1.

Given an array of N integers arr[] where each element represents the maximum length of the jump that can be made forward from that element. This means if arr[i] = x, then we can jump any distance y such that y ≤ x.

Find the minimum number of jumps to reach the end of the array (starting from the first element). If an element is 0, then you cannot move through that element.

Note: Return -1 if you can't reach the end of the array.

Example 1:

Input:

N = 11

arr[] = {1, 3, 5, 8, 9, 2, 6, 7, 6, 8, 9}

Output: 3

Explanation:

First jump from 1st element to 2nd

element with value 3. Now, from here

we jump to 5th element with value 9,

and from here we will jump to the last.

Example 2:

Input :

N = 6

arr = {1, 4, 3, 2, 6, 7}

Output: 2

Explanation:

First we jump from the 1st to 2nd element

and then jump to the last element.

Your task:

You don't need to read input or print anything. Your task is to complete function minJumps() which takes the array arr and it's size N as input parameters and returns the minimum number of jumps. If not possible return -1.

Expected Time Complexity: O(N)

Expected Space Complexity: O(1)

Constraints:

1 ≤ N ≤ 107

0 ≤ arri ≤ 107

2.

Given an array arr[] and an integer K where K is smaller than size of array, the task is to find the Kth smallest element in the given array. It is given that all array elements are distinct.

Example 1:

Input:

N = 6

arr[] = 7 10 4 3 20 15

K = 3

Output : 7

Explanation :

3rd smallest element in the given

array is 7.

Example 2:

Input:

N = 5

arr[] = 7 10 4 20 15

K = 4

Output : 15

Explanation :

4th smallest element in the given

array is 15.

Your Task:

You don't have to read input or print anything. Your task is to complete the function kthSmallest() which takes the array arr[], integers l and r denoting the starting and ending index of the array and an integer K as input and returns the Kth smallest element.

Expected Time Complexity: O(n)

Expected Auxiliary Space: O(log(n))

Constraints:

1 <= N <= 105

1 <= arr[i] <= 105

1 <= K <= N

3.

Given an array arr[] and an integer K where K is smaller than size of array, the task is to find the Kth smallest element in the given array. It is given that all array elements are distinct.

Example 1:

Input:

N = 6

arr[] = 7 10 4 3 20 15

K = 3

Output : 7

Explanation :

3rd smallest element in the given

array is 7.

Example 2:

Input:

N = 5

arr[] = 7 10 4 20 15

K = 4

Output : 15

Explanation :

4th smallest element in the given

array is 15.

Your Task:

You don't have to read input or print anything. Your task is to complete the function kthSmallest() which takes the array arr[], integers l and r denoting the starting and ending index of the array and an integer K as input and returns the Kth smallest element.

Expected Time Complexity: O(n)

Expected Auxiliary Space: O(log(n))

Constraints:

1 <= N <= 105

1 <= arr[i] <= 105

1 <= K <= N

4.

Given an array arr[] and an integer K where K is smaller than size of array, the task is to find the Kth smallest element in the given array. It is given that all array elements are distinct.

Example 1:

Input:

N = 6

arr[] = 7 10 4 3 20 15

K = 3

Output : 7

Explanation :

3rd smallest element in the given

array is 7.

Example 2:

Input:

N = 5

arr[] = 7 10 4 20 15

K = 4

Output : 15

Explanation :

4th smallest element in the given

array is 15.

Your Task:

You don't have to read input or print anything. Your task is to complete the function kthSmallest() which takes the array arr[], integers l and r denoting the starting and ending index of the array and an integer K as input and returns the Kth smallest element.

Expected Time Complexity: O(n)

Expected Auxiliary Space: O(log(n))

Constraints:

1 <= N <= 105

1 <= arr[i] <= 105

1 <= K <= N

5.

Given two unsorted arrays A of size N and B of size M of distinct elements, the task is to find all pairs from both arrays whose sum is equal to X.

Note: All pairs should be printed in increasing order of u. For eg. for two pairs (u1,v1) and (u2,v2), if u1 < u2 then

(u1,v1) should be printed first else second.

Example 1:

Input:

A[] = {1, 2, 4, 5, 7}

B[] = {5, 6, 3, 4, 8}

X = 9

Output:

1 8

4 5

5 4

Explanation:

(1, 8), (4, 5), (5, 4) are the

pairs which sum to 9.

Example 2:

Input:

A[] = {-1, -2, 4, -6, 5, 7}

B[] = {6, 3, 4, 0}

X = 8

Output:

4 4

5 3

Your Task:

You don't need to read input or print anything. Your task is to complete the function allPairs() which takes the array A[], B[], its size N and M respectively, and an integer X as inputs and returns the sorted vector pair values of all the pairs u,v where u belongs to array A and v belongs to array B. If no such pair exists return empty vector pair.

Expected Time Complexity: O(NLog(N))

Expected Auxiliary Space: O(N)

Constraints:

1 ≤ N, M ≤ 106

-106 ≤ X, A[i], B[i] ≤ 106

6.

Given two arrays A and B of equal size N, the task is to find if given arrays are equal or not. Two arrays are said to be equal if both of them contain same set of elements, arrangements (or permutation) of elements may be different though.

Note : If there are repetitions, then counts of repeated elements must also be same for two array to be equal.

Example 1:

Input:

N = 5

A[] = {1,2,5,4,0}

B[] = {2,4,5,0,1}

Output: 1

Explanation: Both the array can be

rearranged to {0,1,2,4,5}

Example 2:

Input:

N = 3

A[] = {1,2,5}

B[] = {2,4,15}

Output: 0

Explanation: A[] and B[] have only

one common value.

Your Task:

Complete check() function which takes both the given array and their size as function arguments and returns true if the arrays are equal else returns false.The 0 and 1 printing is done by the driver code.

Expected Time Complexity : O(N)

Expected Auxilliary Space : O(N)

Constraints:

1<=N<=107

1<=A[],B[]<=1018