Birla Institute of Technology and Science, Pilani
Hyderabad Campus

CS F211 Data Structures and Algorithms
Lab 9

Allowed languages: **C, 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.
- You can implement your solutions in C or C++. Using C++ STL (Standard Template Library) is allowed.

Problem A. Cold Drink Bash

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

Coca Cola is organizing an event at your college fest in which you are participating. There are n Coca Cola bottles in a line, with bottle 1 on the far left and bottle n on the far right. The i-th bottle will increase your happiness by a_i when drunk. a_i can be negative, meaning that drinking that bottle will decrease your happiness. You start with 0 happiness and you will walk from left to right, from the first bottle to the last one. At each bottle, you may choose to drink it or ignore it. You must ensure that your happiness is always non-negative.

What is the largest number of Coca Cola bottles you can drink?

Note:

Assume 1-based indexing.

The time complexity of your solution should be $O(nlogn)$, where n is the number of Coca Cola bottles. Solution of higher time complexity will not be considered.

Input

The first line of input contains n ($1 \leq n \leq 10^5$) — the number of Coca Cola bottles.

The second line of input contains n space separated integers $a_1, a_2, \ldots, a_n$ ($-10^9 \leq a_i \leq 10^9$) — the change in happiness after drinking each bottle.

Output

Print one line containing the maximum number of Coca Cola bottles you can drink without your happiness becoming negative.

Example

standard input	standard output
6 4 -4 1 -3 1 -3	5

Explanation

In the given example, you can

- drink bottle 1, your happiness is now 4, which is non-negative
- skip bottle 2
- drink bottle 3, your happiness is now $4 + 1 = 5$, which is non-negative
- drink bottle 4, your happiness is now $4 + 1 - 3 = 2$, which is non-negative
- drink bottle 5, your happiness is now $4 + 1 - 3 + 1 = 3$, which is non-negative
- drink bottle 6, your happiness is now $4 + 1 - 3 + 1 - 3 = 0$, which is non-negative

It can be shown that it is not possible to drink more than 5 bottles in this example.

Problem B. Plates between Candles

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

There is a long table with a line of plates and candles arranged on top of it. You are given a string s of length n consisting of characters '*' and '|' only, where a '*' represents a plate and a '|' represents a candle. You are also given q queries, where $query_i = [left_i, right_i]$ denotes the substring $s_{left_i \ldots right_i}$ (inclusive).

For each query, you need to find the number of plates between candles that are in the substring. A plate is considered between candles if there is at least one candle to its left and at least one candle to its right in the substring. Note that the candles need not necessarily be on the immediate left/right of the plates. Also, the query intervals may not start/end with a candle.

Note:

Assume 0-based indexing.

The time complexity of your solution should be $O(n+q)$, where n is the length of string s and q is the number of queries. Solution of higher time complexity will not be considered.

Input

The first line of input contains two integers, n and q ($1 \leq n, q \leq 10^6$) — the length of string s and the number of queries, respectively. n can be less than or equal to or greater than q.

The second line of input contains the string s.

The following q lines contain the queries. The i-th line contains two space separated integers, $left_i$ and $right_i$ ($0 \leq left_i \leq right_i < n$) — the i-th query.

Output

For each query, print the number of plates between candles that are in the substring on a new line.

Examples

standard input	standard output
10 1 ||**||**|* 3 8	2
10 2 **|**|***| 2 5 5 9	2 3
11 1 ||**|*||*|| 0 7	3

Explanation

In example 1, the substring for the given query is "*||**|". The number of plates between candles in this substring is 2 (underlined in the substring), as each of the two plates has at least one candle **in the substring** and to its left **and** right.

Problem C. Sum of Subarray Ranges

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. The range of a subarray of a is the difference between the largest and smallest elements in the subarray. Find the sum of all subarray ranges of a.

Note:

Assume 1-based indexing.

The time complexity of your solution should be $O(n)$. Solution of 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 $a_1, a_2, \ldots, a_n$ ($-10^9 \leq a_i \leq 10^9$) — the elements of array a.

Output

Print one line containing the sum of all subarray ranges of a.

Examples

standard input	standard output
3 4 5 6	4

Explanation

The 6 subarrays are the following:

- [4], range = largest - smallest = 4 - 4 = 0
- [5], range = 5 - 5 = 0
- [6], range = 6 - 6 = 0
- [4, 5], range = 5 - 4 = 1
- [5, 6], range = 6 - 5 = 1
- [4, 5, 6], range = 6 - 4 = 2

So the sum of all ranges is $0 + 0 + 0 + 1 + 1 + 2 = 4$.

Problem D. Average Problem

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. Your task is to compute the number of subarrays of a with an arithmetic mean equal to k.

Note:

The arithmetic mean (average) of a subarray is the sum of the elements of the subarray divided by its length. For example, the arithmetic mean of $[1, 4, 4, 5]$ is $14/4 = 3.5$.

A subarray is a contiguous non-empty sequence of elements within an array.

The time complexity of your solution should be $O(nlogn)$. Solution of higher time complexity will not be considered.

Input

The first line of the input contains two integers, n ($1 \leq n \leq 10^6$) and k ($-10^9 \leq k \leq 10^9$) — the size of array a and the required arithmetic mean, respectively.

The second line of the input contains n space separated integers $a_1, a_2, \ldots, a_n$ ($-10^9 \leq a_i \leq 10^9$) — the elements of array a.

Output

Print one line containing the number of subarrays of a with an arithmetic mean equal to k.

Example

standard input	standard output
3 2 2 1 3	3

Explanation

The following subarrays have mean 2: $[2]$, $[1, 3]$, $[2, 1, 3]$.

Problem E. Sparse Matrix Multiplication

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

Given two sparse matrices `mat1` and `mat2` of sizes $m \times k$ and $k \times n$ respectively, find the matrix `mat3` = `mat1` $\times$ `mat2`.

Note:

Assume 1-based indexing.

Input

The first line of input contains three integers m, k and n ($1 \leq m, k, n \leq 100$).

The second line of the input contains p — the number of non-zero elements in `mat1`.

Each line in the following p lines contains three space separated integers (details regarding each non-zero element of `mat1`) — i ($1 \leq i \leq m$), j ($1 \leq j \leq k$) and `mat1[i][j]` (`mat1[i][j]` $\neq 0$).

The next line of input contains q — the number of non-zero elements in `mat2`.

Each line in the following q lines contains three space separated integers (details regarding each non-zero element of `mat2`) — s ($1 \leq s \leq k$), t ($1 \leq t \leq n$) and `mat2[s][t]` (`mat2[s][t]` $\neq 0$).

Output

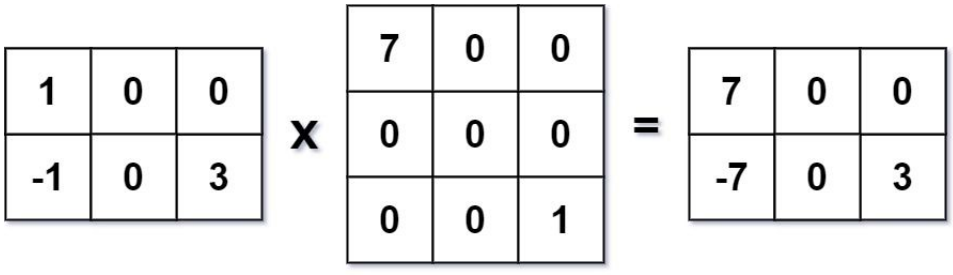
The first line of output should contain r — the number of non-zero elements in `mat3`.

The following r lines of output should contain details of the non-zero elements of `mat3`. For each non-zero element of `mat3`, print three space separated integers i ($1 \leq i \leq m$), j ($1 \leq j \leq n$) and `mat3[i][j]` (`mat3[i][j]` $\neq 0$).

Examples

standard input	standard output
2 3 3 3 1 1 1 2 1 -1 2 3 3 2 1 1 7 3 3 1	3 1 1 7 2 1 -7 2 3 3

Explanation



Problem F. Distant Points

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

You are given an array of size n, representing the points on the X-axis on the Cartesian coordinate plane. In one operation, you can select any two points x_i and x_j, $(i \neq j)$ from the array such that the distance between the two points is at least k and remove the two points from the array.

Find the maximum number of operations you can perform.

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

Input

The first line consists of two integers n $(1 \leq n \leq 10^5)$ and k $(1 \leq k \leq 10^9)$ — the number of points on the x-axis and the minimum permissible distance between the points.

The second line of input contains n space-separated integers $a_1, a_2, \ldots, a_n$ $(-10^9 \leq a_i \leq 10^9)$ — a_i represents the x-coordinate of the i^{th} point

Output

Print the maximum number of operations you can perform.

Examples

standard input	standard output
6 4 6 2 4 3 1 7	2
7 5 1 3 4 6 4 5 8	2

Note

In the first example, we can remove the points with $x = 1$ and $x = 7$, and can also remove the points with $x = 2$ and $x = 6$. We will be left with the points having x-coordinates — 4, 3 since the distance between them is less than 4 (k) we cannot perform further operations. (Note we can remove 6,2 and 7,3 also.)

In the second example, we first remove the points with $x = 1$ and $x = 6$, then we remove the points with $x = 3$ and $x = 8$. We are left with points having x-coordinates — 4, 4, and 5 since the distance between any two points is less than 5 (k), we cannot perform further operations.

Problem G. Integer Concatenation

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

You are given an array of size n consisting of **positive integers only**. Find the largest possible integer that can be formed by concatenating these n integers in some order.

The concatenation of integers works similarly to the concatenation of strings. For example, concatenating 10 and 23 in that order gives 1023.

Note that the resultant integer may not fit in any of the basic data types (*int*, *long long*, etc) so use *string* or *char* array to store the final answer.

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

Input

The first line consists of a single integer n ($1 \leq n \leq 10^5$) — the number of integers in the array

The second line of input contains n space-separated integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq 10^{18}$) — the elements of the array

Output

Print the largest integer that can be formed.

Examples

standard input	standard output
5 3 30 34 5 9	9534330
4 94 546 548 9	994548546

Note

In the first example, the order of concatenation is - 9 5 34 3 30

In the second example, the order of concatenation is - 9 94 548 546

Problem H. Reading Hour

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

The College library has n books. The i^{th} book has a_i number of pages. You have to allocate books to m number of students such that —

- A book will be allocated to exactly one student (i.e every book is allocated).
- Each student has to be allocated at least one book.
- Allotment should be in contiguous order, for example, A student cannot be allocated book 1 and book 3, skipping book 2.

There can be many ways or permutations to do so. In each permutation, one of the m students will be allocated the maximum number of pages. Out of all these permutations, the task is to find that particular permutation in which the maximum number of pages allocated to a student is the minimum of those in all the other permutations and print this minimum value

The time complexity of your solution should be $O(n * log(total\_number\_of\_pages))$

Input

The first line consists of two integers n and m ($1 \leq m \leq n \leq 10^5$) — the number of books and the number of students respectively.

The second line of input contains n space-separated integers $a_1, a_2, \ldots, a_n$ ($1 \leq a_i \leq 10^9$) — a_i represents the number of pages in the i^{th} book.

Output

Print the minimum value.

Examples

standard input	standard output
4 2 22 44 77 100	143
4 4 15 27 110 27	110

Note

In the first example, Allocation can be done in the following ways:

- {22} and {44, 77, 100} Maximum Pages = 221
- {22, 44} and {77, 100} Maximum Pages = 177
- {22, 44, 77} and {100} Maximum Pages = 143.

Therefore, the minimum of these cases is 143, which is selected as the output.

In the second example, each student gets one book.

Problem I. Circular Drive

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

There are n petrol stations along a circular route, where the amount of petrol at the i^{th} station is a_i. You have a car with an unlimited petrol tank and it costs b_i amount of petrol to travel from i^{th} station to the $(i+1)^{th}$ station. You begin the journey with an empty tank at one of the petrol stations.

Find the minimum index (assume 0-based indexing) of the petrol station from which you can start your journey such that you can travel around the circuit once. Print -1 if such a starting petrol station does not exist. Note that you can only travel in one direction. That is, the order in which you visit petrol stations should be i, $i+1$, $i+2$, $\ldots n-1$, 0, 1, 2 $\ldots$, $i-1$ where $0 \leq i \leq n-1$. Completing the circuit means starting at i and ending up at i again.

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 consists of a single integer, n $(1 \leq n \leq 10^6)$— the number of petrol stations

The second line of input contains n space-separated integers $a_1, a_2, \ldots, a_n$ $(1 \leq a_i \leq 10^5)$ — a_i represents the amount of petrol in the i^{th} petrol station.

The third line of input contains n space-separated integers $b_1, b_2, \ldots, b_n$ $(1 \leq b_i \leq 10^5)$ — b_i represents the amount of petrol required to travel from the i^{th} to the $(i+1)^{th}$ petrol station.

Output

Print the minimum starting petrol station's index if you can travel around the circuit once, otherwise print -1

Examples

standard input	standard output
4 4 6 7 4 6 5 3 5	1
4 55 52 33 100 77 61 40 69	-1

Note

In the first example, we cannot start from the pump with index 0 as we can only fill 4 units of petrol and we need 6 units to go to the next station. We can start from the pump with index 1 and complete the circuit.

In the second example, no matter what petrol station we start from, we will not be able to travel around the circuit.

Problem J. Train Schedule

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

You are given the arrival and departure times of n trains in the railway station you are managing. You need to find the minimum number of platforms required for the railway station so that no train is kept waiting.

- Consider that all the trains arrive on the same day and leave on the same day.
- Arrival and departure time can never be the same for a train but we can have the arrival time of one train equal to the departure time of the other.
- At any given instance of time, the same platform can not be used for both departure of a train and the arrival of another train. In such cases, we need different platforms.

The arrival and departure times are given in the 24-hour format, so if a train departs at 12:30 AM, it will be represented as 0030, similarly, if a train departs at 2:45 PM it is represented as 1445.

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

Input

The first line consists of a single integer n ($1 \leq n \leq 10^5$) — the number trains.

The next n lines consist of a pair of integers separated by a single space. That is the i^{th} line out of n lines consists of a_i d_i ($1 \leq a_i < d_i \leq 10^9$) where a_i and d_i represent the arrival and departure times of the i^{th} train respectively. It is guaranteed that the given arrival and departure times for every train will be valid.

Output

Print the minimum number of platforms required so that no train is kept waiting.

Example

standard input	standard output
5 1000 1010 1040 1300 1040 1230 1200 1230 1600 2000	3
3 1000 1100 1200 1300 1335 1340	1

Note

In the first example, between 12:00 and 12:30, there are 3 trains —

- Train arriving at 10:40 and leaving at 13:00
- Train arriving at 10:40 and leaving at 12:30

- Train arriving at 12:00 and leaving at 12:30

at the station, so we need at least 3 platforms.

In the second example, there is no overlap between the trains so 1 platform will suffice.