass if you control him optimally.

Input

The first line of the input contains three integers n, b, a ($1 \le n, b, a \le 2 \cdot 10^5$) — the robot's destination point, the battery capacity and the accumulator capacity, respectively.

The second line of the input contains n integers $s_1, s_2, \dots, s_n$ ($0 \le s_i \le 1$), where s_i is 1 if the i-th segment of distance is exposed to sunlight, and 0 otherwise.

Output

Print one integer — the maximum number of segments the robot can pass if you control him optimally.

Examples

input

```
5 2 1
0 1 0 1 0
```

output

```
5
```

input

```
6 2 1
1 0 0 1 0 1
```

output

```
3
```

Note

In the first example the robot can go through the first segment using the accumulator, and charge levels become $b = 2$ and $a = 0$. The second segment can be passed using the battery, and charge levels become $b = 1$ and $a = 1$. The third segment can be passed using the accumulator, and charge levels become $b = 1$ and $a = 0$. The fourth segment can be passed using the battery, and charge levels become $b = 0$ and $a = 1$. And the fifth segment can be passed using the accumulator.

In the second example the robot can go through the maximum number of segments using battery two times and accumulator one time in any order.

E. Two Teams

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

There are n students standing in a row. Two coaches are forming two teams — the first coach chooses the first team and the second coach chooses the second team.

The i-th student has integer programming skill a_i. All programming skills are **distinct** and between 1 and n, inclusive.

Firstly, the first coach will choose the student with maximum programming skill among all students not taken into any team, **and** k closest students to the left of him and k closest students to the right of him (if there are less than k students to the left or to the right, all of them will be chosen). All students that are chosen leave the row and join the first team. Secondly, the second coach will make the same move (but all students chosen by him join the second team). Then again the first coach will make such move, and so on. This repeats until the row becomes empty (i. e. the process ends when each student becomes to some team).

Your problem is to determine which students will be taken into the first team and which students will be taken into the second team.

Input

The first line of the input contains two integers n and k ($1 \le k \le n \le 2 \cdot 10^5$) — the number of students and the value determining the range of chosen students during each move, respectively.

The second line of the input contains n integers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le n$), where a_i is the programming skill of the i-th student. It is guaranteed that all programming skills are **distinct**.

Output

Print a string of n characters; i-th character should be 1 if i-th student joins the first team, or 2 otherwise.

Examples

input

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

output

```
11111
```

input

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

output

```
22111
```

input

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

output

```
1121122
```

input

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

output

```
21112
```

Note

In the first example the first coach chooses the student on a position 3, and the row becomes empty (all students join the first team).

In the second example the first coach chooses the student on position 4, and the row becomes $[2, 1]$ (students with programming skills $[3, 4, 5]$ join the first team). Then the second coach chooses the student on position 1, and the row becomes empty (and students with programming skills $[1, 2]$ join the second team).

In the third example the first coach chooses the student on position 1, and the row becomes $[1, 3, 5, 4, 6]$ (students with programming skills $[2, 7]$ join the first team). Then the second coach chooses the student on position 5, and the row becomes $[1, 3, 5]$ (students with programming skills $[4, 6]$ join the second team). Then the first coach chooses the student on position 3, and the row becomes $[1]$ (students with programming skills $[3, 5]$ join the first team). And then the second coach chooses the remaining student (and the student with programming skill 1 joins the second team).

In the fourth example the first coach chooses the student on position 3, and the row becomes $[2, 1]$ (students with programming skills $[3, 4, 5]$ join the first team). Then the second coach chooses the student on position 1, and the row becomes empty (and students with programming skills $[1, 2]$ join the second team).

F. Shovels Shop

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

There are n shovels in the nearby shop. The i-th shovel costs a_i bourles.

Misha has to buy **exactly** k shovels. Each shovel can be bought **no more than once**.

Misha can buy shovels by several purchases. During one purchase he can choose any subset of remaining (non-bought) shovels and buy this subset.

There are also m special offers in the shop. The j-th of them is given as a pair (x_j, y_j), and it means that if Misha buys **exactly** x_j shovels **during one purchase** then y_j **most cheapest** of them are for free (i.e. he will not pay for y_j most cheapest shovels during the current purchase).

Misha can use any offer any (possibly, zero) number of times, but he cannot use **more than one** offer during **one purchase** (but he can buy shovels without using any offers).

Your task is to calculate the minimum cost of buying k shovels, if Misha buys them optimally.

Input

The first line of the input contains three integers n, m and k ($1 \le n, m \le 2 \cdot 10^5, 1 \le k \le min(n, 2000)$) — the number of shovels in the shop, the number of special offers and the number of shovels Misha has to buy, correspondingly.

The second line of the input contains n integers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le 2 \cdot 10^5$), where a_i is the cost of the i-th shovel.

The next m lines contain special offers. The j-th of them is given as a pair of integers (x_i, y_i) ($1 \le y_i \le x_i \le n$) and means that if Misha buys exactly x_i shovels during some purchase, then he can take y_i most cheapest of them for free.

Output

Print one integer — the minimum cost of buying k shovels if Misha buys them optimally.

Examples

input

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

output

```
7
```

input

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

output

```
17
```

input

```
5 1 4
2 5 7 4 6
5 4
```

output

```
17
```

Note

In the first example Misha can buy shovels on positions 1 and 4 (both with costs 2) during the first purchase and get one of them for free using the first or the third special offer. And then he can buy shovels on positions 3 and 6 (with costs 4 and 3) during the second purchase and get the second one for free using the first or the third special offer. Then he can buy the shovel on a position 7 with cost 1. So the total cost is $4 + 2 + 1 = 7$.

In the second example Misha can buy shovels on positions 1, 2, 3, 4 and 8 (costs are 6, 8, 5, 1 and 2) and get three cheapest (with costs 5, 1 and 2) for free. And then he can buy shovels on positions 6, 7 and 9 (all with costs 1) without using any special offers. So the total cost is $6 + 8 + 1 + 1 + 1 = 17$.

In the third example Misha can buy four cheapest shovels without using any special offers and get the total cost 17.

G. Minimum Possible LCM

time limit per test: 4 seconds

memory limit per test: 1024 megabytes

input: standard input

output: standard output

You are given an array a consisting of n integers $a_1, a_2, \dots, a_n$.

Your problem is to find such pair of indices i, j ($1 \le i < j \le n$) that $lcm(a_i, a_j)$ is minimum possible.

$lcm(x, y)$ is the least common multiple of x and y (minimum positive number such that both x and y are divisors of this number).

Input

The first line of the input contains one integer n ($2 \le n \le 10^6$) — the number of elements in a.

The second line of the input contains n integers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le 10^7$), where a_i is the i-th element of a.

Output

Print two integers i and j ($1 \le i < j \le n$) such that the value of $lcm(a_i, a_j)$ is minimum among all valid pairs i, j. If there are multiple answers, you can print any.

Examples

input

```
5
2 4 8 3 6
```

output

```
1 2
```

input

```
5
5 2 11 3 7
```

output

```
2 4
```

input

```
6
2 5 10 1 10 2
```

output

```
1 4
```