Sponsored by Telegram

A. Vova and Train

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

Vova plans to go to the conference by train. Initially, the train is at the point 1 and the destination point of the path is the point L. The speed of the train is 1 length unit per minute (i.e. at the first minute the train is at the point 1, at the second minute — at the point 2 and so on).

There are lanterns on the path. They are placed at the points with coordinates divisible by v (i.e. the first lantern is at the point v, the second is at the point $2v$ and so on).

There is also **exactly** one standing train which occupies all the points from l to r inclusive.

Vova can see the lantern at the point p if p is divisible by v and there is no standing train at this position ($p \not\in [l; r]$). Thus, if the point with the lantern is one of the points covered by the standing train, Vova can't see this lantern.

Your problem is to say the number of lanterns Vova will see during the path. Vova plans to go to t different conferences, so you should answer t **independent** queries.

Input

The first line of the input contains one integer t ($1 \le t \le 10^4$) — the number of queries.

Then t lines follow. The i-th line contains four integers L_i, v_i, l_i, r_i ($1 \le L, v \le 10^9, 1 \le l \le r \le L$) — destination point of the i-th path, the period of the lantern appearance and the segment occupied by the standing train.

Output

Print t lines. The i-th line should contain one integer — the answer for the i-th query.

Example

input

```
4
10 2 3 7
100 51 51 51
1234 1 100 199
1000000000 1 1 1000000000
```

output

```
3
0
1134
0
```

Note

For the first example query, the answer is 3. There are lanterns at positions 2, 4, 6, 8 and 10, but Vova didn't see the lanterns at positions 4 and 6 because of the standing train.

For the second example query, the answer is 0 because the only lantern is at the point 51 and there is also a standing train at this point.

For the third example query, the answer is 1134 because there are 1234 lanterns, but Vova didn't see the lanterns from the position 100 to the position 199 inclusive.

For the fourth example query, the answer is 0 because the standing train covers the whole path.

B. Heaters

time limit per test: 1 second

memory limit per test: 256 megabytes

input: standard input

output: standard output

Vova's house is an array consisting of n elements (yeah, this is the first problem, I think, where someone *lives* in the array). There are heaters in some positions of the array. The i-th element of the array is 1 if there is a heater in the position i, otherwise the i-th element of the array is 0.

Each heater has a value r (r is the same for all heaters). This value means that the heater at the position pos can warm up all the elements in range $[pos - r + 1; pos + r - 1]$.

Vova likes to walk through his house while he thinks, and he hates cold positions of his house. Vova wants to switch some of his heaters on in such a way that each element of his house will be warmed up by at least one heater.

Vova's target is to warm up the whole house (all the elements of the array), i.e. if $n = 6$, $r = 2$ and heaters are at positions 2 and 5, then Vova can warm up the whole house if he switches all the heaters in the house on (then the first 3 elements will be warmed up by the first heater and the last 3 elements will be warmed up by the second heater).

Initially, all the heaters are off.

But from the other hand, Vova didn't like to pay much for the electricity. So he wants to switch the **minimum** number of heaters on in such a way that each element of his house is warmed up by at least one heater.

Your task is to find this number of heaters or say that it is impossible to warm up the whole house.

Input

The first line of the input contains two integers n and r $(1 \le n, r \le 1000)$ — the number of elements in the array and the value of heaters.

The second line contains n integers $a_1, a_2, \dots, a_n$ $(0 \le a_i \le 1)$ — the Vova's house description.

Output

Print one integer — the minimum number of heaters needed to warm up the whole house or -1 if it is impossible to do it.

Examples

input Copy

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

output Copy

```
3
```

input Copy

```
5 3
1 0 0 0 1
```

output Copy

```
2
```

input

```
5 10
0 0 0 0 0
```

output

```
-1
```

input

```
10 3
0 0 1 1 0 1 0 0 0 1
```

output

```
3
```

Note

In the first example the heater at the position 2 warms up elements $[1; 3]$, the heater at the position 3 warms up elements $[2, 4]$ and the heater at the position 6 warms up elements $[5; 6]$ so the answer is 3.

In the second example the heater at the position 1 warms up elements $[1; 3]$ and the heater at the position 5 warms up elements $[3; 5]$ so the answer is 2.

In the third example there are no heaters so the answer is `-1`.

In the fourth example the heater at the position 3 warms up elements $[1; 5]$, the heater at the position 6 warms up elements $[4; 8]$ and the heater at the position 10 warms up elements $[8; 10]$ so the answer is 3.

C. Books Queries

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

You have got a shelf and want to put some books on it.

You are given q queries of three types:

1. `L` id — put a book having index id on the shelf to the left from the leftmost existing book;
2. `R` id — put a book having index id on the shelf to the right from the rightmost existing book;
3. `?` id — calculate the minimum number of books you need to pop from the left or from the right in such a way that the book with index id will be leftmost or rightmost.

You can assume that the first book you will put can have any position (it does not matter) and queries of type 3 are always valid (it is guaranteed that the book in each such query is already placed). You can also assume that you don't put the same book on the shelf twice, so ids don't repeat in queries of first two types.

Your problem is to answer all the queries of type 3 in order they appear in the input.

Note that after answering the query of type 3 all the books remain on the shelf and the relative order of books does not change.

If you are Python programmer, consider using PyPy instead of Python when you submit your code.

Input

The first line of the input contains one integer q ($1 \le q \le 2 \cdot 10^5$) — the number of queries.

Then q lines follow. The i-th line contains the i-th query in format as in the problem statement. It is guaranteed that queries are always valid (for query type 3, it is guaranteed that the book in each such query is already placed, and for other types, it is guaranteed that the book was not placed before).

It is guaranteed that there is at least one query of type 3 in the input.

In each query the constraint $1 \le id \le 2 \cdot 10^5$ is met.

Output

Print answers to queries of the type 3 in order they appear in the input.

Examples

input Copy

```
8
L 1
R 2
R 3
? 2
```

```
L 4
? 1
L 5
? 1
```

output

Copy

```
1
1
2
```

input

Copy

```
10
L 100
R 100000
R 123
L 101
? 123
L 10
R 115
? 100
R 110
? 115
```

output

Copy

```
0
2
1
```

Note

Let's take a look at the first example and let's consider queries:

1. The shelf will look like $[1]$;
2. The shelf will look like $[1, 2]$;
3. The shelf will look like $[1, 2, 3]$;
4. The shelf looks like $[1, 2, 3]$ so the answer is 1;
5. The shelf will look like $[4, 1, 2, 3]$;
6. The shelf looks like $[4, 1, 2, 3]$ so the answer is 1;
7. The shelf will look like $[5, 4, 1, 2, 3]$;

8. The shelf looks like $[5, 4, 1, 2, 3]$ so the answer is 2.

Let's take a look at the second example and let's consider queries:

1. The shelf will look like $[100]$;
2. The shelf will look like $[100, 100000]$;
3. The shelf will look like $[100, 100000, 123]$;
4. The shelf will look like $[101, 100, 100000, 123]$;
5. The shelf looks like $[101, 100, 100000, 123]$ so the answer is 0;
6. The shelf will look like $[10, 101, 100, 100000, 123]$;
7. The shelf will look like $[10, 101, 100, 100000, 123, 115]$;
8. The shelf looks like $[10, 101, 100, 100000, 123, 115]$ so the answer is 2;
9. The shelf will look like $[10, 101, 100, 100000, 123, 115, 110]$;
10. The shelf looks like $[10, 101, 100, 100000, 123, 115, 110]$ so the answer is 1.

D. Boxes Packing

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

Maksim has n objects and m boxes, each box has size exactly k. Objects are numbered from 1 to n in order from left to right, the size of the i-th object is a_i.

Maksim wants to pack his objects into the boxes and he will pack objects by the following algorithm: he takes one of the empty boxes he has, goes from left to right through the objects, and if the i-th object fits in the current box (the remaining size of the box is greater than or equal to a_i), he puts it in the box, and the remaining size of the box decreases by a_i. Otherwise he takes the new empty box and continues the process above. If he has no empty boxes and there is at least one object not in some box then Maksim cannot pack the chosen set of objects.

Maksim wants to know the maximum number of objects he can pack by the algorithm above. To reach this target, **he will throw out the leftmost object from the set until the remaining set of objects can be packed in boxes he has**. Your task is to say the maximum number of objects Maksim can pack in boxes he has.

Each time when Maksim tries to pack the objects into the boxes, he will make empty all the boxes he has before do it (and the relative order of the remaining set of objects will not change).

Input

The first line of the input contains three integers n, m, k ($1 \le n, m \le 2 \cdot 10^5$, $1 \le k \le 10^9$) — the number of objects, the number of boxes and the size of each box.

The second line of the input contains n integers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le k$), where a_i is the size of the i-th object.

Output

Print the maximum number of objects Maksim can pack using the algorithm described in the problem statement.

Examples

input Copy

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

output Copy

```
4
```

input Copy

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

output Copy

```
1
```

input Copy

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

output Copy

```
5
```

Note

In the first example Maksim can pack only 4 objects. Firstly, he tries to pack all the 5 objects. Distribution of objects will be $[5], [2, 1]$. Maxim cannot pack the next object in the second box and he has no more empty boxes at all. Next he will throw out the first object and the objects distribution will be $[2, 1], [4, 2]$. So the answer is 4.

In the second example it is obvious that Maksim cannot pack all the objects starting from first, second, third and fourth (in all these cases the distribution of objects is $[4]$), but he can pack the last object ($[1]$).

In the third example Maksim can pack all the objects he has. The distribution will be $[1, 2], [3], [1, 1]$.

E. Binary Numbers AND Sum

time limit per test: 1 second

memory limit per test: 256 megabytes

input: standard input

output: standard output

You are given two huge binary integer numbers a and b of lengths n and m respectively. You will repeat the following process: if $b > 0$, then add to the answer the value $a \ \& \ b$ and divide b by 2 rounding down (i.e. remove the last digit of b), and repeat the process again, otherwise stop the process.

The value $a \ \& \ b$ means bitwise AND of a and b. Your task is to calculate the answer modulo 998244353.

Note that you should add the value $a \ \& \ b$ to the answer in decimal notation, not in binary. So your task is to calculate the answer in decimal notation. For example, if $a = 1010_2 \ (10_{10})$ and $b = 1000_2 \ (8_{10})$, then the value $a \ \& \ b$ will be equal to 8, not to 1000.

Input

The first line of the input contains two integers n and m ($1 \le n, m \le 2 \cdot 10^5$) — the length of a and the length of b correspondingly.

The second line of the input contains one huge integer a. It is guaranteed that this number consists of exactly n zeroes and ones and the first digit is always 1.

The third line of the input contains one huge integer b. It is guaranteed that this number consists of exactly m zeroes and ones and the first digit is always 1.

Output

Print the answer to this problem in decimal notation modulo 998244353.

Examples

input Copy

```
4 4
1010
1101
```

output Copy

```
12
```

input Copy

```
4 5
1001
10101
```

output Copy

```
11
```

Note

The algorithm for the first example:

1. add to the answer $1010_2 \& 1101_2 = 1000_2 = 8_{10}$ and set $b := 110$;
2. add to the answer $1010_2 \& 110_2 = 10_2 = 2_{10}$ and set $b := 11$;
3. add to the answer $1010_2 \& 11_2 = 10_2 = 2_{10}$ and set $b := 1$;
4. add to the answer $1010_2 \& 1_2 = 0_2 = 0_{10}$ and set $b := 0$.

So the answer is $8 + 2 + 2 + 0 = 12$.

The algorithm for the second example:

1. add to the answer $1001_2 \& 10101_2 = 1_2 = 1_{10}$ and set $b := 1010$;
2. add to the answer $1001_2 \& 1010_2 = 1000_2 = 8_{10}$ and set $b := 101$;
3. add to the answer $1001_2 \& 101_2 = 1_2 = 1_{10}$ and set $b := 10$;
4. add to the answer $1001_2 \& 10_2 = 0_2 = 0_{10}$ and set $b := 1$;
5. add to the answer $1001_2 \& 1_2 = 1_2 = 1_{10}$ and set $b := 0$.

So the answer is $1 + 8 + 1 + 0 + 1 = 11$.

F. Yet another 2D Walking

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

Maksim walks on a Cartesian plane. Initially, he stands at the point $(0, 0)$ and in one move he can go to any of four adjacent points (left, right, up, down). For example, if Maksim is currently at the point $(0, 0)$, he can go to any of the following points in one move:

- $(1, 0)$;
- $(0, 1)$;
- $(-1, 0)$;
- $(0, -1)$.

There are also n **distinct** key points at this plane. The i-th point is $p_i = (x_i, y_i)$. It is guaranteed that $0 \le x_i$ and $0 \le y_i$ and there is no key point $(0, 0)$.

Let the first level points be such points that $max(x_i, y_i) = 1$, the second level points be such points that $max(x_i, y_i) = 2$ and so on. Maksim wants to visit all the key points. But he shouldn't visit points of level $i + 1$ if he does not visit all the points of level i. He starts visiting the points from the minimum level of point from the given set.

The distance between two points (x_1, y_1) and (x_2, y_2) is $|x_1 - x_2| + |y_1 - y_2|$ where $|v|$ is the absolute value of v.

Maksim wants to visit all the key points in such a way that the total distance he walks will be minimum possible. Your task is to find this distance.

If you are Python programmer, consider using PyPy instead of Python when you submit your code.

Input

The first line of the input contains one integer n ($1 \le n \le 2 \cdot 10^5$) — the number of key points.

Each of the next n lines contains two integers x_i, y_i ($0 \le x_i, y_i \le 10^9$) — x-coordinate of the key point p_i and y-coordinate of the key point p_i. It is guaranteed that all the points are distinct and the point $(0, 0)$ is not in this set.

Output

Print one integer — the minimum possible total distance Maksim has to travel if he needs to visit all key points in a way described above.

Examples

input

Copy

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

output

Copy

```
15
```

input

Copy

```
5
2 1
1 0
2 0
3 2
0 3
```

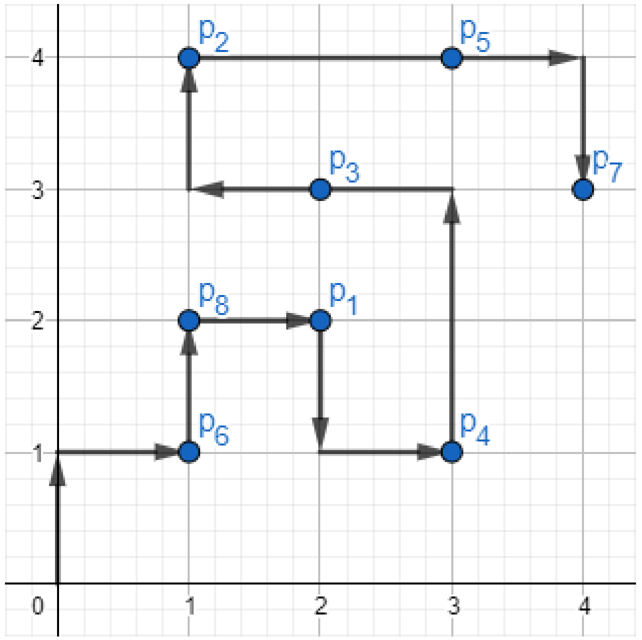
output

Copy

```
9
```

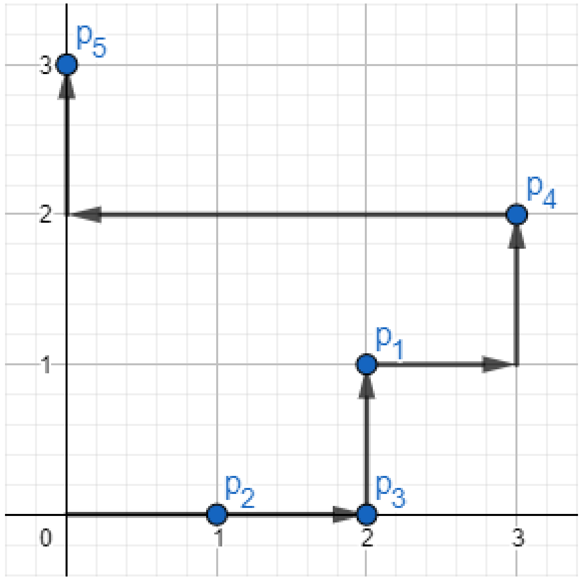
Note

The picture corresponding to the first example:



There is one of the possible answers of length 15.

The picture corresponding to the second example:



There is one of the possible answers of length 9.