A. From Hero to Zero

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

You are given an integer n and an integer k.

In one step you can do one of the following moves:

- decrease n by 1;
- divide n by k if n is divisible by k.

For example, if $n = 27$ and $k = 3$ you can do the following steps: $27 \rightarrow 26 \rightarrow 25 \rightarrow 24 \rightarrow 8 \rightarrow 7 \rightarrow 6 \rightarrow 2 \rightarrow 1 \rightarrow 0$.

You are asked to calculate the minimum number of steps to reach 0 from n.

Input

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

The only line of each query contains two integers n and k ($1 \le n \le 10^{18}$, $2 \le k \le 10^{18}$).

Output

For each query print the minimum number of steps to reach 0 from n in single line.

Example

input

```
2
59 3
1000000000000000000 10
```

output

```
8
19
```

Note

Steps for the first test case are: $59 \rightarrow 58 \rightarrow 57 \rightarrow 19 \rightarrow 18 \rightarrow 6 \rightarrow 2 \rightarrow 1 \rightarrow 0$.

In the second test case you have to divide n by k 18 times and then decrease n by 1.

B. Catch Overflow!

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

You are given a function f written in some basic language. The function accepts an integer value, which is immediately written into some variable x. x is an integer variable and can be assigned values from 0 to $2^{32} - 1$. The function contains three types of commands:

- for n — for loop;
- end — every command between "for n" and corresponding "end" is executed n times;
- add — adds 1 to x.

After the execution of these commands, value of x is returned.

Every "for n" is matched with "end", thus the function is guaranteed to be valid. "for n" can be immediately followed by "end"."add" command can be outside of any for loops.

Notice that "add" commands might overflow the value of x! It means that the value of x becomes greater than $2^{32} - 1$ after some "add" command.

Now you run $f(0)$ and wonder if the resulting value of x is correct or some overflow made it incorrect.

If overflow happened then output "OVERFLOW!!!", otherwise print the resulting value of x.

Input

The first line contains a single integer l ($1 \le l \le 10^5$) — the number of lines in the function.

Each of the next l lines contains a single command of one of three types:

- for n ($1 \le n \le 100$) — for loop;
- end — every command between "for n" and corresponding "end" is executed n times;
- add — adds 1 to x.

Output

If overflow happened during execution of $f(0)$, then output "OVERFLOW!!!", otherwise print the resulting value of x.

Examples

input

```
9
add
for 43
end
for 10
for 15
add
end
add
end
```

output

```
161
```

input

```
2
for 62
end
```

output

```
0
```

input

```
11
for 100
for 100
for 100
for 100
for 100
add
end
end
end
end
end
```

output

```
OVERFLOW!!!
```

Note

In the first example the first "add" is executed 1 time, the second "add" is executed 150 times and the last "add" is executed 10 times. Note that "for n" can be immediately followed by "end" and that "add" can be outside of any for loops.

In the second example there are no commands "add", thus the returning value is 0.

In the third example "add" command is executed too many times, which causes x to go over $2^{32} - 1$.

C. Electrification

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

At first, there was a legend related to the name of the problem, but now it's just a formal statement.

You are given n points $a_1, a_2, \dots, a_n$ on the OX axis. Now you are asked to find such an integer point x on OX axis that $f_k(x)$ is minimal possible.

The function $f_k(x)$ can be described in the following way:

- form a list of distances $d_1, d_2, \dots, d_n$ where $d_i = |a_i - x|$ (distance between a_i and x);
- sort list d in non-descending order;
- take d_{k+1} as a result.

If there are multiple optimal answers you can print any of them.

Input

The first line contains single integer T ($1 \le T \le 2 \cdot 10^5$) — number of queries. Next $2 \cdot T$ lines contain descriptions of queries. All queries are independent.

The first line of each query contains two integers n, k ($1 \le n \le 2 \cdot 10^5$, $0 \le k < n$) — the number of points and constant k.

The second line contains n integers $a_1, a_2, \dots, a_n$ ($1 \le a_1 < a_2 < \dots < a_n \le 10^9$) — points in ascending order.

It's guaranteed that $\sum n$ doesn't exceed $2 \cdot 10^5$.

Output

Print T integers — corresponding points x which have minimal possible value of $f_k(x)$. If there are multiple answers you can print any of them.

Example

input

```
3
3 2
1 2 5
2 1
1 1000000000
1 0
4
```

output

```
3
500000000
4
```

D. Array Splitting

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

You are given an array $a_1, a_2, \dots, a_n$ and an integer k.

You are asked to divide this array into k non-empty consecutive subarrays. Every element in the array should be included in exactly one subarray. Let $f(i)$ be the index of subarray the i-th element belongs to. Subarrays are numbered from left to right and from 1 to k.

Let the cost of division be equal to $\sum\limits_{i=1}^{n}(a_i \cdot f(i))$. For example, if $a = [1, -2, -3, 4, -5, 6, -7]$ and we divide it into 3 subbarays in the following way: $[1, -2, -3], [4, -5], [6, -7]$, then the cost of division is equal to $1 \cdot 1 - 2 \cdot 1 - 3 \cdot 1 + 4 \cdot 2 - 5 \cdot 2 + 6 \cdot 3 - 7 \cdot 3 = -9$.

Calculate the maximum cost you can obtain by dividing the array a into k non-empty consecutive subarrays.

Input

The first line contains two integers n and k ($1 \le k \le n \le 3 \cdot 10^5$).

The second line contains n integers $a_1, a_2, \dots, a_n$ ($|a_i| \le 10^6$).

Output

Print the maximum cost you can obtain by dividing the array a into k nonempty consecutive subarrays.

Examples

input

```
5 2
-1 -2 5 -4 8
```

output

```
15
```

input

```
7 6
-3 0 -1 -2 -2 -4 -1
```

output

```
-45
```

input

```
4 1
3 -1 6 0
```

output

```
8
```

E. Minimal Segment Cover

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

You are given n intervals in form $[l; r]$ on a number line.

You are also given m queries in form $[x; y]$. What is the minimal number of intervals you have to take so that every point (**not necessarily integer**) from x to y is covered by at least one of them?

If you can't choose intervals so that every point from x to y is covered, then print -1 for that query.

Input

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

Each of the next n lines contains two integer numbers l_i and r_i ($0 \le l_i < r_i \le 5 \cdot 10^5$) — the given intervals.

Each of the next m lines contains two integer numbers x_i and y_i ($0 \le x_i < y_i \le 5 \cdot 10^5$) — the queries.

Output

Print m integer numbers. The i-th number should be the answer to the i-th query: either the minimal number of intervals you have to take so that every point (**not necessarily integer**) from x_i to y_i is covered by at least one of them or -1 if you can't choose intervals so that every point from x_i to y_i is covered.

Examples

input

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

output

```
1
2
1
```

input

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

output

```
1
1
-1
-1
```

Note

In the first example there are three queries:

1. query $[1; 3]$ can be covered by interval $[1; 3]$;
2. query $[1; 4]$ can be covered by intervals $[1; 3]$ and $[2; 4]$. There is no way to cover $[1; 4]$ by a single interval;
3. query $[3; 4]$ can be covered by interval $[2; 4]$. It doesn't matter that the other points are covered besides the given query.

In the second example there are four queries:

1. query $[1; 2]$ can be covered by interval $[1; 3]$. Note that you can choose any of the two given intervals $[1; 3]$;
2. query $[1; 3]$ can be covered by interval $[1; 3]$;
3. query $[1; 4]$ can't be covered by any set of intervals;
4. query $[1; 5]$ can't be covered by any set of intervals. Note that intervals $[1; 3]$ and $[4; 5]$ together don't cover $[1; 5]$ because even non-integer points should be covered. Here 3.5, for example, isn't covered.

F. The Number of Subpermutations

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

You have an array $a_1, a_2, \dots, a_n$.

Let's call some subarray $a_l, a_{l+1}, \dots, a_r$ of this array a *subpermutation* if it contains all integers from 1 to $r-l+1$ exactly once. For example, array $a = [2, 2, 1, 3, 2, 3, 1]$ contains 6 subarrays which are subpermutations: $[a_2 \dots a_3]$, $[a_2 \dots a_4]$, $[a_3 \dots a_3]$, $[a_3 \dots a_5]$, $[a_5 \dots a_7]$, $[a_7 \dots a_7]$.

You are asked to calculate the number of subpermutations.

Input

The first line contains one integer n ($1 \le n \le 3 \cdot 10^5$).

The second line contains n integers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le n$).

This array can contain the same integers.

Output

Print the number of subpermutations of the array a.

Examples

input

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

output

```
7
```

input

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

output

```
6
```

Note

There are 7 subpermutations in the first test case. Their segments of indices are $[1, 4]$, $[3, 3]$, $[3, 6]$, $[4, 7]$, $[6, 7]$, $[7, 7]$ and $[7, 8]$.

In the second test case 6 subpermutations exist: $[1, 1]$, $[2, 2]$, $[2, 3]$, $[3, 4]$, $[4, 4]$ and $[4, 5]$.

G. Yet Another Partiton Problem

time limit per test: 5 seconds
memory limit per test: 512 megabytes
input: standard input
output: standard output

You are given array $a_1, a_2, \dots, a_n$. You need to split it into k subsegments (so every element is included in exactly one subsegment).

The weight of a subsegment $a_l, a_{l+1}, \dots, a_r$ is equal to $(r - l + 1) \cdot \max\limits_{l \le i \le r}(a_i)$. The weight of a partition is a total weight of all its segments.

Find the partition of minimal weight.

Input

The first line contains two integers n and k ($1 \le n \le 2 \cdot 10^4$, $1 \le k \le \min(100, n)$) — the length of the array a and the number of subsegments in the partition.

The second line contains n integers $a_1, a_2, \dots, a_n$ ($1 \le a_i \le 2 \cdot 10^4$) — the array a.

Output

Print single integer — the minimal weight among all possible partitions.

Examples

input

```
4 2
6 1 7 4
```

output

```
25
```

input
4 3 6 1 7 4

output
21

input
5 4 5 1 5 1 5

output
21

Note

The optimal partition in the first example is next: $6\ 1\ 7 \ \Big|\ 4$.

The optimal partition in the second example is next: $6 \ \Big|\ 1 \ \Big|\ 7\ 4$.

One of the optimal partitions in the third example is next: $5 \ \Big|\ 1\ 5 \ \Big|\ 1 \ \Big|\ 5$.