**PROBLEMA 1**

#include <iostream>

using namespace std;

#define DIM 100

//prototipo funzione di conteggio delle occorrenze

int conteggio\_occorrenze(int \*, int, int, int);

int main()

{

int casi\_test;

int dim, value;

int A[DIM];

cout << "Inserire il numero di casi di test: ";

cin >> casi\_test;

int iter = 0;

while(iter < casi\_test){

cout << "Inserire l'elemento da ricercare: ";

cin >> value;

cout << "Inserire la dimensione del vettore: ";

cin >> dim;

int counter = 0;

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

cout << "Elemento " << i << " : ";

cin >> A[i];

}

//DEBUG

cout << "n.ro casi test: " << casi\_test << " elem: " << value << " dim: " << dim << endl;

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

cout << A[i] << " ";

}

cout << "\n";

//array

counter = conteggio\_occorrenze(A,value,0,dim-1);

cout << "Occorrenze elemento " << value << " : " << counter << endl;

iter++;

}

return 0;

}

int conteggio\_occorrenze(int \* A, int e, int index\_first, int index\_last)

{

int counter;

int index\_mid;

if((index\_first > index\_last) or (index\_first == index\_last and A[index\_first] != e)) return 0;

if(index\_first == index\_last and A[index\_first] == e) return 1;

index\_mid = index\_first + index\_last / 2;

counter = count(A, e, index\_first,index\_mid) + count(A, e, 1 + index\_mid, index\_last);

return counter;

}

//COMPLESSITA’ 🡪 O(nlogn)

**PROBLEMA 2**

#include <iostream>

#include <cmath>

#include <vector>

using namespace std;

vector<int> set;

vector<int> prime;

bool isPrime(int number)

{

int rootNumber = sqrt(number);

bool condition = true;

if(number == 1) return false;

for (int i = 2; i <= rootNumber; i++){

if(number % i == 0) return false;

}

return true;

}

void view()

{

int dim = set.size();

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

cout << set[i] << " ";

}

cout << "\n";

}

void sum(int totalSum, int N, int S, int index)

{

if(totalSum == S and set.size() == N)

{

view();

return;

}

if(totalSum > S or index == prime.size()) return;

set.push\_back(prime[index]);

sum(totalSum + prime[index], N, S, index+1);

set.pop\_back();

sum(totalSum, N, S, index+1);

}

void generatePrime(int S, int N, int P)

{

for(int i = P+1; i <= S ; i++)

{

if(isPrime(i)) prime.push\_back(i);

}

if(prime.size() < N) return;

sum(0, N, S, 0);

}

int main()

{

int S, N, P;

int casi\_test, iter;

cout << "Inserire il n.ro dei casi di test: ";

cin >> casi\_test;

while(iter < casi\_test){

cout << "CASO DI TEST " << iter+1 << endl;

generatePrime(S,N,P);

iter++;

}

return 0;

}