Reward: https://codefights.com/img/coins_new.png **1000**

Find the average, mode and median values of an array of integers.

* In case that there are two or more modes , return mode=0
* If the array.size() is even, the median will be a semisum of numbers.

[Show Less](javascript:void(0);)

**Input (A)** → array.integer :

Array with the data

**Output** → array.integer :

Array: [average,mode,median]

<https://codefights.com/feed/68dtz4DJy4KkejCcB>

------------ACEPTADO-----------

#include <iostream>

#include <stdio.h>

#include <vector>

#include <map>

#include <algorithm>

std::vector<int> StatisticalExcercise(std::vector<int > A) {

*/\* uso map para contar las veces que sale un elemento*

*<elemento, veces que esta elemento en el array> \*/*

     std::map<int, int> cont;

     int sum = 0;

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

        int key = A[i];

        sum += A[i];

        if(cont.count(key)) {

                cont[key]++;

        } else {

                cont[key] = 1;

        }

     }

*/\* busco el maximo de veces que sale un key (elemento) \*/*

     int max = 0;

     for(std::map<int, int>::iterator it = cont.begin(); it != cont.end(); it++) {

                max = std::max(max, it->second);

     }

     int moda = 0;

     int contMax = 0;

     int keyMax = 0;

*/\* si el elemento que esta mas veces es el unico que esta esa*

*cantidad de veces entonces hay moda, de lo contrario la moda es cero*

*it->first = clave (elemento)*

*it->second = valor (veces que esta el elemento) \*/*

     for(std::map<int, int>::iterator it = cont.begin(); it != cont.end(); it++) {

        if(it->second == max) {

                 contMax++;

                         keyMax = it->first;

        }

     }

     if(contMax == 1) {

        moda = keyMax;

     }

     int mediana = 0;

*/\* ordeno el array para obtener la mediana \*/*

         std::sort(A.begin(), A.end());

     if(A.size() % 2 == 0) {

        mediana = ( A[A.size()/2] + A[A.size()/2-1]  ) /2;

     } else {

        mediana = A[A.size()/2];

     }

     int prom = sum / A.size();

     std::vector<int> R;

     R.push\_back(prom);

     R.push\_back(moda);

     R.push\_back(mediana);

     return R;

}

int main() {

        std::vector<int> V;

        int arr[] = {12, 5, 3, 5 , 2};

        int n = sizeof(arr)/sizeof(int);

        for(int i = 0; i < n; i++) {

                V.push\_back(arr[i]);

        }

        std::vector<int> res = StatisticalExcercise(V);

        printf("Prom: %d ", res[0]);

        printf("**\n**Moda: %d ", res[1]);

        printf("**\n**Mediana: %d", res[2]);

        return 0;

}

[lyenliang](https://codefights.com/profile/lyenliang) 's solution:

int[] StatisticalExercise(int[] A) {

Arrays.sort(A);

int result[] = new int[3];

int sum = 0;

int len = A.length;

HashMap<Integer, Integer> seen = new HashMap<Integer, Integer>(); // value -> number of occurences

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

sum += A[i];

if( seen.containsKey(A[i]) ) {

int numOccurences = seen.get(A[i]) + 1;

seen.put(A[i], numOccurences);

} else {

seen.put(A[i], 1);

}

}

int count = 0;

result[1] = 0;

boolean multiModes = false;

for(Map.Entry<Integer, Integer> entry : seen.entrySet()) {

if(entry.getValue() > count) {

multiModes = false;

result[1] = entry.getKey();

count = entry.getValue();

} else if(entry.getValue() == count) {

multiModes = true;

}

}

if(multiModes) {

result[1] = 0;

}

result[0] = sum/len; // average

if(len % 2 == 0) {

result[2] = (A[len/2] + A[len/2 - 1]) / 2; // semisum

} else {

result[2] = A[(len-1)/2];

}

return result;

}

[Christoph\_34](https://codefights.com/profile/Christoph_34) 's solution:

int[] StatisticalExercise(int[] A) {

int a = 0;

int c = 0;

int m = 0;

int n = 0;

int z = 0;

int l = A.length;

Arrays.sort(A);

for(int x: A)

a += x;

a /= l;

if(l % 2 == 0)

c = (A[l/2] + A[l/2-1]) / 2;

else

c = A[l/2];

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

int x = 1;

for(int j = i + 1; j < l; j++) {

if(A[i] == A[j])

x++;

else

break;

}

if(x > m) {

m = x;

z = i;

}

else if(x > n)

n = x;

}

if(m == n)

m = 0;

else

m = A[z];

return new int[] {a, m, c};

}