Educational Codeforces Round 72 (Rated for Div. 2)

A. Creating a Character

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

You play your favourite game yet another time. You chose the character you didn't play before. It has str points of strength and int points of intelligence. Also, at start, the character has exp free experience points you can invest either in strength or in intelligence (by investing one point you can either raise strength by 1 or raise intelligence by 1).

Since you'd like to make some fun you want to create a jock character, so it has more strength than intelligence points (resulting strength is **strictly greater** than the resulting intelligence).

Calculate the number of different character builds you can create (for the purpose of replayability) if you must **invest all free points**. Two character builds are different if their strength and/or intellect are different.

Input

The first line contains the single integer T ($1 \le T \le 100$) — the number of queries. Next T lines contain descriptions of queries — one per line.

This line contains three integers str, int and exp ($1 \le str, int \le 10^8$, $0 \le exp \le 10^8$) — the initial strength and intelligence of the character and the number of free points, respectively.

Output

Print T integers — one per query. For each query print the number of different character builds you can create.

Example

input
4 5 3 4 2 1 0 3 5 5 4 10 6
output
3 1 2 0

Note

In the first query there are only three appropriate character builds: $(str = 7, int = 5)$, $(8, 4)$ and $(9, 3)$. All other builds are either too smart or don't use all free points.

In the second query there is only one possible build: $(2, 1)$.

In the third query there are two appropriate builds: $(7, 6)$, $(8, 5)$.

In the fourth query all builds have too much brains.

B. Zmei Gorynich

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

You are fighting with Zmei Gorynich — a ferocious monster from Slavic myths, a huge dragon-like reptile with multiple heads!

Initially Zmei Gorynich has x heads. You can deal n types of blows. If you deal a blow of the i-th type, you decrease the number of Gorynich's heads by $min(d_i, curX)$, there $curX$ is the current number of heads. But if after this blow Zmei Gorynich has at least one head, he grows h_i new heads. If $curX = 0$ then Gorynich is defeated.

You can deal each blow any number of times, in any order.

For example, if $curX = 10$, $d = 7$, $h = 10$ then the number of heads changes to 13 (you cut 7 heads off, but then Zmei grows 10 new ones), but if $curX = 10$, $d = 11$, $h = 100$ then number of heads changes to 0 and Zmei Gorynich is considered defeated.

Calculate the minimum number of blows to defeat Zmei Gorynich!

You have to answer t independent queries.

Input
The first line contains one integer t ($1 \le t \le 100$) – the number of queries.

The first line of each query contains two integers n and x ($1 \le n \le 100$, $1 \le x \le 10^9$) — the number of possible types of blows and the number of heads Zmei initially has, respectively.

The following n lines of each query contain the descriptions of types of blows you can deal. The i-th line contains two integers d_i and h_i ($1 \le d_i, h_i \le 10^9$) — the description of the i-th blow.

Output
For each query print the minimum number of blows you have to deal to defeat Zmei Gorynich.

If Zmei Gorynuch cannot be defeated print -1.

Example

input
3 3 10 6 3 8 2 1 4 4 10 4 1 3 2 2 6 1 100 2 15 10 11 14 100
output
2 3 -1

Note
In the first query you can deal the first blow (after that the number of heads changes to $10 - 6 + 3 = 7$), and then deal the second blow.

In the second query you just deal the first blow three times, and Zmei is defeated.

In third query you can not defeat Zmei Gorynich. Maybe it's better to convince it to stop fighting?

C. The Number Of Good Substrings

time limit per test: 4 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given a binary string s (recall that a string is binary if each character is either 0 or 1).

Let $f(t)$ be the decimal representation of integer t written in binary form (possibly with leading zeroes). For example $f(011) = 3, f(00101) = 5, f(00001) = 1, f(10) = 2, f(000) = 0$ and $f(000100) = 4$.

The substring $s_l, s_{l+1}, \ldots, s_r$ is good if $r - l + 1 = f(s_l \ldots s_r)$.

For example string $s = 1011$ has 5 good substrings: $s_1 \ldots s_1 = 1$, $s_3 \ldots s_3 = 1$, $s_4 \ldots s_4 = 1$, $s_1 \ldots s_2 = 10$ and $s_2 \ldots s_4 = 011$.

Your task is to calculate the number of good substrings of string s.

You have to answer t independent queries.

Input
The first line contains one integer t ($1 \le t \le 1000$) — the number of queries.

The only line of each query contains string s ($1 \le |s| \le 2 \cdot 10^5$), consisting of only digits 0 and 1.

It is guaranteed that $\sum\limits_{i=1}^{t} |s_i| \le 2 \cdot 10^5$.

Output
For each query print one integer — the number of good substrings of string s.

Example

input
4 0110 0101 00001000 0001000
output
4 3 4 3

D. Coloring Edges

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

You are given a directed graph with n vertices and m directed edges without self-loops or multiple edges.

Let's denote the k-coloring of a digraph as following: you color each edge in one of k colors. The k-coloring is **good** if and only if there no cycle formed by edges of same color.

Find a good k-coloring of given digraph with minimum possible k.

Input
The first line contains two integers n and m ($2 \le n \le 5000$, $1 \le m \le 5000$) — the number of vertices and edges in the digraph, respectively.

Next m lines contain description of edges — one per line. Each edge is a pair of integers u and v ($1 \le u, v \le n$, $u \ne v$) — there is directed edge from u to v in the graph.

It is guaranteed that each ordered pair (u, v) appears in the list of edges at most once.

Output
In the first line print single integer k — the number of used colors in a good k-coloring of given graph.

In the second line print m integers $c_1, c_2, \ldots, c_m$ ($1 \le c_i \le k$), where c_i is a color of the i-th edge (in order as they are given in the input).

If there are multiple answers print any of them (you still have to minimize k).

Examples

input
4 5 1 2

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

output

```
1
1 1 1 1 1
```

input

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

output

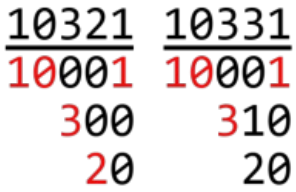
```
2
1 1 2
```

E. Sum Queries?

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

Let's define a *balanced* multiset the following way. Write down the sum of all elements of the multiset in its decimal representation. For each position of that number check if the multiset includes at least one element such that the digit of the element and the digit of the sum at that position are the same. If that holds for every position, then the multiset is *balanced*. Otherwise it's *unbalanced*.

For example, multiset $\{20, 300, 10001\}$ is *balanced* and multiset $\{20, 310, 10001\}$ is *unbalanced*:



The red digits mark the elements and the positions for which these elements have the same digit as the sum. The sum of the first multiset is 10321, every position has the digit required. The sum of the second multiset is 10331 and the second-to-last digit doesn't appear in any number, thus making the multiset *unbalanced*.

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

You are asked to perform some queries on it. The queries can be of two types:

- $1\ i\ x$ — replace a_i with the value x;
- $2\ l\ r$ — find the ***unbalanced*** subset of the multiset of the numbers $a_l, a_{l+1}, \ldots, a_r$ with the minimum sum, or report that no *unbalanced* subset exists.

Note that the empty multiset is *balanced*.

For each query of the second type print the lowest sum of the *unbalanced* subset. Print `-1` if no *unbalanced* subset exists.

Input

The first line contains two integers n and m ($1 \le n, m \le 2 \cdot 10^5$) — the number of elements in the array and the number of queries, respectively.

The second line contains n integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i < 10^9$).

Each of the following m lines contains a query of one of two types:

- $1\ i\ x$ ($1 \le i \le n$, $1 \le x < 10^9$) — replace a_i with the value x;
- $2\ l\ r$ ($1 \le l \le r \le n$) — find the ***unbalanced*** subset of the multiset of the numbers $a_l, a_{l+1}, \ldots, a_r$ with the lowest sum, or report that no *unbalanced* subset exists.

It is guaranteed that there is at least one query of the second type.

Output

For each query of the second type print the lowest sum of the *unbalanced* subset. Print `-1` if no *unbalanced* subset exists.

Example

input

```
4 5
300 10001 20 20
2 1 3
1 1 310
2 1 3
```

2 3 3 2 3 4

output
-1 330 -1 40

Note

All the subsets of multiset $\{20, 300, 10001\}$ are *balanced*, thus the answer is -1.

The possible *unbalanced* subsets in the third query are $\{20, 310\}$ and $\{20, 310, 10001\}$. The lowest sum one is $\{20, 310\}$. Note that you are asked to choose a subset, not a subsegment, thus the chosen elements might not be adjancent in the array.

The fourth query includes only the empty subset and subset $\{20\}$. Both of them are *balanced*.

The last query includes the empty subset and the subsets $\{20\}$, $\{20\}$ and $\{20, 20\}$. Only $\{20, 20\}$ is *unbalanced*, its sum is 40. Note that you are asked to choose a multiset, thus it might include equal elements.

F. Forced Online Queries Problem

time limit per test: 5 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

You are given an undirected graph with n vertices numbered from 1 to n. Initially there are no edges.

You are asked to perform some queries on the graph. Let $last$ be the answer to the latest query of the second type, it is set to 0 before the first such query. Then the queries are the following:

- $1\ x\ y$ ($1 \le x, y \le n$, $x \neq y$) — add an undirected edge between the vertices $(x + last - 1)\ mod\ n + 1$ and $(y + last - 1)\ mod\ n + 1$ if it doesn't exist yet, otherwise remove it;
- $2\ x\ y$ ($1 \le x, y \le n$, $x \neq y$) — check if there exists a path between the vertices $(x + last - 1)\ mod\ n + 1$ and $(y + last - 1)\ mod\ n + 1$, which goes only through currently existing edges, and set $last$ to 1 if so and 0 otherwise.

Good luck!

Input

The first line contains two integer numbers n and m ($2 \le n, m \le 2 \cdot 10^5$) — the number of vertices and the number of queries, respectively.

Each of the following m lines contains a query of one of two aforementioned types. It is guaranteed that there is at least one query of the second type.

Output

Print a string, consisting of characters '0' and '1'. The i-th character should be the answer to the i-th query of the second type. Therefore the length of the string should be equal to the number of queries of the second type.

Examples

input
5 9 1 1 2 1 1 3 2 3 2 1 2 4 2 3 4 1 2 4 2 3 4 1 1 3 2 4 3

output
1010

input
3 9 1 1 2 1 2 3 1 3 1 2 1 3 1 3 2 2 2 3 1 1 2 2 1 2 2 1 2

output
1101

Note

The converted queries in the first example are:

- 1 1 2
- 1 1 3
- 2 3 2
- 1 3 5
- 2 4 5
- 1 2 4
- 2 3 4
- 1 2 4
- 2 5 4

The converted queries in the second example are:

- 1 1 2
- 1 2 3
- 1 3 1
- 2 1 3
- 1 1 3
- 2 3 1
- 1 2 3
- 2 2 3
- 2 1 2
