Birla Institute of Technology and Science, Pilani
Hyderabad Campus

CS F211 Data Structures and Algorithms
Lab 5: Recursion, Prefix Sum, Sliding Window, Bitmasking

Allowed languages: **C**

General Tips

- Indent your code appropriately and use proper variable names. These increase readability and writability of the code. Also, use comments wherever necessary.
- Use a proper IDE or text editors like Sublime Text or VSCode as they help to run and test your code on multiple test-cases easily. However, in the lab you will be using command line interface.
- Try to use functions as much as possible in your code. Functions increase reusability and the pass-by-value feature provides a significant help sometimes. Modularizing your code also helps you to debug efficiently.

Problem A. RBS

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 MB

A bracket sequence is any sequence of '(' and ')'. For example ()(), (()),)(are all bracket sequences. A bracket sequence is said to be **regular** if it is possible to obtain a correct arithmetic expression by inserting characters + and 1 into this sequence. For example, sequences (())(), () and (()(())) are regular, while)(, (() and (()))(are not.

You are given an even number n. Find all regular bracket sequences of length n.

Expected Time complexity of your solution — $O(2^n)$. Otherwise, it will not be considered.

Input

The first line of input contains n ($2 \leq n \leq 14$) — the length of the regular bracket sequence.

It is guaranteed that n is even.

Output

In the first line, print the number of generated bracket sequences, say k. Each of the following k lines should contain a single regular bracket sequence of length n (n given in the input). The bracket sequences can be printed in any order, but all possible sequences must be present.

Examples

standard input	standard output
4	2 ()() (())
6	5 ((())) (()()) (())() ()(()) ()()()

Problem B. Alternate Death

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 MB

You are found trespassing in King Ganon's favorite garden and are sent away to be executed. Executions in Ganon's prisons are very odd. If there are n prisoners to be executed, the prisoners are seated in a row, and then —

- The first prisoner is executed, then the third prisoner is executed, then the fifth, and so on.
- Once the n^{th} (if n is odd) or $(n-1)^{th}$ (if n is even) prisoner is executed, the process repeats, but this time it starts from the end and every alternate prisoner is executed.
- This process repeats till only one prisoner is left and s/he is set free.

You bribe the prison guard so that you can choose where to sit. Given the number of prisoners (including you), find which position to sit so that you can survive the execution. (Assume 1-based indexing)

Expected Time complexity of your solution — $O(log\, n)$**. Otherwise, it will not be considered.**

Input

Input consists of a single integer, n ($1 \le n \le 10^{18}$) — the number of prisoners (including yourself) to be executed.

Use **long long** to prevent errors due to integer overflow

Output

Print a single integer - the position where you have to sit inorder to survive the execution.

Example

standard input	standard output
11	8
12	6
10000000	6150102

Note

Assume 1-based indexing

In the first example, all the prisoners are alive initially - 1 2 3 4 5 6 7 8 9 10 11 (the numbers indicate the prisoners that are alive)

- After first round - 2 4 6 8 10
- After second round - 4 8
- After third round - 8

So, the prisoner in the 8^{th} position will be set free.

In the second example, all the prisoners alive initially - 1 2 3 4 5 6 7 8 9 10 11 12

- After first round - 2 4 6 8 10 12
- After second round - 2 6 10

- After third round - 6

So, the prisoner in the $6th$ position will be set free.

Problem C. Inclusive Substring

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 MB

You are given a string s of length n consisting of the first k lowercase letters only. Note that s may not contain all the k lowercase letters. Find the number of substrings of s that contain at least one occurrence of all the k letters.

Expected time complexity of your solution — $O(n * k)$. Otherwise, it will not be considered.

Input

The first line of the input consists of two integers, n ($1 \leq n \leq 10^5$) and k ($1 \leq k \leq 26$) — the length of the string and the number of alphabets that can be a part of the string. Assume 1-based indexing for the lowercase alphabets, i.e., a is letter 1, b is letter 2 and so on.

The second line of input consists of a string, s of length n.

Output

Print a single integer — the number of substrings that contain at least one occurrence of the k letters

Examples

standard input	standard output

`6 3`
`abcabc`
	`10`

`6 4`
`abaabc`
	`0`

Note

In the first example, the substrings that contain at least one occurrence of the first 3 letters are (the range in the parentheses below denote the start and end indices of the substring assuming 1-based indexing) *abc* (1-3), *bca* (2-4), *cab* (3-5), *abc* (4-6), *abca* (1-4), *bcab* (2-5), *cabc* (3-6) *abcab* (1-5), *bcabc* (2-6) and *abcabc* (1-6). Note that *abc* (1-3) and *abc* (4-6) are counted as 2 distinct substrings as their start and end indices are different. Also, note that a substring is formed by using the contiguous letters from the start index till the end index.

In the second example, no substring contains at least one occurrence of a, b, c, and d.

Problem D. Quizzers

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 MB

The annual inter-school quiz competition and SNMP international wants to send a team of size k to the competition. To do so, they conduct a quiz of their own for n of their students. The students are seated in a row and their scores are given in the form of an array. The school selects k students seated contiguously such that their average score is greater than or equal to a given threshold. Your task is to find **how many** such teams can be formed.

Note — Two teams, A and B are said to be different if and only if there is at least one student in Team A that is not there in Team B.

Expected Time Complexity of your solution — $O(n)$. Otherwise, it will not be considered.

Input

The first line of the input consists of three integers, n, k ($1 \le k \le n \le 10^5$) and t ($1 \le t \le 10^9$) — the number of students giving the quiz, the size of a team and the threshold respectively.

The second line of input contains n space separated integers $a_1, a_2, \ldots, a_n$ ($1 \le a_i \le 10^9$) — the scores of the students.

Output

Print a single integer — denoting the number of teams of size k (satisfying the above conditions) are possible. Each team should be exactly of size k.

Example

standard input	standard output
8 3 4 3 3 2 2 5 5 5 8	3
10 4 5 10 12 15 21 29 7 3 5 2 31	6

Note

Assume 1-based indexing.The teams are given in terms of the input array indices.

In the first example, the possible teams are — (4 - 6), (5 - 7), (6 - 8).

In the second example, the possible teams are — (1 - 4), (2 - 5), (3 - 6), (4 - 7), (5 - 8), (7 - 10)

Problem E. Dynamike's Dynamites

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 MB

Dynamike loves blowing things up, his latest obsession being dynamites. He has n dynamites arranged in a row, some of them are functional while the others are not. The functional dynamites are denoted by 1 while the non-functional dynamites are denoted by 0.

Dynamike can choose any one functional dynamite and blow it up. If there are any functional dynamites adjacent to it, they blow up as well, further blowing up dynamites adjacent to them creating a chain reaction. This continues till there are no functional dynamites adjacent to the ones that have blown up.

Dynamike can choose to fix at most k non-functional dynamites and make them functional. What is the maximum number of dynamites, Dynamike can blow up after fixing at most k non-functional dynamites?

Expected Time Complexity — $O(n)$. Otherwise, it will not be considered.

Input

The first line consists of two integers, n and k ($1 \leq k \leq n \leq 10^5$) - the total number of dynamites and the maximum number of non-functional dynamites that can be fixed.

The second line consists of n space-separated integers, $a_1, a_2, \ldots, a_n$ — Each a_i is either a 0 (for non-functional dynamites) or 1 (for functional dynamites)

Output

Print a single integer, the maximum number of dynamites, Dynamike can blow up.

Example

standard input	standard output

`11 2`
`1 1 1 0 0 0 1 1 1 1 0`
	`6`

`19 3`
`0 0 1 1 0 0 1 1 1 0 1 1 0 0 0 1 1 1 1`
	`10`

Note

Assume 1-based indexing

In the first example, if the 5th and 6th dynamite are fixed and then Dynamike starts by blowing up the 7th dynamite, then by the chain reaction explained in the statement, all dynamites from the 5th to the 10th positions blow up leading to a total of 6 dynamites that blow up. The same result could have been achieved by fixing the 6th and the 11th dynamite.

In the second example, if the dynamites in positions 5, 6 and 10 are fixed and dynamite at position 10 is blown up, then the chain reaction causes all dynamites from position 3 to position 12 to blow up, leading to a total of 10 dynamites blowing up.

Problem F. XOR of XOR Array

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 MB

Let C be the XOR array of two arrays A and B of size n and m respectively. The size of array C is nm and the i-th element of C is defined as follows:
$C_i = A_{\lfloor i/m \rfloor} \oplus B_{i\%m}$, $0 \leq i < nm$

Your task is to find the bitwise XOR of all the elements of C.

Note:

Here, $\oplus$ refers to the bitwise XOR operation.
For two integers $x \geq 0$ and $y > 0$,

- $\lfloor x/y \rfloor$ is the quotient of the Euclidean division of x by y
- $x\%y$ is the remainder of the Euclidean division of x by y

Assume 0-based indexing.

The time complexity of your solution should be $O(n + m)$. Any solution with a higher time complexity will not be considered.

Input

The first line of input contains n ($1 \leq n \leq 10^6$) — the size of the array A.

The second line of input contains n space separated integers $A_0, A_1, \ldots, A_{n-1}$ ($1 \leq A_i \leq 10^9$) — the elements of the array A. The elements of A may not be unique.

The third line of input contains m ($1 \leq m \leq 10^6$) — the size of the array B.

The fourth line of input contains m space separated integers $B_0, B_1, \ldots, B_{m-1}$ ($1 \leq B_i \leq 10^9$) — the elements of the array B. The elements of B may not be unique.

Output

Print one line containing the bitwise XOR of all the elements in C, the XOR array of A and B.

Examples

standard input	standard output

3
2 1 3
4
10 2 5 0
	13

Note

In the given example, the elements of C are: $[8, 0, 7, 2, 11, 3, 4, 1, 9, 1, 6, 3]$. Their bitwise XOR is 13.

Problem G. Fair Distribution of KFC

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 MB

You and your $(k-1)$ friends are at KFC and have just received your order. Your order consists of n chicken buckets, and the i-th bucket contains a_i chicken pieces. Now, you have to distribute the buckets amongst yourselves. Since none of you like sharing your food, **all the chicken pieces in the same bucket must go to the same person** in your group and cannot be split up. Also, all of you are against wastage of food so **every bucket must be given to some person** in your group.

The unfairness of a distribution is defined as the maximum total chicken pieces obtained by a single person in the distribution.
Return the minimum unfairness of all distributions.

Assume 1-based indexing.

The time complexity of your solution should be $O(k^n)$. Any solution with a higher time complexity will not be considered.

Input

The first line of input contains two integers, n and k $(2 \leq k \leq n \leq 8)$ — the number of chicken buckets in your order and the total number of people in your group (including you), respectively.

The second line of input contains n space-separated integers $a_1, a_2, \ldots, a_n$ $(1 \leq a_i \leq 10^6)$ — the number of chicken pieces in each of the n buckets. These n integers may not be unique.

Output

Print one line containing the minimum unfairness of all distributions.

Examples

standard input	standard output

`5 2`
`8 15 10 20 8`
	`31`

`5 2`
`8 15 10 20 8`
	`31`

Note

In the given example, one optimal distribution is $[8, 15, 8]$ and $[10, 20]$:

- The 1st person receives buckets 1, 2 and 5, which has a total of $8 + 15 + 8 = 31$ chicken pieces.
- The 2nd person receives buckets 3 and 4, which has a total of $10 + 20 = 30$ chicken pieces.

The unfairness of this distribution is $max(31, 30) = 31$.

Let us consider another possible distribution, $[8, 10, 8]$ and $[15, 20]$:

- The 1st person receives buckets 1, 3 and 5, which has a total of $8 + 10 + 8 = 26$ chicken pieces.
- The 2nd person receives buckets 2 and 4, which has a total of $15 + 20 = 35$ chicken pieces.

The unfairness of this distribution is $max(26, 35) = 35$, which is greater than the previous distribution.

In fact, it can be shown that there is no distribution with an unfairness less than 31.

Problem H. Data Trouble

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 MB

After CCIT imposed a daily data limit, you have gained unauthorized access to their servers in an attempt to bypass the limit. You have a data array of size n, whose elements are the amount of data you can add to your daily limit (in Gigabytes). You can take any subarray of the data array and add it to your daily limit, as long as it is *safe*. A subarray is *safe* if it satisfies both of these conditions:

- it has at least one element
- the sum of the elements of the subarray is a multiple of a given integer k

Count the total number of *safe* subarrays of the data array.

Note

- A subarray is a contiguous part of the array.
- An integer x is a multiple of k if there exists an integer n such that $x = nk$

The answer may not fit into 32-bit integer type, so you should use `long long` to avoid integer overflow.

Assume 1-based indexing.

The time complexity of your solution should be $O(n)$. Any solution with a higher time complexity will not be considered.

Input

The first line of input contains two integers, n and k $(1 \leq n, k \leq 10^6)$.

The second line of input contains n space separated integers $d_1, d_2, \ldots, d_n$ $(1 \leq d_i \leq 10^9)$ — the elements of the data array. The elements of the data array may not be unique.

Output

Print one line containing the total number of *safe* subarrays of the data array.

Examples

standard input	standard output

`5 6`
`23 2 4 6 7`
	`4`

Note

In the given example,

- $[1, 5]$ is a *safe* subarray as $d_1 + d_2 + d_3 + d_4 + d_5 = 23 + 2 + 4 + 6 + 7 = 42$, which is a multiple of 6
- $[2, 3]$ is a *safe* subarray as $d_2 + d_3 = 2 + 4 = 6$, which is a multiple of 6
- $[2, 4]$ is a *safe* subarray as $d_2 + d_3 + d_4 = 2 + 4 + 6 = 12$, which is a multiple of 6
- $[4, 4]$ is a *safe* subarray as $d_4 = 6$, which is a multiple of 6

There are no other *safe* subarrays of the given data array.

Problem I. Count Kool Subarrays

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 MB

You are given an array A of n integers and an integer k. A subarray of this array is called *kool* if there are exactly k odd numbers in it.

Count the total number of *kool* subarrays of A.

Note

- A subarray is a contiguous part of the array.

The answer may not fit into 32-bit integer type, so you should use `long long` to avoid integer overflow.

Assume 1-based indexing.

The time complexity of your solution should be $O(n)$. Any solution with a higher time complexity will not be considered.

Input

The first line of input contains two integers, n and k ($1 \le n, k \le 10^6$). k can be less than, equal to or even greater than n.

The second line of input contains n space separated integers $A_1, A_2, \ldots, A_n$ ($1 \le A_i \le 10^9$) — the elements array. The elements of A may not be unique.

Output

Print one line containing the total number of *kool* subarrays of array A.

Examples

standard input	standard output
5 3 1 1 2 1 1	2

Note

In the given example,

- $[1, 4]$ is a *kool* subarray as A_1, A_2 and A_4 are the 3 odd numbers in the subarray
- $[2, 5]$ is a *kool* subarray as A_2, A_4 and A_5 are the 3 odd numbers in the subarray

There are no other *kool* subarrays of the given data array.

Problem J. Random Online Number

Input file:	`standard input`
Output file:	`standard output`
Time limit:	1 second
Memory limit:	256 MB

Come up with an online algorithm which reads integers from the input one-at-a-time and prints a random integer from all the integers in the input till then with equal probability. Your algorithm should not use more than $O(1)$ extra space.

Note

You cannot use any data structure to solve this problem. You cannot store the input sequence in an array or any other data structure. Your program should be interactive. You must read the integers from the sequence one-at-a-time, and print the required output after reading every integer. Your solution need not be deterministic.

Input

The first line contains n — the number of integers in the sequence.

The following n lines contain one integer each — an element of the input sequence. It is guaranteed that all the integers in the sequence are distinct.

Output

For each element of the input sequence, print one integer from all the integers in the sequence till that integer with equal probability on a new line.

Example

standard input	standard output
5 10 -57 21 109 0	10 10 21 109 109

Note

The output may differ but it should satisfy the following conditions:

1. After reading the 1st integer, your algorithm must print 10 with probability $\frac{1}{1}$
2. After reading the 2nd integer, your algorithm must print one of $10, -57$ with probability $\frac{1}{2}$
3. After reading the 3rd integer, your algorithm must print one of $10, -57, 21$ with probability $\frac{1}{3}$
4. After reading the 4th integer, your algorithm must print one of $10, -57, 21, 109$ with probability $\frac{1}{4}$
5. After reading the 5th integer, your algorithm must print one of $10, -57, 21, 109, 0$ with probability $\frac{1}{5}$