Luis Gerardo Bravo Ramones

A01282014

Oportunidad 2

**M2C**

class Solution {

public:

int Contador(vector<int>& nums, int mayor, int iInicio, int iFinal) {

int cont = 0;

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

if (nums[i] == mayor)

cont++;

}

return cont;

}

int Divide(vector<int>& nums, int iInicio, int iFinal) {

if (iInicio == iFinal) return nums[iInicio];

int mitad = (iFinal-iInicio)/2 + iInicio;

int izq = Divide(nums, iInicio, mitad);

int der = Divide(nums, mitad+1, iFinal);

if (izq == der) return izq;

int izqCantidad = Contador(nums, izq, iInicio, iFinal);

int derCantidad = Contador(nums, der, iInicio, iFinal);

return izqCantidad > derCantidad ? izq : der;

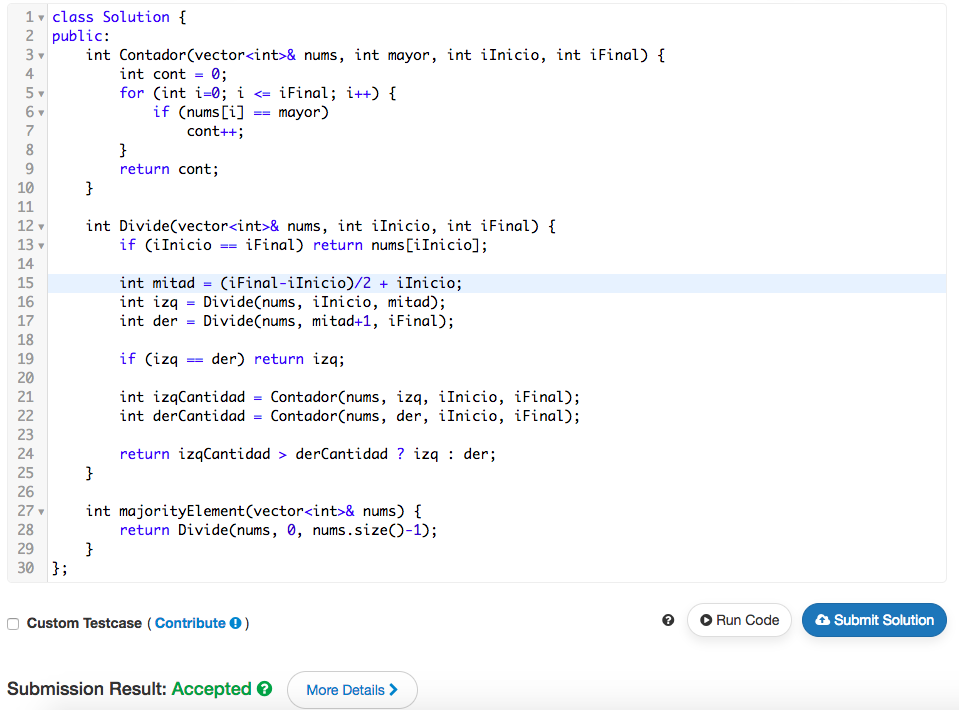
}

int majorityElement(vector<int>& nums) {

return Divide(nums, 0, nums.size()-1);

}

};



**M2E**

#include <bits/stdc++.h>

using namespace std;

string angryProfessor(int k, vector <int> a, int count) {

// Complete this function

return count < k ? "YES" : "NO";

}

int main() {

int t;

cin >> t;

for(int a0 = 0; a0 < t; a0++){

int n;

int k;

int count = 0;

cin >> n >> k;

vector<int> a(n);

for(int a\_i = 0; a\_i < n; a\_i++){

cin >> a[a\_i];

if (a[a\_i] <= 0)

count++;

}

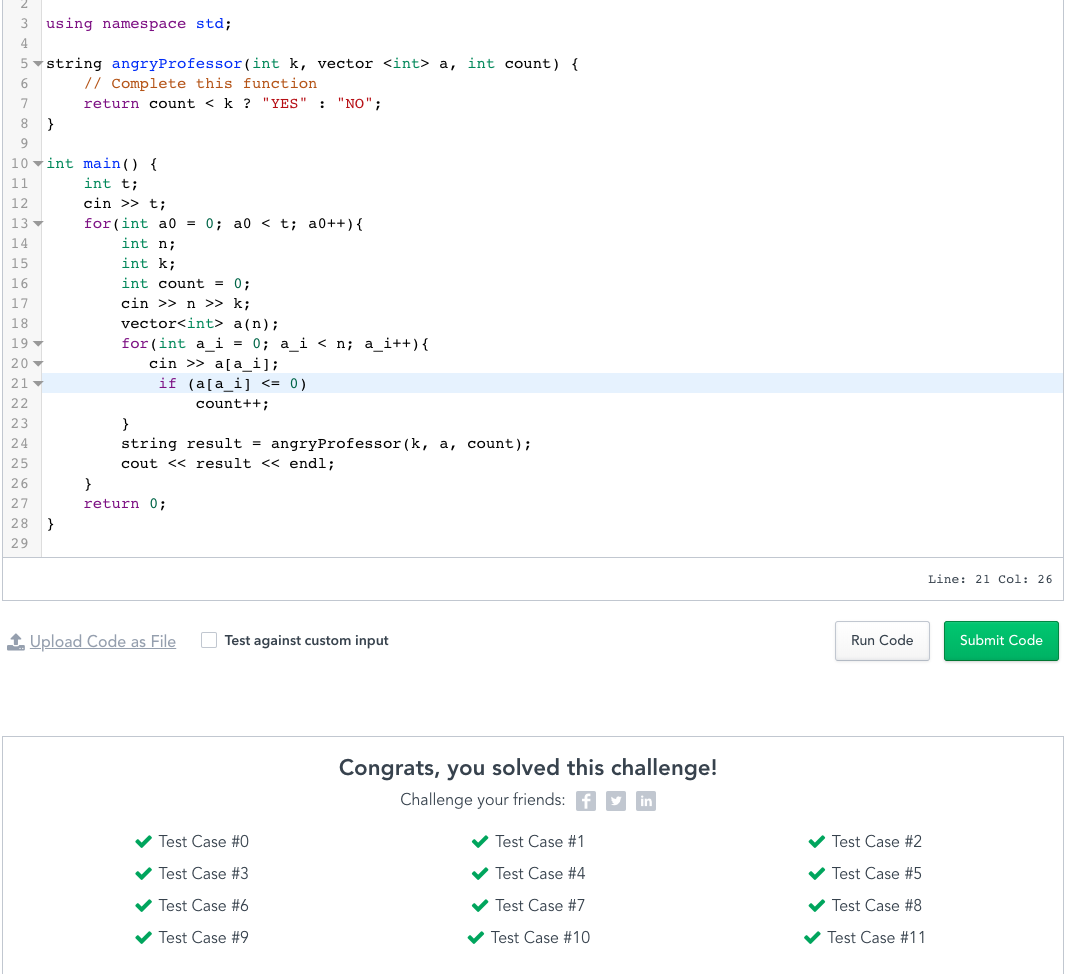
string result = angryProfessor(k, a, count);

cout << result << endl;

}

return 0;

}

****

**M2G**

class Solution {

public:

int mitad;

int pivote = 0, aux, j;

int elemPivote;

void Particion(int inicio, int fin, int &pivote, vector<int>& nums) {

elemPivote = nums[inicio];

j = inicio;

for(int i=inicio+1; i <= fin; i++) {

if (nums[i] < elemPivote) {

j++;

//Intercambia arreglo[i] con arreglo[j]

aux = nums[i];

nums[i] = nums[j];

nums[j] = aux;

}

}

pivote = j;

//Intercambia arreglo[inicio] con arreglo[pivote]

aux = nums[inicio];

nums[inicio] = nums[pivote];

nums[pivote] = aux;

}

void QuickSort(int inicio, int fin, vector<int>& nums) {

if (inicio < fin) {

Particion(inicio, fin, pivote, nums);

QuickSort(inicio, pivote-1, nums);

QuickSort(pivote+1, fin, nums);

}

}

int findKthLargest(vector<int>& nums, int k) {

QuickSort(0, nums.size()-1, nums);

for(int i=0; i < nums.size(); i++)

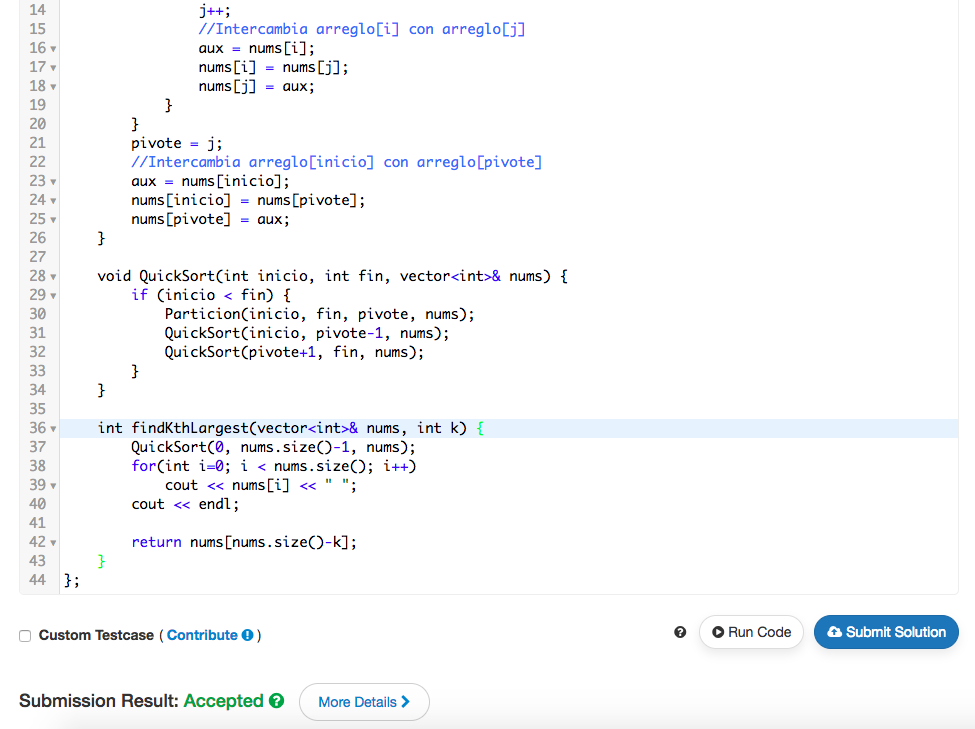
cout << nums[i] << " ";

cout << endl;

return nums[nums.size()-k];

}

};



**M2J**

class Solution {

public:

int minimo(int a, int b) {

return a < b ? a : b;

}

int minimumTotal(vector<vector<int>>& triangle) {

vector<int> camino(triangle.back());

for(int layer = triangle.size()-2; layer >= 0; layer--) {

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

camino[i] = minimo(camino[i], camino[i+1]) + triangle[layer][i];

}

}

return camino[0];

}

};