Codeforces Round #532 (Div. 2)

A. Roman and Browser

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

This morning, Roman woke up and opened the browser with n opened tabs numbered from 1 to n. There are two kinds of tabs: those with the information required for the test and those with social network sites. Roman decided that there are too many tabs open so he wants to close some of them.

He decided to accomplish this by closing every k-th ($2 \leq k \leq n-1$) tab. Only then he will decide whether he wants to study for the test or to chat on the social networks. Formally, Roman will choose one tab (let its number be b) and then close all tabs with numbers $c = b + i \cdot k$ that satisfy the following condition: $1 \leq c \leq n$ and i is an integer (it may be positive, negative or zero).

For example, if $k = 3$, $n = 14$ and Roman chooses $b = 8$, then he will close tabs with numbers 2, 5, 8, 11 and 14.

After closing the tabs Roman will calculate the amount of remaining tabs with the information for the test (let's denote it e) and the amount of remaining social network tabs (s). Help Roman to calculate the maximal absolute value of the difference of those values $|e - s|$ so that it would be easy to decide what to do next.

Input

The first line contains two integers n and k ($2 \leq k < n \leq 100$) — the amount of tabs opened currently and the distance between the tabs closed.

The second line consists of n integers, each of them equal either to 1 or to -1. The i-th integer denotes the type of the i-th tab: if it is equal to 1, this tab contains information for the test, and if it is equal to -1, it's a social network tab.

Output

Output a single integer — the maximum absolute difference between the amounts of remaining tabs of different types $|e - s|$.

Examples

input

```
4 2
1 1 -1 1
```

output

```
2
```

input

```
14 3
-1 1 -1 -1 1 -1 -1 1 -1 -1 1 -1 -1 1
```

output

```
9
```

Note

In the first example we can choose $b = 1$ or $b = 3$. We will delete then one tab of each type and the remaining tabs are then all contain test information. Thus, $e = 2$ and $s = 0$ and $|e - s| = 2$.

In the second example, on the contrary, we can leave opened only tabs that have social networks opened in them.

B. Build a Contest

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

Arkady coordinates rounds on some not really famous competitive programming platform. Each round features n problems of distinct difficulty, the difficulties are numbered from 1 to n.

To hold a round Arkady needs n new (not used previously) problems, one for each difficulty. As for now, Arkady creates all the problems himself, but unfortunately, he can't just create a problem of a desired difficulty. Instead, when he creates a problem, he evaluates its difficulty from 1 to n and puts it into the problems pool.

At each moment when Arkady can choose a set of n new problems of distinct difficulties from the pool, he holds a round with these problems and removes them from the pool. Arkady always creates one problem at a time, so if he can hold a round after creating a problem, he immediately does it.

You are given a sequence of problems' difficulties in the order Arkady created them. For each problem, determine whether Arkady held the round right after creating this problem, or not. Initially the problems pool is empty.

Input

The first line contains two integers n and m ($1 \le n, m \le 10^5$) — the number of difficulty levels and the number of problems Arkady created.

The second line contains m integers $a_1, a_2, \ldots, a_m$ ($1 \le a_i \le n$) — the problems' difficulties in the order Arkady created them.

Output

Print a line containing m digits. The i-th digit should be 1 if Arkady held the round after creation of the i-th problem, and 0 otherwise.

Examples

input

```
3 11
2 3 1 2 2 2 3 2 2 3 1
```

output

```
00100000001
```

input

```
4 8
4 1 3 3 2 3 3 3
```

output

```
00001000
```

Note

In the first example Arkady held the round after the first three problems, because they are of distinct difficulties, and then only after the last problem.

C. NN and the Optical Illusion

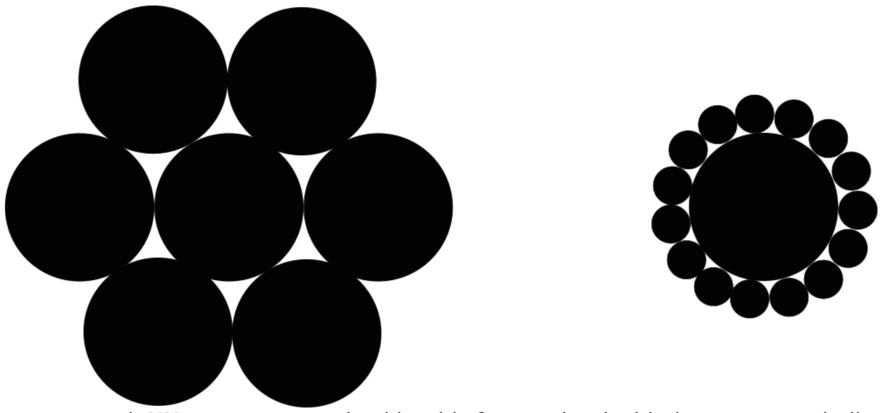
time limit per test: 2 seconds

memory limit per test: 256 megabytes

input: standard input

output: standard output

NN is an experienced internet user and that means he spends a lot of time on the social media. Once he found the following image on the Net, which asked him to compare the sizes of inner circles:



It turned out that the circles are equal. NN was very surprised by this fact, so he decided to create a similar picture himself.

He managed to calculate the number of outer circles n and the radius of the inner circle r. NN thinks that, using this information, you can exactly determine the radius of the outer circles R so that the inner circle touches all of the outer ones externally and each pair of neighboring outer circles also touches each other. While NN tried very hard to guess the required radius, he didn't manage to do that.

Help NN find the required radius for building the required picture.

Input

The first and the only line of the input file contains two numbers n and r ($3 \le n \le 100$, $1 \le r \le 100$) — the number of the outer circles and the radius of the inner circle respectively.

Output

Output a single number R — the radius of the outer circle required for building the required picture.

Your answer will be accepted if its relative or absolute error does not exceed 10^{-6}.

Formally, if your answer is a and the jury's answer is b. Your answer is accepted if and only when $\frac{|a-b|}{max(1,|b|)} \le 10^{-6}$.

Examples

input
```
3 1
```
output
```
6.4641016
```

input
```
6 1
```
output
```
1.0000000
```

input
```
100 100
```
output
```
3.2429391
```

D. Dasha and Chess

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

This is an interactive task.

Dasha and NN like playing chess. While playing a match they decided that normal chess isn't interesting enough for them, so they invented a game described below.

There are 666 black rooks and 1 white king on the chess board of size 999×999. The white king wins if he gets checked by rook, or, in other words, if he moves onto the square which shares either a row or column with a black rook.

The sides take turns, starting with white. NN plays as a white king and on each of his turns he moves a king to one of the squares that are adjacent to his current position either by side or diagonally, or, formally, if the king was on the square (x, y), it can move to the square (nx, ny) if and only $\max(|nx - x|, |ny - y|) = 1$, $1 \le nx, ny \le 999$. NN is also forbidden from moving onto the squares occupied with black rooks, however, he can move onto the same row or column as a black rook.

Dasha, however, neglects playing by the chess rules, and instead of moving rooks normally she moves one of her rooks on any space devoid of other chess pieces. It is also possible that the rook would move onto the same square it was before and the position wouldn't change. However, she can't move the rook on the same row or column with the king.

Each player makes 2000 turns, if the white king wasn't checked by a black rook during those turns, black wins.

NN doesn't like losing, but thinks the task is too difficult for him, so he asks you to write a program that will always win playing for the white king. Note that Dasha can see your king and play depending on its position.

Input

In the beginning your program will receive 667 lines from input. Each line contains two integers x and y ($1 \le x, y \le 999$) — the piece's coordinates. The first line contains the coordinates of the king and the next 666 contain the coordinates of the rooks. The first coordinate denotes the number of the row where the piece is located, the second denotes the column. It is guaranteed that initially the king isn't in check and that all pieces occupy different squares.

Output

After getting king checked, you program should terminate immediately without printing anything extra.

Interaction

To make a move with the king, output two integers x and y ($1 \le x, y \le 999$) — the square to which the king would be moved. The king cannot move onto the square already occupied by a rook. It is guaranteed that the king would always have a valid move.

After each of your turns read the rook's turn in the following format: a single line containing three integers k, x and y ($1 \le k \le 666$, $1 \le x_i, y_i \le 999$) — the number of the rook that would move and the square it would move to. It is guaranteed that the rook wouldn't move to a square already occupied by another chess piece, but it can move onto the square where it was before the turn so that its position wouldn't change. It is guaranteed that the move does not put your king into a check. If your king got in check, all three integers would be equal to -1 and in that case your program should terminate immediately.

After printing your turn do not forget to output end of line and flush the output. Otherwise you will get `Idleness limit exceeded`. To do this, use:

- `fflush(stdout)` or `cout.flush()` in C++;
- `System.out.flush()` in Java;
- `flush(output)` in Pascal;

- `stdout.flush()` in Python;
- see documentation for other languages.

Answer "`0 0 0`" instead of a correct answer means that you made an invalid query. Exit immediately after receiving "`0 0 0`" and you will see `Wrong answer` verdict. Otherwise you can get an arbitrary verdict because your solution will continue to read from a closed stream.

Hacks are not allowed for this problem.

Example

input

```
999 999
1 1
1 2
2 1
2 2
1 3
2 3
<...>
26 13
26 14
26 15
26 16

1 700 800

2 1 2

<...>

-1 -1 -1
```

output

```
999 998

999 997

<...>

999 26
```

Note

The example is trimmed. The full initial positions of the rooks in the first test are available at https://pastebin.com/qQCTXgKP. It is not guaranteed that they will behave as in the example.

E. Andrew and Taxi

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

Andrew prefers taxi to other means of transport, but recently most taxi drivers have been acting inappropriately. In order to earn more money, taxi drivers started to drive in circles. Roads in Andrew's city are one-way, and people are not necessary able to travel from one part to another, but it pales in comparison to insidious taxi drivers.

The mayor of the city decided to change the direction of certain roads so that the taxi drivers wouldn't be able to increase the cost of the trip endlessly. More formally, if the taxi driver is on a certain crossroads, they wouldn't be able to reach it again if he performs a nonzero trip.

Traffic controllers are needed in order to change the direction the road goes. For every road it is known how many traffic controllers are needed to change the direction of the road to the opposite one. It is allowed to change the directions of roads one by one, meaning that each traffic controller can participate in reversing two or more roads.

You need to calculate the minimum number of traffic controllers that you need to hire to perform the task and the list of the roads that need to be reversed.

Input

The first line contains two integers n and m ($2 \le n \le 100\,000$, $1 \le m \le 100\,000$) — the number of crossroads and the number of roads in the city, respectively.

Each of the following m lines contain three integers u_i, v_i and c_i ($1 \le u_i, v_i \le n$, $1 \le c_i \le 10^9$, $u_i \ne v_i$) — the crossroads the road starts at, the crossroads the road ends at and the number of traffic controllers required to reverse this road.

Output

In the first line output two integers the minimal amount of traffic controllers required to complete the task and amount of roads k which should be reversed. k should not be minimized.

In the next line output k integers separated by spaces — numbers of roads, the directions of which should be reversed. The roads are numerated from 1 in the order they are written in the input. If there are many solutions, print any of them.

Examples

input

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

output

```
2 2
1 3
```

input

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

output

```
3 3
3 4 7
```

Note

There are two simple cycles in the first example: $1 \to 5 \to 2 \to 1$ and $2 \to 3 \to 4 \to 5 \to 2$. One traffic controller can only reverse the road $2 \to 1$ and he can't destroy the second cycle by himself. Two traffic controllers can reverse roads $2 \to 1$ and $2 \to 3$ which would satisfy the condition.

In the second example one traffic controller can't destroy the cycle $1 \to 3 \to 2 \to 1$. With the help of three controllers we can, for example, reverse roads $1 \to 3$,$2 \to 4$, $1 \to 5$.

F. Ivan and Burgers

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

Ivan loves burgers and spending money. There are n burger joints on the street where Ivan lives. Ivan has q friends, and the i-th friend suggested to meet at the joint l_i and walk to the joint r_i $(l_i \leq r_i)$. While strolling with the i-th friend Ivan can visit all joints x which satisfy $l_i \leq x \leq r_i$.

For each joint Ivan knows the cost of the most expensive burger in it, it costs c_i burles. Ivan wants to visit some subset of joints on his way, in each of them he will buy the most expensive burger and spend the most money. But there is a small issue: his card broke and instead of charging him for purchases, the amount of money on it changes as follows.

If Ivan had d burles before the purchase and he spent c burles at the joint, then after the purchase he would have $d \oplus c$ burles, where $\oplus$ denotes the bitwise XOR operation.

Currently Ivan has $2^{2^{100}} - 1$ burles and he wants to go out for a walk. Help him to determine the maximal amount of burles he can spend if he goes for a walk with the friend i. The amount of burles he spends is defined as the difference between the initial amount on his account and the final account.

Input

The first line contains one integer n $(1 \leq n \leq 500\,000)$ — the number of burger shops.

The next line contains n integers $c_1, c_2, \ldots, c_n$ $(0 \leq c_i \leq 10^6)$, where c_i — the cost of the most expensive burger in the burger joint i.

The third line contains one integer q $(1 \leq q \leq 500\,000)$ — the number of Ivan's friends.

Each of the next q lines contain two integers l_i and r_i $(1 \leq l_i \leq r_i \leq n)$ — pairs of numbers of burger shops between which Ivan will walk.

Output

Output q lines, i-th of which containing the maximum amount of money Ivan can spend with the friend i.

Examples

input

```
4
7 2 3 4
3
1 4
2 3
1 3
```

output

```
7
3
7
```

input

```
5
12 14 23 13 7
15
1 1
1 2
1 3
1 4
1 5
2 2
2 3
2 4
2 5
3 3
3 4
3 5
4 4
4 5
5 5
```

output

```
12
14
27
27
31
14
25
26
30
23
26
29
13
13
7
```

Note

In the first test, in order to spend the maximum amount of money with the first and third friends, Ivan just needs to go into the first burger. With a second friend, Ivan just go to the third burger.

In the second test for a third friend (who is going to walk from the first to the third burger), there are only 8 options to spend money — 0, 12, 14, 23, $12 \oplus 14 = 2$, $14 \oplus 23 = 25$, $12 \oplus 23 = 27$, $12 \oplus 14 \oplus 23 = 20$. The maximum amount of money it turns out to spend, if you go to the first and third burger — $12 \oplus 23 = 27$.