F. Yura and Developers

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

Yura has a team of k developers and a list of n tasks numbered from 1 to n. Yura is going to choose some tasks to be done this week. Due to strange Looksery habits the numbers of chosen tasks should be a segment of consecutive integers containing **no less than 2 numbers**, i. e. a sequence of form $l, l+1, ..., r$ for some $1 \le l < r \le n$.

Every task i has an integer number a_i associated with it denoting how many man-hours are required to complete the i-th task. Developers are not self-confident at all, and they are actually afraid of difficult tasks. Knowing that, Yura decided to pick up a hardest task (the one that takes the biggest number of man-hours to be completed, among several hardest tasks with same difficulty level he chooses arbitrary one) and complete it on his own. So, if tasks with numbers $[l, r]$ are chosen then the developers are left with $r - l$ tasks to be done by themselves.

Every developer can spend any integer amount of hours over any task, but when they are done with the whole assignment there should be exactly a_i man-hours spent over the i-th task.

The last, but not the least problem with developers is that one gets angry if he works more than another developer. A set of tasks $[l, r]$ is considered *good* if it is possible to find such a distribution of work that allows to complete all the tasks and to have every developer working for the same amount of time (amount of work performed by Yura doesn't matter for other workers as well as for him).

For example, let's suppose that Yura have chosen tasks with following difficulties: $a = [1, 2, 3, 4]$, and he has three developers in his disposal. He takes the hardest fourth task to finish by himself, and the developers are left with tasks with difficulties $[1, 2, 3]$. If the first one spends an hour on the first task and an hour on the third one, the second developer spends two hours on the second task and the third developer spends two hours on the third task, then they are done, since every developer worked exactly for two hours and every task has been worked over for the required amount of time. As another example, if the first task required two hours instead of one to be completed then it would be impossible to assign the tasks in a way described above.

Besides work, Yura is fond of problem solving. He wonders how many pairs (l, r) $(1 \le l < r \le n)$ exists such that a segment $[l, r]$ is *good*? Yura has already solved this problem, but he has no time to write the code. Please, help Yura and implement the solution for this problem.

Input

The first line of input contains two positive integers: n and k $(1 \le n \le 300\,000,\ 1 \le k \le 1\,000\,000)$, the number of tasks in the list and the number of developers in Yura's disposal.

The second line contains n integers a_i $(1 \le a_i \le 10^9)$.

Output

Output a single integer — the number of pairs (l, r) satisfying the conditions from the statement.

Examples

input

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

output

```
3
```

input

```
4 2
4 4 7 4
```

output

```
6
```

Note

In the first sample there are three good segments:

1. $[1;3]$ — the hardest task requires 3 man-hours, so there are tasks left that require 1 and 2 man-hours. A solution is to make first developer work on the first task for an hour, while second and third developers work on the second task. Each developer works exactly one hour.
2. $[1;4]$ — the hardest task requires 4 man-hours, so there are tasks left that require 1, 2 and 3 man-hours. If the first developer spends an hour on the first task and an hour on the third one, the second developer spends two hours on the second task and the third developer spends two hours on the third task, then they are done, since every developer worked exactly for two hours.
3. $[3;4]$ — the hardest task requires 4 man-hours, so there is only one task left that requires 3 man-hours. A solution is to make each developer work for an hour.