Name: Jose Carlos Moreno Soria

**Problem 1**

#include <iostream>

using namespace std;

int main()

{

int nums[] = {3,2,3,3,2,3};

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

n = n/2;

int a = -1;

int b = -1;

int ac = 0;

int bc = 0;

for(int i : nums){

if(a==-1){

a=i;

ac++;

continue;

}

if(b==-1 && a!=i){

b=i;

bc++;

continue;

}

if(a==i){

ac++;

}

if(b==i){

bc++;

}

}

int maj;

if(ac>n){

maj = a;

}

else if(bc>n){

maj = b;

}

else{

maj=-1;

}

cout<<maj<<endl;

return 0;

}

The time complexity of the algorithm is f(n)=O(n) given that the most significant operation is the for loop and it only executes once for the length of the input array.

**Problem 2**

**Problem 3**

**Problem 4**

**Problem 5**

#include <iostream>

#include <vector>

using namespace std;

int shortest(int nums[], int arsz, int k){

int len=0;

int sum=0;

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

sum+=nums[i];

if(sum<k){

len++;

}

else{

int c=1;

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

if(sum-nums[i-j]>k){

sum=sum-nums[i-j];

len--;

}

else{break;}

}

}

}

if(sum<k){return -1;}

else{return len;}

}

int main()

{

int nums[] = {3,2,1,5,6,4};

int arsz = sizeof(nums)/sizeof(int);

int k = 12;

cout<< shortest(nums, arsz, k) << endl;

return 0;

}

The time complexity is f(n)=O(n) because in the worst case scenario of every element being equal and the sum being equal to 2 elements, the algorithm would add and remove every element once, executing total operations of twice the length of the array, which is linear time.