Subarrays and Magic

Mike is learning new concepts of programming.

Here are the things that he has learnt about subarrays.

1. A subarray is defined by an interval of indices: a first and a last element, and everything between them.
2. Total number of subarrays of an array of length 'n' are **(n*(n+1))/2**.

Mike loves the subarrays which have the following properties :

- The subarray must consist of at least 2 elements.
- It should start and end with the same element (say **X**) and have exactly **K** occurences of element **X** between the start and end elements (excluding start and end elements).

He loves these subarrays more than he loves his girlfriend.

Mike has a magical power but he can use it **at most M times**. He can change any element of the array to any integer using this magical power.

Mike wants to maximize the number of subarrays that he loves. But, he is not able to use his magical power efficiently. He asks for your help. Help Mike to maximize the number of subarrays he loves.

Input Format

First line of the input contains three space seperated integers **N K M**.

Second line contains **N** space seperated integers where **i^{th}** integer represents **Arr[i]**.

Constraints

$1 \leq N \leq 10^6$
$0 \leq K \leq N$
$0 \leq M \leq 10^9$
$0 \leq Arr[i] \leq 10^6$

Output Format

Single integer representing maximum number of subarrays that Mike loves.

Sample Input

6 0 1
1 2 1 2 3 1

Sample Output

4

Explanation

One of the ways to maximize the subarrays that Mike loves is to change the **5^{th} element (1 based indexing) to 2**.

The new array will be {1 , 2 , 1 , 2 , 2 , 1 }

All possible subarrays are :
{
{1}, {1,2}, **{1,2,1}**, {1,2,1,2}, {1,2,1,2,2 }, {1,2,1,2,2,1}
{2}, {2,1}, **{2,1,2}**, {2,1,2,2}, {2,1,2,2,1},
{1}, {1,2}, {1,2,2}, **{1,2,2,1}**,
{2}, **{2,2}**, {2,2,1},
{2}, {2,1},
{1}
}

Mike loves subarrays **{1,2,1}**, **{1,2,2,1}**, since they start and end with '**1**', and have exactly **0** occurences of '**1**' in between them.

Also subarrays **{2,1,2}** and **{2,2}** start and end with '**2**', and have exactly **0** occurences of '**2**' in between.

Hence, the output is **4**.