Problem A. Snow removal

Input file:	`Standard input`
Output file:	`Standard output`
Time limit:	4.5 seconds
Memory limit:	512 mebibytes

The Bear was very happy to see the winter come early. He had hoped that he would no longer be forced to wade through the forest in mud up to his knees while watching out for annoying mosquitoes. Unfortunately, the winter decided to make the Bear's life even harder and it snowed so much that our protagonist can barely move through the woods.

The Bear does not want to give up. He resolved to clean up the forest, or at least remove enough snow to uncover paths leading to his three favorite glades. The forest consists of n glades connected with m (bidirectional) roads. The glades are numbered from 1 to n. It seems that some of the roads might go through tunnels or bridges, which is a bit surprising to the Bear, but he currently has greater issues to deal with. For each road the Bear knows how much time is needed to remove the snow from it.

How much time does the Bear need to clean up enough roads to be able to move between his den (located at the glade 1) and his favorite glades, walking only through cleaned roads?

Input

The first line of the input contains two integers n and m ($4 \le n \le 50\,000$, $3 \le m \le 300\,000$). They denote respectively the number of glades in the forest and the number of roads connecting them. The second line contains three distinct integers p_1, p_2, p_3 ($2 \le p_i \le n$) – the indices of the Bear's favorite glades. Each of the next m lines describes a single road. The i-th of these lines contains three integers a_i, b_i, d_i ($1 \le a_i, b_i \le n$, $a_i \ne b_i$, $1 \le d_i \le 1\,000\,000$). They describe a road connecting glades a_i and b_i that takes d_i seconds to clean. Each pair of glades is connected by at most one road. Any two glades from the set $\{1, p_1, p_2, p_3\}$ can be reached from each other by walking through the roads.

Output

Output the minimum number of seconds required to remove snow from roads that allow moving between glades 1, p_1, p_2 and p_3.

Examples

Standard input	Standard output
7 8 3 6 7 1 2 2 1 4 3 2 3 4 3 5 3 3 7 5 4 5 3 4 6 5 5 7 4	18

Problem B. Rebroadcast

Input file:	`Standard input`
Output file:	`Standard output`
Time limit:	3 seconds
Memory limit:	512 mebibytes

Byteland's TV is going to rebroadcast the last match of their national volleyball team. The match will be broadcast in a traditional manner: a couple of plays, a pause for studio commentary (so that the invited experts can talk a bit), another couple of plays, another studio commentary...

On one hand, the station would like to show as many plays as possible (the longer the broadcast, the more advertisements are shown). On the other hand, the volleyball fans from Byteland are rather impatient: if Byteland's team is playing poorly in a given segment of the broadcast, the viewers might become discouraged and switch the channel to a soap opera. Therefore, it's best to show only a part of the match – a number of contiguous parts of the transmission, in every one of which Byteland's team wins more balls than it loses.

What is the maximum number of plays the TV station can show without losing their viewers' attention?

Input

The input contains a nonempty string consisting only of the uppercase characters `W` and `L`. The characters denote the results of consecutive plays: `W` denotes a ball won by the Byteland's team and `L` – a lost one. The string's length does not exceed 300 000.

Output

Output a single integer: the maximum number of plays the TV can show in accordance with the described rules.

Examples

Standard input	Standard output
WLWLLLWWLWLWLLWL	12

The TV station can show 12 plays by showing two contiguous segments: the plays 1-3 (two wins and one loss) and 7-15 (five wins and four losses).

Problem C. Hideouts

Input file:	`Standard input`
Output file:	`Standard output`
Time limit:	3.5 seconds
Memory limit:	512 mebibytes

The Byteland police has just conducted a raid on the leaders of the local mafia. Many of its members were arrested, but the main boss managed to escape. During the interrogations it was determined that he is most likely hiding in the mountains, on the flat like a plane, perfectly northern slope of the Bytelandian Sycamore [1] The mafia has built two hideouts there.

Unfortunately, the exact location of the hideouts remains unknown. The police only has a poor sketch, illustrating two paths between the hideouts. It consist of a simple polygon drawn on a sheet of paper. It is known that two of its vertices correspond to the hideouts, and the two broken lines between them are the paths that connect them.

The police also knows that one of the hideouts is located at a strictly greater height than the other. Moreover, the paths were constructed so that while moving from the hideout located higher to the one located lower, one does not encounters any segments going upwards. This allows quickly skiing from one hideout to the other in the winter.

It is not known which way points north on the given sketch. Your task is to determine the validity of the police officers' guesses.

Input

The first line of the input contains a single integer n ($3 \le n \le 10^5$) — the number of vertices of the polygon. The next n lines describe the vertices of the polygon, in order in which they appear on the polygon's boundary. The i-th vertex is described by two integers x_i, y_i ($|x_i|, |y_i| \le 10^9$) denoting its coordinates on the sheet of paper. The polygon has no self-intersections, and no two of its consecutive sides are parallel.

The next line of the input contains a single integer q ($1 \le q \le 10^5$) denoting the number of queries about the possible direction of the north on the sketch. Each of the next q lines describes a single query. Each query is described by two integers a_i and b_i ($|a_i|, |b_i| \le 10^9$). The nonzero vector (a_i, b_i) points to the potential north on the sketch.

Output

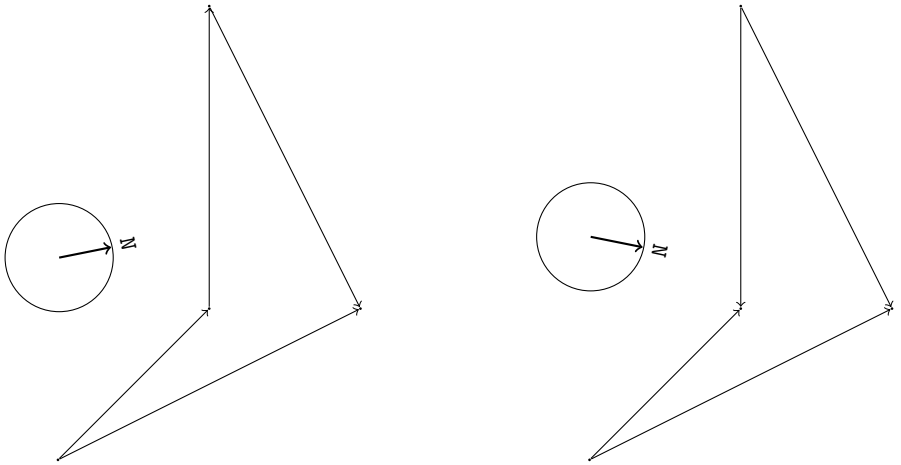
Output q lines. The i-th line should contain the word `YES`, if assuming the northern direction from the i-th query, there exist two vertices connected by paths going only down or to the sides. Otherwise, it should contain the word `NO`.

[1] That means its contour lines run exactly along the parallels and the height drops fastest towards the north.

Examples

Standard input	Standard output
4 0 0 1 1 1 3 2 1 2 5 1 5 -1	YES NO

Here are the sketches with the possible choices of the northern direction:



The hideouts for the first query would be located in the first and fourth vertices of the polygon.

Problem D. Multisets

Input file:	`Standard input`
Output file:	`Standard output`
Time limit:	30 seconds
Memory limit:	512 mebibytes

We say a multiset of positive integers $\mathcal{A}$ is *s-smaller* than a multiset $\mathcal{B}$ if the smallest integer x that has a different number of occurrences in the two multisets appears more times in the multiset $\mathcal{A}$.

For example, $\{1, 2, 3\}$ is *s-smaller* than $\{1, 3, 3, 5\}$ and $\{1, 1, 4, 4\}$ is *s-smaller* than $\{1, 1, 4\}$.

We are looking at an integer sequence S. Note that every contiguous subsequence of S induces a multiset (the set of its elements with each element occurring as many times as it does in the subsequence). Your task is to find the k-th *s-smallest* multiset induced by a nonempty contiguous subsequence of S.

Input

The first line of the standard input contains two integers n and k ($1 \le n \le 150\,000$, $1 \le k \le \frac{n(n+1)}{2}$), the length of S, and the index of the sought multiset. The second line consists of n positive integers, none of which exceeds 10^6 — the sequence S.

Output

Output a single line containing the multiset induced by the k-th *s-smallest* contiguous subsequence of S. To avoid ambiguity, output the elements of the multiset in ascending order.

Example

Standard input	Standard output
6 5 1 2 1 3 5 1	1 1 2

The multisets induced by the 21 nonempty contiguous subsequences of $1, 2, 1, 3, 5, 1$ are, in sorted order: $\{1, 1, 1, 2, 3, 5\}$, $\{1, 1, 2, 3, 5\}$, $\{1, 1, 2, 3, 5\}$, $\{1, 1, 2, 3\}$, $\{1, 1, 2\}$, $\{1, 1, 3, 5\}$, $\{1, 2, 3, 5\}$, $\{1, 2, 3\}$, $\{1, 2\}$, $\{1, 2\}$, $\{1, 3, 5\}$, $\{1, 3, 5\}$, $\{1, 3\}$, $\{1, 5\}$, $\{1\}$, $\{1\}$, $\{1\}$, $\{2\}$, $\{3, 5\}$, $\{3\}$, $\{5\}$.

Problem E. Screen

Input file:	Standard input
Output file:	Standard output
Time limit:	3 seconds
Memory limit:	512 mebibytes

Little Tomek is celebrating his birthday today and Wojtek gave him an exciting present – a small programmable computer with a monochrome display.

The screen of the device consists of $n \times m$ pixels – arranged in n rows and m columns. Each pixel can be either black or white. To manipulate the display, Tomek can issue only one kind of instruction: 'toggle the color of the pixel at position (x, y)'.

Tomek would like to paint a black disk (filled circle) on white background. The disk has to be centered at the midpoint of any fixed pixel and consists of all pixels whose midpoints are in distance at most r from the center, where r is any positive real number. All pixels that do not belong to the disk should be white. Additionally, the disk should be fully contained within the screen area – that is, if we painted a circle of the same size on an infinite screen, the number of cells painted black would be the same. Note that, in particular, this means that painting the whole screen black does not usually yield a proper disk.

Given the current contents of the display, compute the minimum number of instructions Tomek needs to issue to paint a disk.

Input

The first line of the input contains two integers n, m ($1 \le n, m \le 50$). The next n lines, with m characters each, describe the initial contents of Tomek's screen. The j-th character in the i-th of these lines describes the initial state of the pixel at position (j, i): '`#`' if the pixel is black and '`.`' if it is white.

Output

Output one number – the minimum number of toggles needed to make the screen display a disk.

Example

Standard input	Standard output
3 3 #.. ### .#.	2

An example disk that Tomek can paint with two instructions (flip $(1, 1)$ and flip $(2, 1)$) is:

```
.#.
###
.#.
```

This disk is centered at pixel $(2, 2)$ and can be assumed to have any r fulfilling $1 \le r < \sqrt{2}$.

Problem F. Crates

Input file:	`Standard input`
Output file:	`Standard output`
Time limit:	15 seconds
Memory limit:	512 mebibytes

The Bear has gathered s cubic units of honey and wants to rent out some space to store it. The price for renting the storage space is \$1 per square unit of base area. The honey has to be delivered in crates with no lids. Fortunately, the Bear is in possession of n empty crates with no lids, into which he can pour honey. They are all described by their height and base area. If h_i and a_i denote respectively the height and area of the i-th crate, then $h_i \cdot p_i$ cubic units of honey can be poured into it.

The base area of each crate is a power of two. If a crate has base area x, we can insert any number of other crates inside it, as long as the sum of their base areas does not exceed x. Of course if we choose to do that, we will be able to pour less honey directly into the bigger crate. For example, if we insert a crate of base area 1 and height 5 into a crate of base area 4 and height 3, we will be able to fill out the whole first crate, but we will be able to pour only $3 \cdot (4 - 1)$ cubic units of honey into the second one, since $1 \cdot 3$ units are already occupied by the first crate (and two units of height stick out above the first crate).

The Bear would like to know the minimum price he needs to pay to rent out storage space for all of his honey.

Input

The first line of the input contains two integers n and s ($1 \le n \le 100\,000$, $1 \le s \le 10^{18}$), denoting respectively the number of crates and the number of cubic units of honey the Bear has. The next n lines each contain two integers a_i and h_i ($1 \le a_i, h_i \le 1\,000\,000\,000$); a_i is equal to 2^k for some nonnegative integer k. The total volume of all the crates does not exceed 10^{18}, but is enough to store all of the collected honey.

Output

Output one integer: the minimum amount (in dollars) the Bear has to pay for storing the honey.

Example

Standard input	Standard output
3 5 1 2 1 2 2 1	3

The Bear rents out storage space with a base area of 3 square units. He pours 2 cubic units of honey into each of the first and second crates. Then he inserts the second crate into the third one and pours one unit of honey directly into the third crate.

Problem G. Popcount

Input file:	`Standard input`
Output file:	`Standard output`
Time limit:	2 seconds
Memory limit:	512 mebibytes

Bitek is fascinated by the *popcount* of numbers. The popcount of a natural number n is the number of 1 bits in the binary representation of n. For example, $popcount(23) = popcount(10111_2) = 4$.

By experimenting and playing with popcounts of different numbers, he arrived at the following theorem, which he subsequently managed to prove:

Theorem 1 (Bitek's theorem)**.** *For any $k \geq 1$, the popcount of any positive multiple of $2^k - 1$ is at least k. In other words, $2^k - 1$ has the smallest popcount out of all of its positive multiples.*

Proof. Left as an exercise for the reader. □

Bitek wants to generalize his theorem. In order to do that, he first needs an efficient way of finding the minimum popcount among the multiples of any natural number, not only those specified above.

Input

The first line of the input contains an integer n ($1 \leq n \leq 10^6$).

Output

Output the minimum popcount among multiples of n, i.e. the minimum possible value of $popcount(n \cdot m)$, for $m \geq 1$.

Example

Standard input	Standard output
11	2
63	6

The smallest multiple of 11 with a popcount of 2 is 33: $popcount(33) = popcount(100001_2) = 2$.
According to Bitek's theorem, no multiple of $63 = 2^6 - 1$ can have a popcount smaller than six.

Problem H. Gift boxes

Input file:	`Standard input`
Output file:	`Standard output`
Time limit:	3 seconds
Memory limit:	512 mebibytes

Anna bought a birthday gift for Jan Kanty. Right now her biggest concern is deciding how to pack it — she has some square and some round boxes, but they just... don't feel special enough. To make her present more memorable, she's going to pack the gift twice. A box a can be fit into the box b if it is possible for a to be inside b and not touch any of its sides. The gift is small enough to be put into any of the boxes. How many combinations of a gift-in-a-box-in-a-box are possible?

Input

The first line of the standard input consists of one integer N ($1 \le N \le 10^6$), the number of boxes at Anna's disposal. Each of the following lines consists of a character and an integer L ($1 \le L \le 10^9$). If the character is a `C`, the box is shaped like a circle of radius L. If the character is an `S`, the box is a square, and L denotes its side length.

Output

Output the number of two-box combinations in which one box can be put inside the other.

Example

Standard input	Standard output
6 C1 S1 S3 C3 S2 C2	13

Problem I. Coprime sets

Input file:	`Standard input`
Output file:	`Standard output`
Time limit:	5 seconds
Memory limit:	512 mebibytes

How many subsets of the set $\{1, 2, \ldots, n\}$ are coprime? A set of integers is called coprime if every two of its elements are coprime. Two integers are coprime if their greatest common divisor equals 1.

Input

The first line of input contains two integers n and m ($1 \le n \le 3000$, $1 \le m \le 10^9 + 10$).

Output

Output the number of coprime subsets of $\{1, 2, \ldots, n\}$, modulo m.

Example

Standard input	Standard output
4 7	5

There are 12 coprime subsets of $\{1, 2, 3, 4\}$: $\emptyset$, $\{1\}$, $\{2\}$, $\{3\}$, $\{4\}$, $\{1, 2\}$, $\{1, 3\}$, $\{1, 4\}$, $\{2, 3\}$, $\{3, 4\}$, $\{1, 2, 3\}$, $\{1, 3, 4\}$.

Problem J. Piggybank

Input file:	`Standard input`
Output file:	`Standard output`
Time limit:	7 seconds
Memory limit:	512 mebibytes

The Dragon stores his savings in a piggybank. For several days the following procedure took place. Each day the Dragon either put exactly one dollar into the piggybank or took a nonnegative integer amount of money from the piggybank. Now the Dragon wonders whether his savings could have been substantially larger if he had put more money into the piggybank.

More precisely, the Dragon asks himself the following type of questions: "If I decided to put an additional z dollars into the piggybank each day since the day number a, how large would have the largest amount of money in the period between days a and b been?" Help the Dragon resolve such scenarios.

Input

The first line of the input contains two integers n and m ($1 \le n, m \le 500\,000$), the number of days in the entire considered period of time and the number of queries. The second line contains n integers $s_1, s_2, \ldots, s_n$ that represent the amount of money in the piggybank after each day ($0 \le s_i \le s_{i-1} + 1$, and $s_0 = 0$).

m lines follow. The i-th of those lines describes a single query: three integers a_i, b_i and z_i ($1 \le a_i \le b_i \le n$, $1 \le z_i \le 10^9$) that represent the interval of days on which additional saving takes place and the amount of money saved each day. Note that the Dragon cannot alter the past and therefore each query should be considered separately.

Output

Output m lines with answers to the respective queries. Each line should contain a single integer: the largest amount of money in the piggybank between the days number a_i and b_i provided that the Dragon has decided to put z_i more dollars into the piggybank on each of those days.

Example

Standard input	Standard output
9 3 1 2 1 2 3 4 1 2 0 1 9 1 5 7 4 4 4 2	10 13 4

Problem K. Juggling

Input file:	`Standard input`
Output file:	`Standard output`
Time limit:	3 seconds
Memory limit:	512 mebibytes

The Dragon is learning how to juggle. When browsing the Internet, he learned a number of siteswaps. A siteswap is a sequence of numbers that describes a juggling pattern. More precisely, the sequence $a_1, a_2, \ldots, a_m$ is called a *siteswap* if and only if:

- the sequence contains only nonnegative integers;
- for each $1 \le i < j \le m$, the numbers $a_i + i$ and $a_j + j$ give a different remainder modulo m.

The number of balls used in a juggling pattern is equal to the arithmetic mean of the numbers in the siteswap. For example, the number of balls used in the siteswap $5, 1$ (called the shower siteswap) is $\frac{5+1}{2} = 3$.

We say that the sequence $a_1, a_2, \ldots, a_m$ has *a full period* $a_1, a_2, \ldots, a_k$ if $k|m$ and $a_i = a_{i+k}$ for $i \in \{1, 2, \ldots, m-k\}$. Two sequences are called *cyclically equivalent* if they have the same length and one of them can be obtained from the other by moving its prefix to its end, without altering the order of the elements in the prefix.

It might happen that two siteswaps describe the same juggling pattern. This happens when full periods of the two siteswaps are cyclically equivalent. For example, $7, 5, 3, 1$ describes the same juggling pattern as $3, 1, 7, 5, 3, 1, 7, 5$, since their full periods of length 4, that is, $7, 5, 3, 1$ and $3, 1, 7, 5$, are cyclically equivalent.

A siteswap describing a given juggling pattern is called *good* if it is the lexicographically greatest siteswap that describes this pattern. Thus each juggling pattern is described by exactly one good siteswap. For example, for the juggling pattern described by the siteswaps $1, 4, 4$ and $4, 1, 4, 4, 1, 4$, the only good siteswap is $4, 4, 1$.

The Dragon asked you to compute the number of good siteswaps that correspond to juggling patterns of length m performed using n balls. This number can be quite big, so output it modulo $10^9 + 33$.

Input

The only line of the input contains two integers n and m ($1 \le n \le 1\,000\,000$, $1 \le m \le 1\,000$), the number of balls and the length of the sequences.

Output

Output one integer, the number of good siteswaps modulo $10^9 + 33$.

Example

Standard input	Standard output
3 3	12

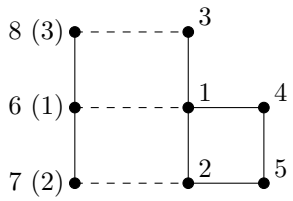
Here is the list of all good siteswaps for $n = 3$ and $m = 3$:

$$9,0,0 \quad 8,0,1 \quad 7,2,0 \quad 7,1,1 \quad 6,3,0 \quad 6,1,2 \quad 6,0,3 \quad 5,3,1 \quad 5,2,2 \quad 5,0,4 \quad 4,4,1 \quad 4,2,3.$$

Problem L. Fibonacci graphs

Input file:	`Standard input`
Output file:	`Standard output`
Time limit:	3 seconds
Memory limit:	512 mebibytes

Let us introduce the newest invention of the Bytelandian graph theorists: the Fibonacci graphs. The n-th Fibonacci graph G_n has F_n vertices numbered from 1 to F_n (the n-th Fibonacci number[1]). The graphs G_1 and G_2 are simply one vertex graphs, with the only vertex labeled as 1. For $n > 2$, the graph G_n is constructed by taking the previous two graphs (G_{n-2} and G_{n-1}) and adding edges between the equally labeled vertices. Hence G_n has F_n vertices and F_{n-2} more edges than G_{n-1} and $Gn - 2$ combined. Afterwards, the vertices from G_{n-2} are renumbered by adding F_{n-1} to each label. Here is an example of constructing G_6 from G_4 and G_5:



Given the labels of two vertices u and v of the graph G_n, find the length of the shortest path between them, and the remainder of division of the number of shortest paths between them by $10^9 + 7$.

Input

The first line of the input contains a single integer n ($1 \le n \le 2\,000$). The second line contains two integers u and v ($1 \le u, v \le F_n, u \ne v$).

Output

Output one line containing two integers: the distance between u and v in G_n, and the number of shortest paths between u and v in G_n (modulo $10^9 + 7$).

Example

Standard input	Standard output
6 5 8	4 5

There are five shortest paths between 5 and 8 in G_6: ($5 \to 2 \to 1 \to 3 \to 8$), ($5 \to 2 \to 1 \to 6 \to 8$), ($5 \to 2 \to 7 \to 6 \to 8$), ($5 \to 4 \to 1 \to 3 \to 8$), ($5 \to 4 \to 1 \to 6 \to 8$).

[1]The Fibonacci numbers F_n are defined as follows: $F_1 = F_2 = 1, F_n = F_{n-2} + F_{n-1}$ for $n > 2$.