Educational Codeforces Round 18

A. New Bus Route

1 second, 256 megabytes

There are n cities situated along the main road of Berland. Cities are represented by their coordinates — integer numbers $a_1, a_2, ..., a_n$. All coordinates are pairwise distinct.

It is possible to get from one city to another only by bus. But all buses and roads are very old, so the Minister of Transport decided to build a new bus route. The Minister doesn't want to spend large amounts of money — he wants to choose two cities in such a way that the distance between them is minimal possible. The distance between two cities is equal to the absolute value of the difference between their coordinates.

It is possible that there are multiple pairs of cities with minimal possible distance, so the Minister wants to know the quantity of such pairs.

Your task is to write a program that will calculate the minimal possible distance between two pairs of cities and the quantity of pairs which have this distance.

Input

The first line contains one integer number n ($2 \le n \le 2\cdot 10^5$).

The second line contains n integer numbers $a_1, a_2, ..., a_n$ ($-10^9 \le a_i \le 10^9$). All numbers a_i are pairwise distinct.

Output

Print two integer numbers — the minimal distance and the quantity of pairs with this distance.

input

```
4
6 -3 0 4
```

output

```
2 1
```

input

```
3
-2 0 2
```

output

```
2 2
```

In the first example the distance between the first city and the fourth city is $|4 - 6| = 2$, and it is the only pair with this distance.

B. Counting-out Rhyme

1 second, 256 megabytes

n children are standing in a circle and playing the counting-out game. Children are numbered clockwise from 1 to n. In the beginning, the first child is considered the leader. The game is played in k steps. In the i-th step the leader counts out a_i people in clockwise order, starting from the next person. The last one to be pointed at by the leader is eliminated, and the next player after him becomes the new leader.

For example, if there are children with numbers $[8, 10, 13, 14, 16]$ currently in the circle, the leader is child 13 and $a_i = 12$, then counting-out rhyme ends on child 16, who is eliminated. Child 8 becomes the leader.

You have to write a program which prints the number of the child to be eliminated on every step.

Input

The first line contains two integer numbers n and k ($2 \le n \le 100$, $1 \le k \le n - 1$).

The next line contains k integer numbers $a_1, a_2, ..., a_k$ ($1 \le a_i \le 10^9$).

Output

Print k numbers, the i-th one corresponds to the number of child to be eliminated at the i-th step.

input

```
7 5
10 4 11 4 1
```

output

```
4 2 5 6 1
```

input

```
3 2
2 5
```

output

```
3 2
```

Let's consider first example:

- In the first step child 4 is eliminated, child 5 becomes the leader.
- In the second step child 2 is eliminated, child 3 becomes the leader.
- In the third step child 5 is eliminated, child 6 becomes the leader.
- In the fourth step child 6 is eliminated, child 7 becomes the leader.
- In the final step child 1 is eliminated, child 3 becomes the leader.

C. Divide by Three

1 second, 256 megabytes

A positive integer number n is written on a blackboard. It consists of not more than 10^5 digits. You have to transform it into a *beautiful* number by erasing some of the digits, and you want to erase as few digits as possible.

The number is called beautiful if it consists of at least one digit, doesn't have leading zeroes and is a multiple of 3. For example, `0`, `99`, `10110` are beautiful numbers, and `00`, `03`, `122` are not.

Write a program which for the given n will find a beautiful number such that n can be transformed into this number by erasing as few digits as possible. You can erase an arbitraty set of digits. For example, they don't have to go one after another in the number n.

If it's impossible to obtain a beautiful number, print `-1`. If there are multiple answers, print any of them.

Input

The first line of input contains n — a positive integer number without leading zeroes ($1 \le n < 10^{100000}$).

Output

Print one number — any beautiful number obtained by erasing as few as possible digits. If there is no answer, print -1.

input

```
1033
```

output

```
33
```

input

```
10
```

output

```
0
```

input

```
11
```

output

```
-1
```

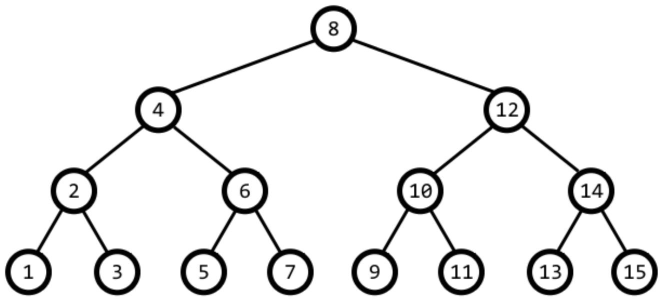
In the first example it is enough to erase only the first digit to obtain a multiple of 3. But if we erase the first digit, then we obtain a number with a leading zero. So the minimum number of digits to be erased is two.

D. Paths in a Complete Binary Tree

3 seconds, 256 megabytes

T is a complete binary tree consisting of n vertices. It means that exactly one vertex is a root, and each vertex is either a leaf (and doesn't have children) or an inner node (and has exactly two children). All leaves of a complete binary tree have the same depth (distance from the root). So n is a number such that $n + 1$ is a power of 2.

In the picture you can see a complete binary tree with $n = 15$.



Vertices are numbered from 1 to n in a special recursive way: we recursively assign numbers to all vertices from the left subtree (if current vertex is not a leaf), then assign a number to the current vertex, and then recursively assign numbers to all vertices from the right subtree (if it exists). In the picture vertices are numbered exactly using this algorithm. It is clear that for each size of a complete binary tree exists exactly one way to give numbers to all vertices. This way of numbering is called *symmetric*.

You have to write a program that for given n answers q queries to the tree.

Each query consists of an integer number u_i ($1 \le u_i \le n$) and a string s_i, where u_i is the number of vertex, and s_i represents the path starting from this vertex. String s_i doesn't contain any characters other than '`L`', '`R`' and '`U`', which mean traverse to the left child, to the right child and to the parent, respectively. Characters from s_i have to be processed from left to right, considering that u_i is the vertex where the path starts. If it's impossible to process a character (for example, to go to the left child of a leaf), then you have to skip it. The answer is the number of vertex where the path represented by s_i ends.

For example, if $u_i = 4$ and $s_i =$ «`UURL`», then the answer is 10.

Input

The first line contains two integer numbers n and q ($1 \le n \le 10^{18}$, $q \ge 1$). n is such that $n + 1$ is a power of 2.

The next $2q$ lines represent queries; each query consists of two consecutive lines. The first of these two lines contains u_i ($1 \le u_i \le n$), the second contains non-empty string s_i. s_i doesn't contain any characters other than '`L`', '`R`' and '`U`'.

It is guaranteed that the sum of lengths of s_i (for each i such that $1 \le i \le q$) doesn't exceed 10^5.

Output

Print q numbers, i-th number must be the answer to the i-th query.

input
```
15 2
4
UURL
8
LRLLLLLLLL
```
output
```
10
5
```

E. Colored Balls

1 second, 256 megabytes

There are n boxes with colored balls on the table. Colors are numbered from 1 to n. i-th box contains a_i balls, all of which have color i. You have to write a program that will divide all balls into sets such that:

- each ball belongs to exactly one of the sets,
- there are no empty sets,
- there is no set containing two (or more) balls of different colors (each set contains only balls of one color),
- there are no two sets such that the difference between their sizes is greater than 1.

Print the minimum possible number of sets.

Input

The first line contains one integer number n ($1 \le n \le 500$).

The second line contains n integer numbers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le 10^9$).

Output

Print one integer number — the minimum possible number of sets.

input

```
3
4 7 8
```

output

```
5
```

input

```
2
2 7
```

output

```
4
```

In the first example the balls can be divided into sets like that: one set with 4 balls of the first color, two sets with 3 and 4 balls, respectively, of the second color, and two sets with 4 balls of the third color.

F. Mages and Monsters

2 seconds, 256 megabytes

Vova plays a computer game known as Mages and Monsters. Vova's character is a mage. Though as he has just started, his character knows no spells.

Vova's character can learn new spells during the game. Every spell is characterized by two values x_i and y_i — damage per second and mana cost per second, respectively. Vova doesn't have to use a spell for an integer amount of seconds. More formally, if he uses a spell with damage x and mana cost y for z seconds, then he will deal $x \cdot z$ damage and spend $y \cdot z$ mana (no rounding). If there is no mana left (mana amount is set in the start of the game and it remains the same at the beginning of every fight), then character won't be able to use any spells. It is prohibited to use multiple spells simultaneously.

Also Vova can fight monsters. Every monster is characterized by two values t_j and h_j — monster kills Vova's character in t_j seconds and has h_j health points. Mana refills after every fight (or Vova's character revives with full mana reserve), so previous fights have no influence on further ones.

Vova's character kills a monster, if he deals h_j damage to it in no more than t_j seconds using his spells (it is allowed to use more than one spell in a fight) and spending no more mana than he had at the beginning of the fight. **If monster's health becomes zero exactly in t_j seconds (it means that the monster and Vova's character kill each other at the same time), then Vova wins the fight**.

You have to write a program which can answer two types of queries:

- $1\ x\ y$ — Vova's character learns new spell which deals x damage per second and costs y mana per second.
- $2\ t\ h$ — Vova fights the monster which kills his character in t seconds and has h health points.

Note that queries are given in a different form. Also remember that Vova's character knows no spells at the beginning of the game.

For every query of second type you have to determine if Vova is able to win the fight with corresponding monster.

Input

The first line contains two integer numbers q and m $(2 \le q \le 10^5, 1 \le m \le 10^{12})$ — the number of queries and the amount of mana at the beginning of every fight.

i-th of each next q lines contains three numbers k_i, a_i and b_i $(1 \le k_i \le 2, 1 \le a_i, b_i \le 10^6)$.

Using them you can restore queries this way: let j be the index of the last query of second type with positive answer ($j = 0$ if there were none of these).

- If $k_i = 1$, then character learns spell with $x = (a_i + j)\ mod\ 10^6 + 1$, $y = (b_i + j)\ mod\ 10^6 + 1$.
- If $k_i = 2$, then you have to determine if Vova is able to win the fight against monster with $t = (a_i + j)\ mod\ 10^6 + 1$, $h = (b_i + j)\ mod\ 10^6 + 1$.

Output

For every query of second type print `YES` if Vova is able to win the fight with corresponding monster and `NO` otherwise.

input

```
3 100
1 4 9
2 19 49
2 19 49
```

output

```
YES
NO
```

In first example Vova's character at first learns the spell with 5 damage and 10 mana cost per second. Next query is a fight with monster which can kill character in 20 seconds and has 50 health points. Vova kills it in 10 seconds (spending 100 mana). Next monster has 52 health, so Vova can't deal that much damage with only 100 mana.