### K-th Largest Sum Contiguous Subarray

You are given an array **Arr** of size **N**. You have to find the **K**-th **largest** sum of contiguous **subarray** within the array elements.

**Example 1:**

**Input:**

N = 3

K = 2

Arr = {3,2,1}

**Output:**

5

**Explanation:**

The different subarray sums we can get from the array

are = {6,5,3,2,1}. Where 5 is the 2nd largest.

**Example 2:**

**Input:**

N = 4

K = 3

Arr = {2,6,4,1}

**Output:**

11

**Explanation:**

The different subarray sums we can get from the array

are = {13,12,11,10,8,6,5,4,2,1}. Where 11 is the 3rd largest.

//{ Driver Code Starts

import java.io.\*;

import java.util.\*;

class IntArray

{

public static int[] input(BufferedReader br, int n) throws IOException

{

String[] s = br.readLine().trim().split(" ");

int[] a = new int[n];

for(int i = 0; i < n; i++)

a[i] = Integer.parseInt(s[i]);

return a;

}

public static void print(int[] a)

{

for(int e : a)

System.out.print(e + " ");

System.out.println();

}

public static void print(ArrayList<Integer> a)

{

for(int e : a)

System.out.print(e + " ");

System.out.println();

}

}

class CodingMaxima {

public static void main(String[] args) throws IOException {

BufferedReader br = new BufferedReader(new InputStreamReader(System.in));

int t;

t = Integer.parseInt(br.readLine());

while(t-- > 0){

int N;

N = Integer.parseInt(br.readLine());

int K;

K = Integer.parseInt(br.readLine());

int[] Arr = IntArray.input(br, N);

Solution obj = new Solution();

int res = obj.kthLargest(N, K, Arr);

System.out.println(res);

}

}

}

// } Driver Code Ends

class Solution {

public static int kthLargest(int N, int K, int[] arr) {

ArrayList<Integer> ar = new ArrayList<>();

int sum=0;

for(int i=0;i<N;i++){

sum=0;

for(int j=i;j<N;j++){

sum+=arr[j];

ar.add(sum);

}

}

Collections.sort(ar,Collections.reverseOrder());

return ar.get(K-1);

}

}