**Problem No: 01**

**Topic: Algorithms**

**Problem Title:**

Given a list of n integers, find the largest integer in the list and its complexity.

**Objectives:**

To find the largest integer and its complexity.

**Source Code:**

#include <iostream>

#include <cstdlib>

#include <ctime>

using namespace std;

void inputArray(int \*a, int n);

void outputArray(int \*a, int n);

int maxInt (int \*a, int n);

int main()

{

int n;

cout << "Enter n: ";

cin >> n;

int a[n], i;

inputArray(a, n);

outputArray(a, n);

cout << "Largest Integer: " << maxInt(a, n) << endl;

return 0;

}

void inputArray(int \*a, int n)

{

srand((unsigned) time(0));

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

a[i] = rand() % (n + 1);

}

void outputArray(int \*a, int n)

{

cout << "Array:\n";

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

cout << a[i] << " ";

cout << endl;

}

int maxInt (int \*a, int n)

{

int max = 0;

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

if(a[i] > max)

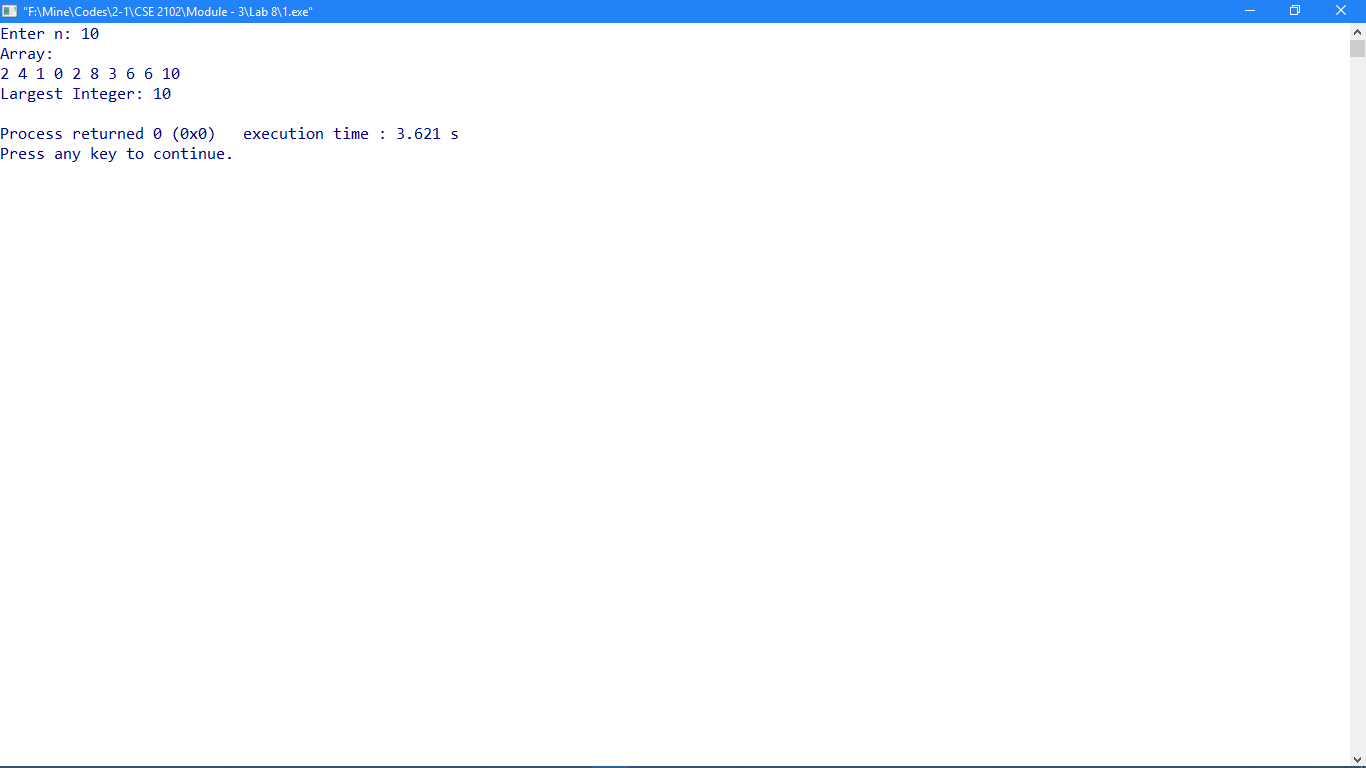
max = a[i];

}

return max;

}

**Output:**



**Problem No: 02**

**Topic: Algorithms**

**Problem Title:**

Given a list of n integers, find the first and last occurrences of the largest integer in the list.

**Objectives:**

To find the first and last occurrences of the largest integer.

**Source Code:**

#include <iostream>

#include <cstdlib>

#include <ctime>

using namespace std;

int first, last, m;

void inputArray(int \*a, int n);

void outputArray(int \*a, int n);

void findInt (int \*a, int n);

int main()

{

int n;

cout << "Enter n: ";

cin >> n;

int a[n], i;

inputArray(a, n);

outputArray(a, n);

findInt(a, n);

cout << "Largest Integer: " << m << endl;

cout << "First Occurrence: " << first

<< "\nLast Occurrence: " << last << endl;

return 0;

}

void inputArray(int \*a, int n)

{

srand((unsigned) time(0));

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

a[i] = rand() % (n + 1);

}

void outputArray(int \*a, int n)

{

cout << "Array:\n";

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

cout << a[i] << " ";

cout << endl;

}

void findInt (int \*a, int n)

{

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

if(a[i] > m){

m = a[i];

first = i;

}

}

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

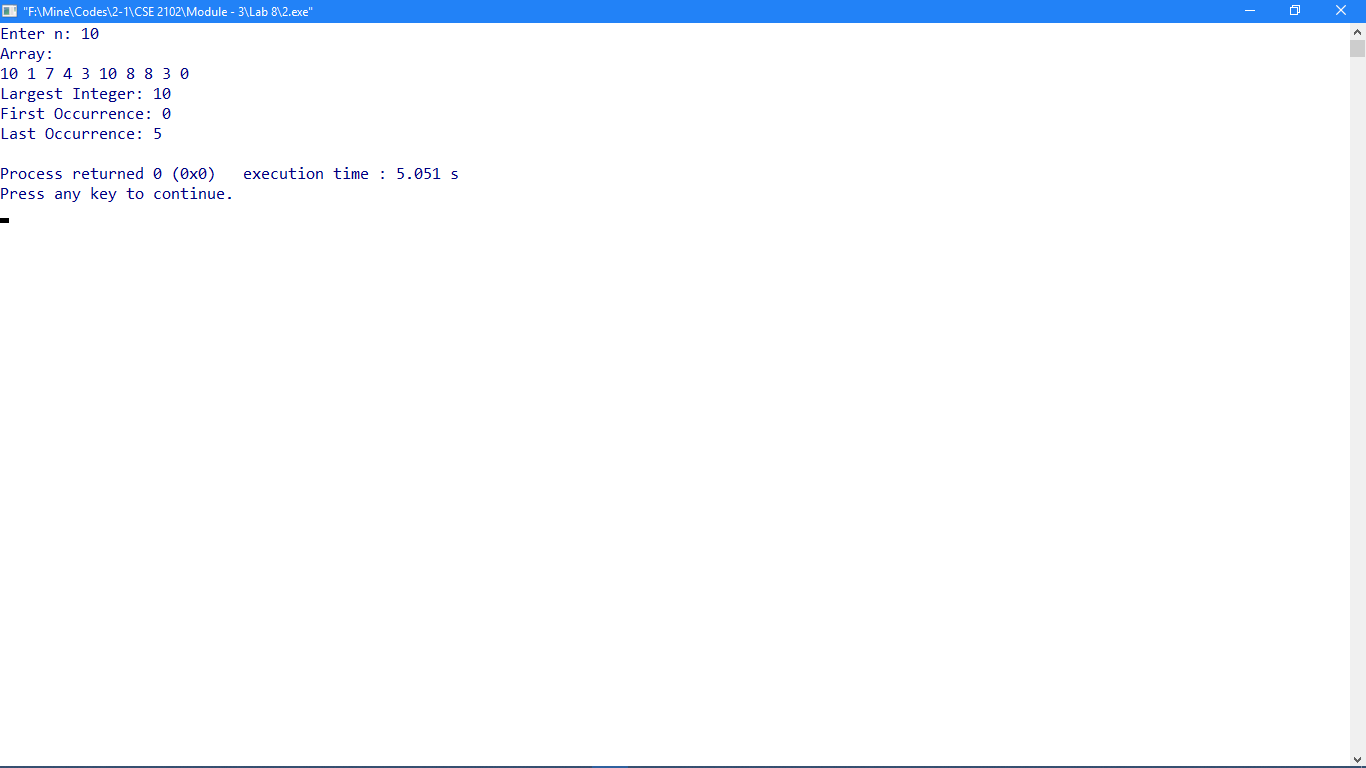
if(a[i] == m)

last = i;

}

}

**Output:**



**Problem No: 03**

**Topic: Algorithms**

**Problem Title:**

Given a list of n distinct integers, determine the position of an integer in the list using a linear search.

**Objectives:**

To learn linear searching.

**Source Code:**

#include <iostream>

#include <cstdlib>

#include <ctime>

using namespace std;

void inputArray(int \*a, int n);

void outputArray(int \*a, int n);

int linearSearch(int \*a, int n, int x);

int main()

{

int n, x;

cout << "Enter n: ";

cin >> n;

int a[n], i;

inputArray(a, n);

outputArray(a, n);

cout << "Enter integer: ";

cin >> x;

cout << "position of " << x << " is " << linearSearch(a, n, x) << endl;

return 0;

}

void inputArray(int \*a, int n)

{

srand((unsigned) time(0));

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

a[i] = rand() % (n + 1);

}

void outputArray(int \*a, int n)

{

cout << "Array:\n";

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

cout << a[i] << " ";

cout << endl;

}

int linearSearch(int \*a, int n, int x)

{

int pos, flag = 0;

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

if(a[i] == x){

pos = i;

flag = 1;

break;

}

if(flag == 0)

cout << "Not Found\n";

else

return pos;

}

**Output:**



**Problem No: 04**

**Topic: Algorithms**

**Problem Title:**

Given a list of n distinct integers, determine the position of an integer in the list using a binary search.

**Objectives:**

To learn binary searching.

**Source Code:**

#include <iostream>

#include <cstdlib>

#include <ctime>

using namespace std;

void inputArray(int \*a, int n);

void outputArray(int \*a, int n);

void bubbleSort(int \*a, int n);

int binarySearch(int \*a, int n, int x);

int main()

{

int n, x;

cout << "Enter n: ";

cin >> n;

int a[n], i;

inputArray(a, n);

outputArray(a, n);

bubbleSort(a, n);

cout << "Sorted ";

outputArray(a, n);

cout << "Enter integer: ";

cin >> x;

cout << "position of " << x << " is " << binarySearch(a, n, x) << endl;

return 0;

}

void inputArray(int \*a, int n)

{

srand((unsigned) time(0));

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

a[i] = rand() % (n \* n);

}

void outputArray(int \*a, int n)

{

cout << "Array:\n";

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

cout << a[i] << " ";

cout << endl;

}

void bubbleSort(int \*a, int n)

{

int i, j, temp;

for(i = 0; i < n - 1; i++){

for(j = 0; j < n - 1 - i; j++){

if(a[j] > a[j + 1]){

temp = a[j];

a[j] = a[j + 1];

a[j + 1] = temp;

}

}

}

}

int binarySearch(int \*a, int n, int x)

{

int l, r, mid, flag = 0;

l = 0;

r = n;

while(l <= r){

mid = (l + r) / 2;

if(x < a[mid])

r = mid;

else if(x > a[mid])

l = mid;

else if(x == a[mid])

return mid;

}

}

**Output:**



**Problem No: 05**

**Topic: Algorithms**

**Problem Title:**

Given a list of n integers, sort them using a bubble sort

**Objectives:**

To learn bubble sorting.

**Source Code:**

#include <iostream>

#include <cstdlib>

#include <ctime>

using namespace std;

void inputArray(int \*a, int n);

void outputArray(int \*a, int n);

void bubbleSort(int \*a, int n);

int main()

{

int n, x;

cout << "Enter n: ";

cin >> n;

int a[n], i;

inputArray(a, n);

outputArray(a, n);

bubbleSort(a, n);

cout << "Sorted ";

outputArray(a, n);

return 0;

}

void inputArray(int \*a, int n)

{

srand((unsigned) time(0));

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

a[i] = rand() % (n \* n);

}

void outputArray(int \*a, int n)

{

cout << "Array:\n";

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

cout << a[i] << " ";

cout << endl;

}

void bubbleSort(int \*a, int n)

{

int i, j, temp;

for(i = 0; i < n - 1; i++){

for(j = 0; j < n - 1 - i; j++){

if(a[j] > a[j + 1]){

temp = a[j];

a[j] = a[j + 1];

a[j + 1] = temp;

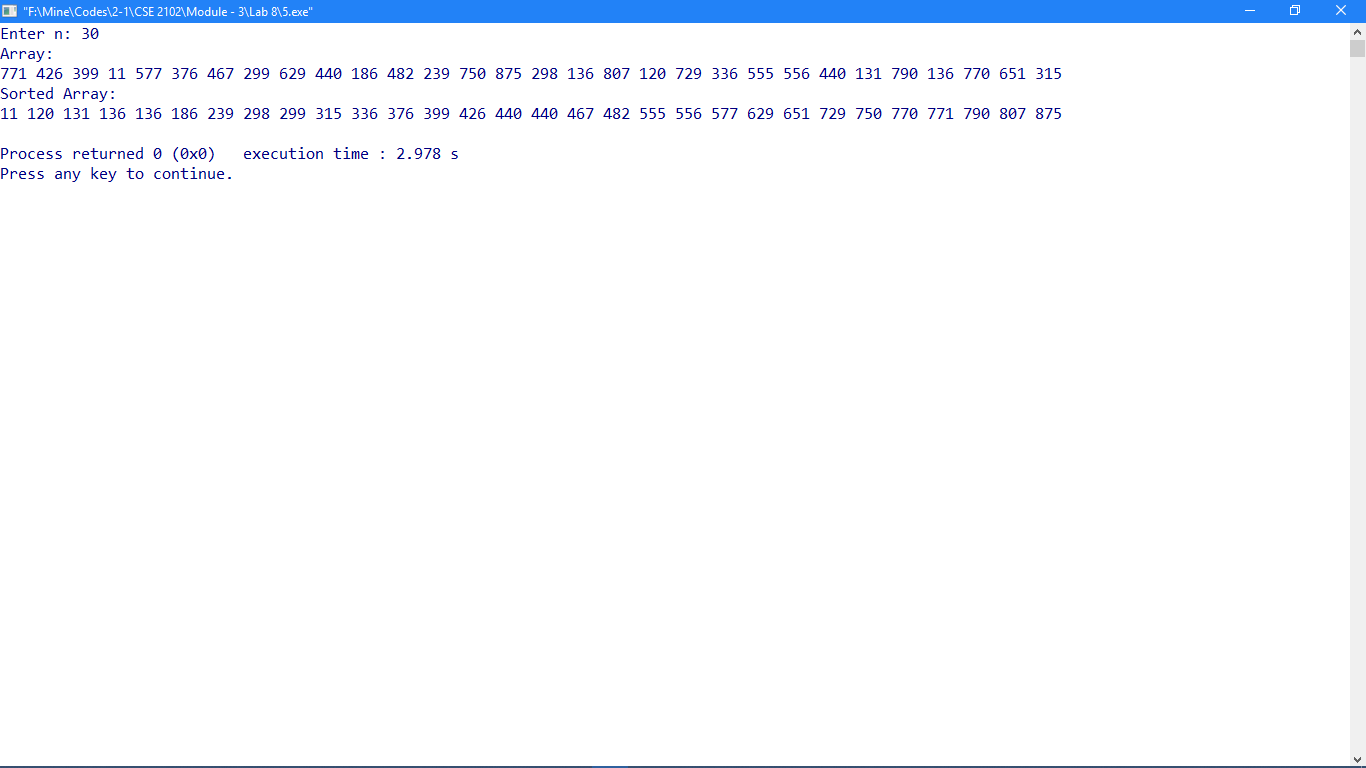
}

}

}

}

**Output:**



**Problem No: 06**

**Topic: Algorithms**

**Problem Title:**

Given a list of n integers, sort them using a insertion sort.

**Objectives:**

To learn insertion sorting.

**Source Code:**

#include <iostream>

#include <cstdlib>

#include <ctime>

using namespace std;

void inputArray(int \*a, int n);

void outputArray(int \*a, int n);

void insertionSort(int \*a, int n);

int main()

{

int n, x;

cout << "Enter n: ";

cin >> n;

int a[n], i;

inputArray(a, n);

outputArray(a, n);

insertionSort(a, n);

cout << "Sorted ";

outputArray(a, n);

return 0;

}

void inputArray(int \*a, int n)

{

srand((unsigned) time(0));

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

a[i] = rand() % (n \* n);

}

void outputArray(int \*a, int n)

{

cout << "Array:\n";

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

cout << a[i] << " ";

cout << endl;

}

void insertionSort(int \*a, int n)

{

int i, j, key;

for(i = 1; i < n; i++){

key = a[i];

j = i - 1;

while((j >= 0) && (a[j] > key)){

a[j + 1] = a[j];

j--;

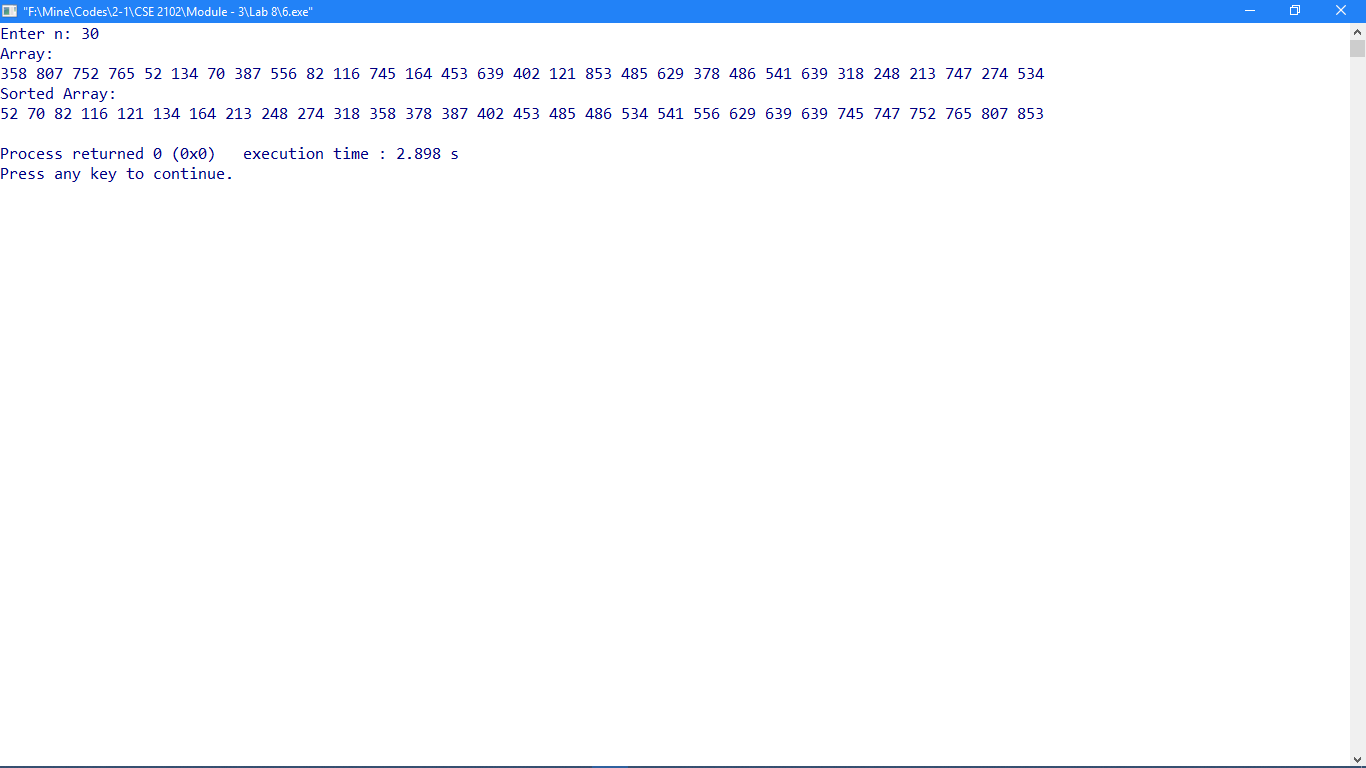
}

a[j + 1] = key;

}

}

**Output:**



**Problem No: 07**

**Topic: Algorithms**

**Problem Title:**

Given an ordered list of n distinct integers and an integer x, find the number of comparisons used to determine the position of an integer in the list using a binary search and using a linear search.

**Objectives:**

To compare linear searching and binary searching.

**Source Code:**

#include <iostream>

#include <cstdlib>

#include <ctime>

using namespace std;

void inputArray(int \*a, int \*b, int n);

void outputArray(int \*a, int n);

void bubbleSort(int \*a, int n);

int linearSearch(int \*a, int n, int x);

int binarySearch(int \*a, int n, int x);

int bi\_com;

int li\_com;

int main()

{

int n, x;

cout << "Enter n: ";

cin >> n;

int a[n], b[n], i;

inputArray(a, b, n);

outputArray(a, n);

bubbleSort(a, n);

cout << "Sorted ";

outputArray(a, n);

//outputArray(b, n);

cout << "Enter integer: ";

cin >> x;

cout << "position of " << x << " is " << binarySearch(a, n, x) << " (binary search)";

cout << " comparisons: " << bi\_com << endl;

cout << "position of " << x << " is " << linearSearch(b, n, x) << " (linear search)";

cout << " comparisons: " << li\_com << endl;

return 0;

}

void inputArray(int \*a, int \*b, int n)

{

srand((unsigned) time(0));

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

a[i] = rand() % (n \* n);

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

b[i] = a[i];

}

void outputArray(int \*a, int n)

{

cout << "Array:\n";

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

cout << a[i] << " ";

cout << endl;

}

void bubbleSort(int \*a, int n)

{

int i, j, temp;

for(i = 0; i < n - 1; i++){

for(j = 0; j < n - 1 - i; j++){

if(a[j] > a[j + 1]){

temp = a[j];

a[j] = a[j + 1];

a[j + 1] = temp;

}

}

}

}

int binarySearch(int \*a, int n, int x)

{

int l, r, mid, flag = 0;

bi\_com = 0;

l = 0;

r = n;

while(l <= r){

bi\_com++;

mid = (l + r) / 2;

if(x < a[mid])

r = mid;

else if(x > a[mid])

l = mid;

else if(x == a[mid])

return mid;

}

}

int linearSearch(int \*a, int n, int x)

{

int pos, flag = 0;

li\_com = 0;

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

li\_com++;

if(a[i] == x){

pos = i;

flag = 1;

break;

}

}

if(flag == 0)

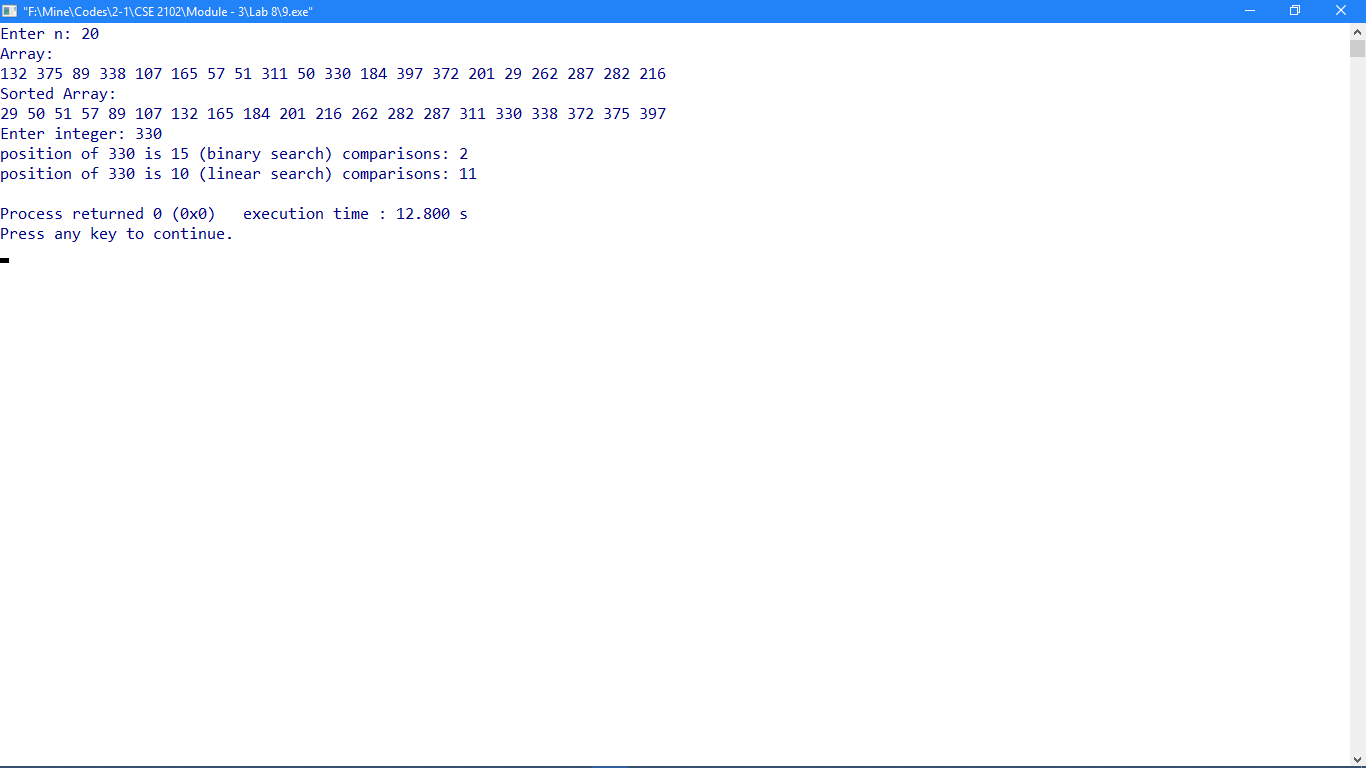
cout << "Not Found\n";

else

return pos;

}

**Output:**



**Problem No: 08**

**Topic: Algorithms**

**Problem Title:**

Given a list of n integers, determine the number of comparisons used by the bubble sort and by the insertion sort to sort this list.

**Objectives:**

To compare bubble sorting and insertion sorting.

**Source Code:**

#include <iostream>

#include <cstdlib>

#include <ctime>

using namespace std;

void inputArray(int \*a, int \*b, int n);

void outputArray(int \*a, int n);

void bubbleSort(int \*a, int n);

void insertionSort(int \*a, int n);

int bub\_com;

int ins\_com;

int main()

{

int n, x;

cout << "Enter n: ";

cin >> n;

int a[n], b[n], i;

inputArray(a, b, n);

outputArray(a, n);

bubbleSort(a, n);

cout << "Bubble sorted ";

outputArray(a, n);

cout << "no. of swaps (bubble sort): " << bub\_com << endl;

outputArray(b, n);

insertionSort(b, n);

cout << "Insertion sorted ";

outputArray(b, n);

cout << "no. of swaps (insertion sort): " << ins\_com << endl;

return 0;

}

void inputArray(int \*a, int \*b, int n)

{

srand((unsigned) time(0));

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

a[i] = rand() % (n + 100);

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

b[i] = a[i];

}

void outputArray(int \*a, int n)

{

cout << "Array:\n";

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

cout << a[i] << " ";

cout << endl;

}

void bubbleSort(int \*a, int n)

{

int i, j, temp;

bub\_com = 0;

for(i = 0; i < n - 1; i++){

for(j = 0; j < n - 1 - i; j++){

if(a[j] > a[j + 1]){

bub\_com++;

temp = a[j];

a[j] = a[j + 1];

a[j + 1] = temp;

}

}

}

}

void insertionSort(int \*a, int n)

{

int i, j, key;

ins\_com = 0;

for(i = 1; i < n; i++){

key = a[i];

j = i - 1;

while((j >= 0) && (a[j] > key)){

a[j + 1] = a[j];

ins\_com++;

j = j -1;

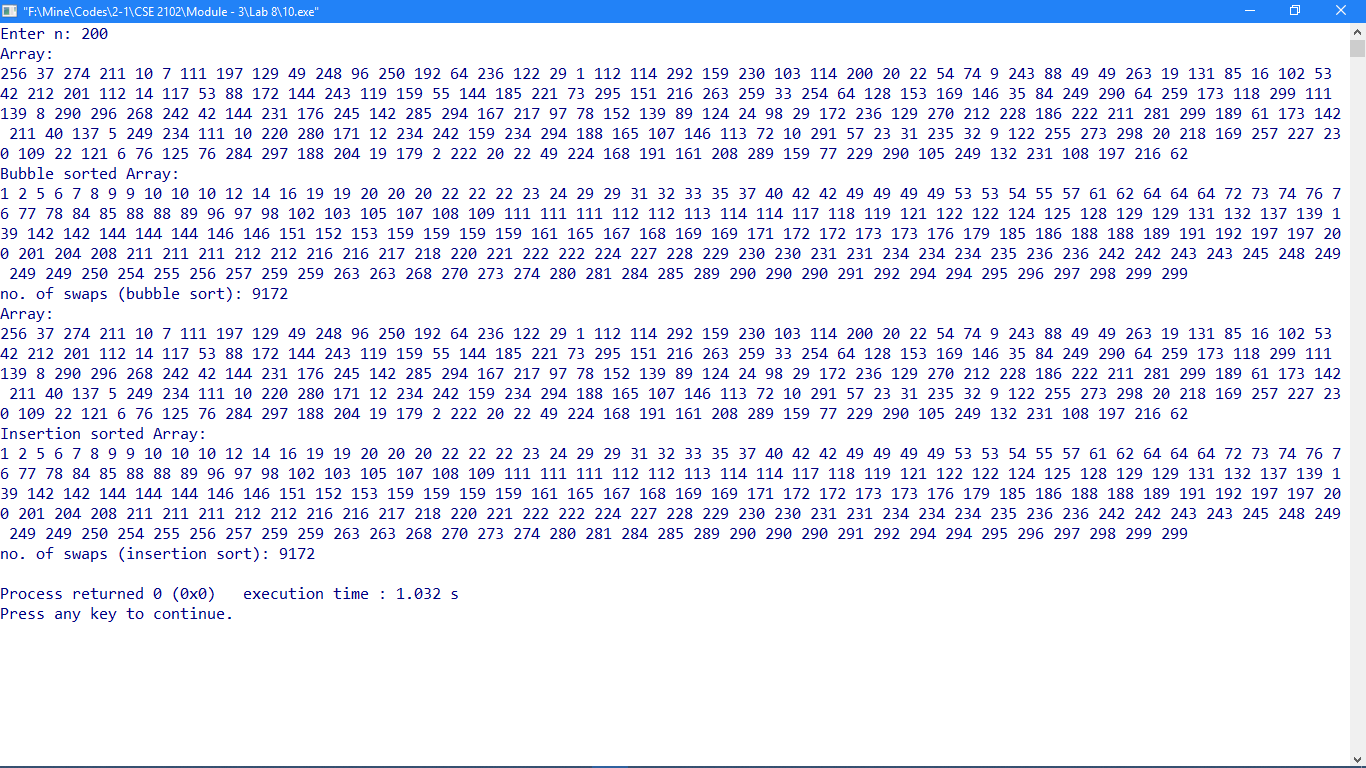
}

a[j + 1] = key;

}

}

**Output:**



**Problem No: 09**

**Topic: Algorithms**

**Problem Title:**

Given a set of identification numbers, use a hash function to assign them to memory locations where there are k memory locations.

**Objectives:**

To implement hash table using a hash function.

**Source Code:**

#include <iostream>

using namespace std;

void input(int a[], int n);

void display(int a[], int n);

int count = 0;

int main()

{

int n, no;

cout << "Enter the size of array: ";

cin >> n;

int a[n] = {0};

while(1){

cout << "1. insert\n2. display\n3. exit\n\n";

cin >> no;

cout << endl << endl;

if(no == 1)

input(a, n);

if(no == 2)

display(a, n);

if(no == 3)

break;

}

return 0;

}

void input(int a[], int n)

{

int i, k, j;

if(count == n){

cout << "Array Full\n";

return;

}

cout << "Enter an element: ";

cin >> k;

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

j = (k + i) % n;

if(a[j] == 0){

a[j] = k;

count ++;

break;}

}

}

void display(int a[], int n)

{

int i;

if(count == 0){

cout << "Array empty\n";

return;}

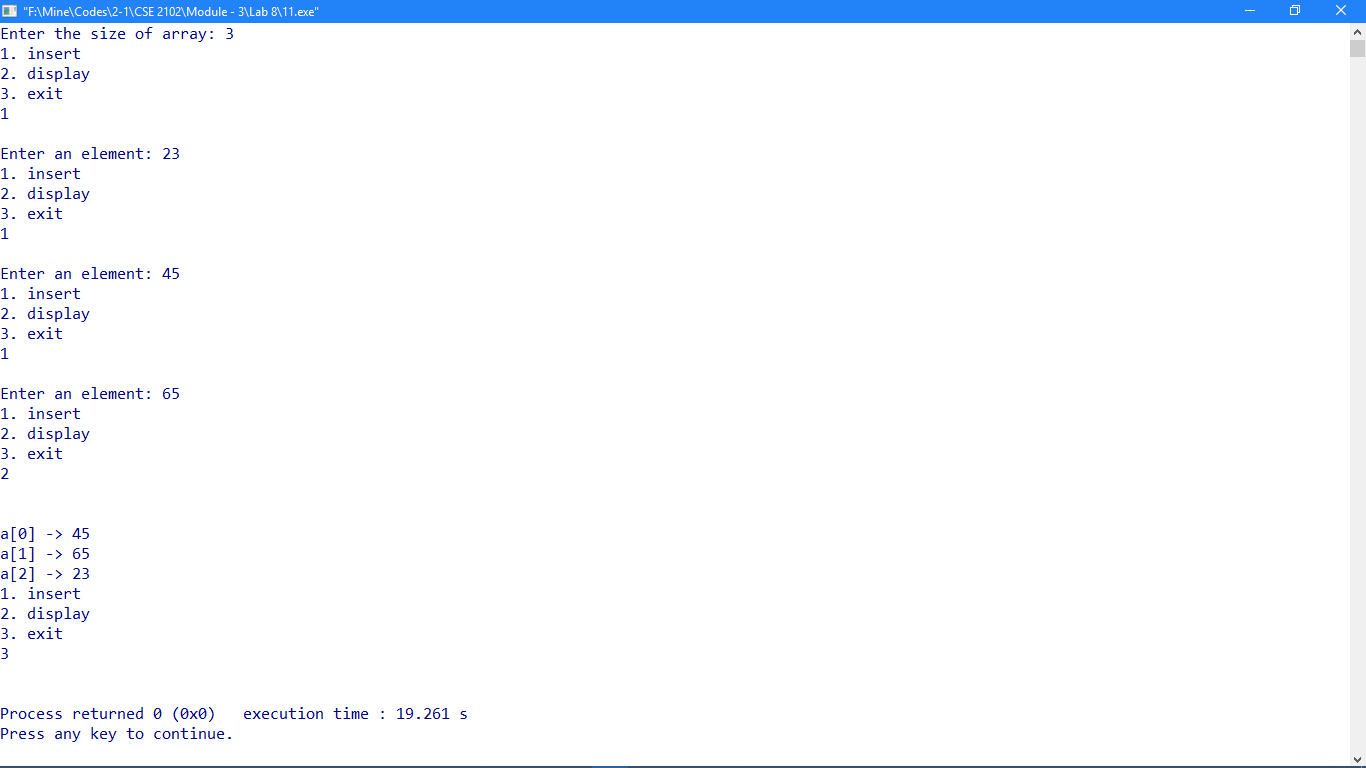
for(i = 0; i < n; i++)

cout <<"\na[" << i << "] -> " << a[i] << " ";

cout << endl;

}

**Output:**



**Problem No: 10**

**Topic: Number Theory**

**Problem Title:**

Given a positive integer, determine whether it is prime.

**Objectives:**

To determine prime numbers.

**Source Code:**

#include <iostream>

#include <cmath>

using namespace std;

void checkPrime(int n);

int prime[] = {2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101};

int main()

{

int n;

cout << "Enter an positive integer: ";

cin >> n;

checkPrime(n);

return 0;

}

void checkPrime(int n)

{

int i, flag = 1;

if(n == 1){

cout << n << " is not a prime\n";

return;

}

for(i = 0; ; i++){

if(n == prime[i]){

flag = 1;

break;

}

if(prime[i] > sqrt(n))

break;

else if(n % prime[i] == 0){

flag = 0;

break;

}

else

flag = 1;

}

if(flag == 1)

cout << n << " is a prime\n";

else

cout << n << " is not a prime\n";

}

**Output:**



**Problem No: 11**

**Topic: Number Theory**

**Problem Title:**

Given a positive integer, determine whether it is Mersenne prime.

**Objectives:**

To determine Mersenne prime numbers.

**Source Code:**

#include <iostream>

#include <cmath>

using namespace std;

bool checkPrime(int n);

void checkMersennePrime(int n);

int prime[] = {2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101};

int main()

{

int n;

cout << "Enter an positive integer: ";

cin >> n;

if(checkPrime(n) == 1)

checkMersennePrime(n);

else

cout << n << " is not a prime / Mersenne prime.\n";

return 0;

}

bool checkPrime(int n)

{

int i, flag = 1;

if(n == 1)

return 0;

for(i = 0; ; i++){

if(n == prime[i]){

flag = 1;

break;

}

if(prime[i] > sqrt(n))

break;

else if(n % prime[i] == 0){

flag = 0;

break;

}

else

flag = 1;

}

if(flag == 1)

return 1;

else

return 0;

}

void checkMersennePrime(int n)

{

int n1, flag = 0, p;

n1 = n + 1;

for(int i = 1; ; i++){

p = (int)pow(2, i);

if(p > n1)

break;

else if(p == n1){

flag = 1;

break;

}

}

if(flag == 1)

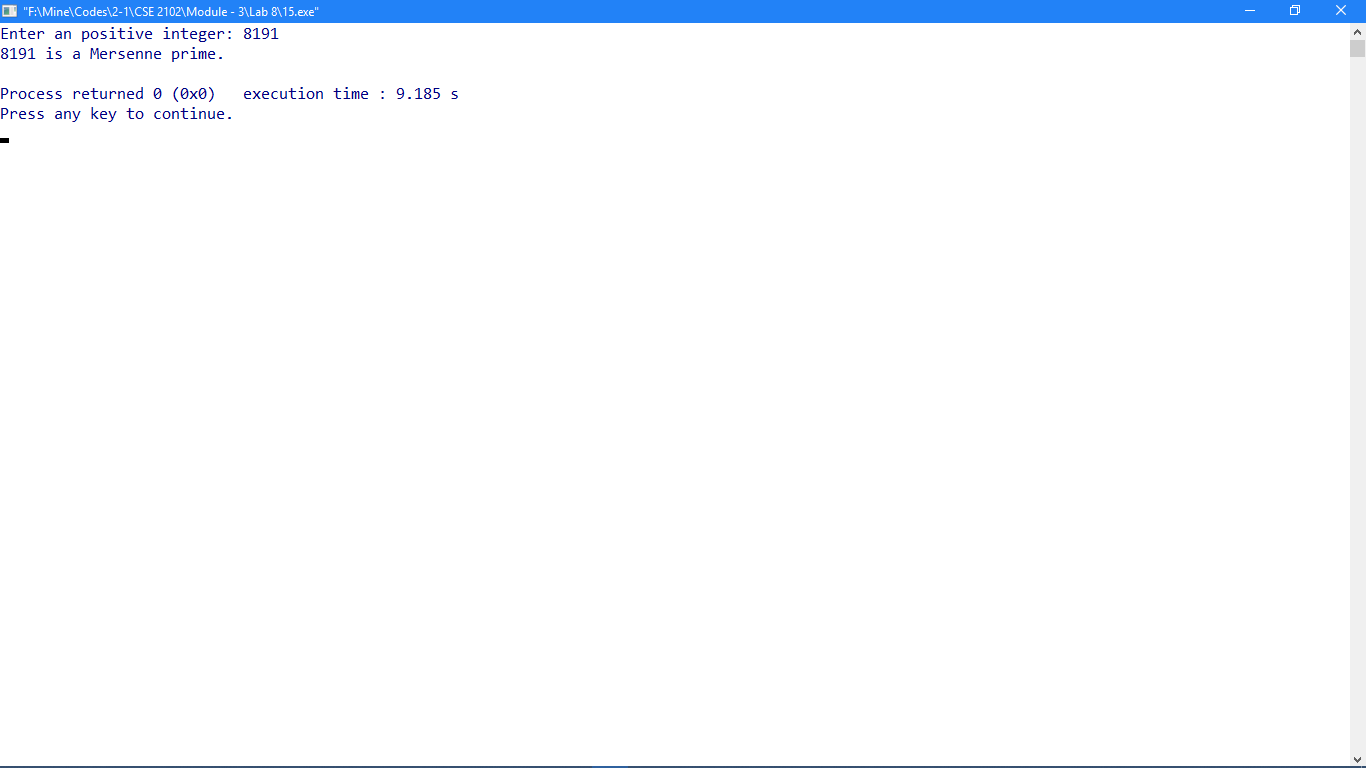
cout << n << " is a Mersenne prime.\n";

else

cout << n << " is not a Mersenne prime.\n";

}

**Output:**



**Problem No: 12**

**Topic: Number Theory**

**Problem Title:**

The polynomial f(n) = n2-n+41 has the interesting property that f(n) is prime for all positive integers n not exceeding 40. Given a positive integer n, find the value of f(n) whether f(n) is prime or not. **Objectives:**

To determine prime numbers.

**Source Code:**

#include <iostream>

#include <cmath>

using namespace std;

void checkPrime(int n);

int func(int n){return ((n \* n) - n + 41);};

int prime[] = {2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101};

int main(){

int n;

cout << "Enter an positive integer: ";

cin >> n;

cout << "f(n) = n^2 - n + 41\n" << "f(" << n << ") = " << func(n) << endl;

checkPrime(func(n));

return 0;

}

void checkPrime(int n)

{

int i, flag = 1;

if(n == 1){

cout << n << " is not a prime\n";

return;

}

for(i = 0; ; i++){

if(n == prime[i]){

flag = 1;

break;

}

if(prime[i] > sqrt(n))

break;

else if(n % prime[i] == 0){

flag = 0;

break;

}

else

flag = 1;

}

if(flag == 1)

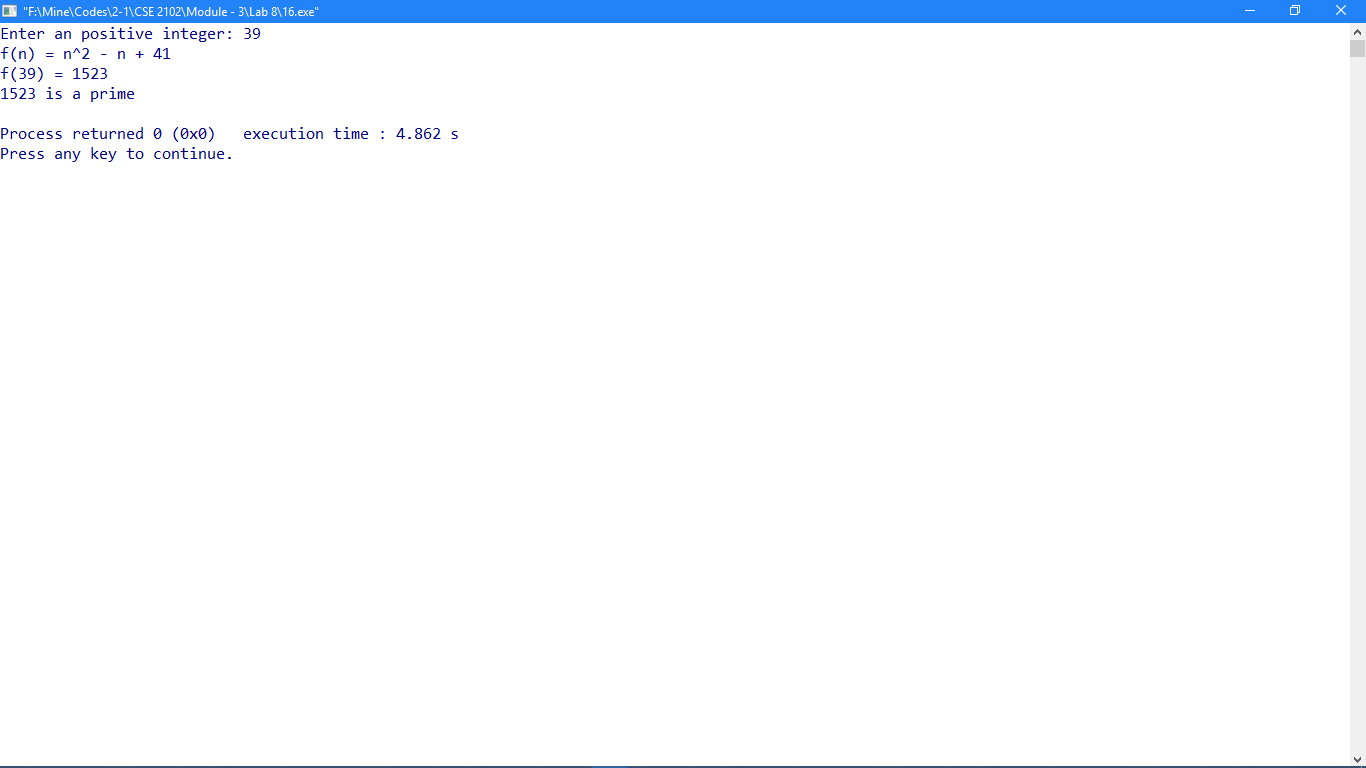
cout << n << " is a prime\n";

else

cout << n << " is not a prime\n";

}

**Output:**



**Problem No: 13**

**Topic: Number Theory**

**Problem Title:**

Given an integer n, whether f(n) = n2+1 is prime or not.

**Objectives:**

To determine prime numbers.

**Source Code:**

#include <iostream>

#include <cmath>

using namespace std;

void checkPrime(int n);

int func(int n){return ((n \* n) +1);};

int prime[] = {2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101};

int main()

{

int n;

cout << "Enter an positive integer: ";

cin >> n;

cout << "f(n) = n^2 + 1\n" << "f(" << n << ") = " << func(n) << endl;

checkPrime(func(n));

return 0;

}

void checkPrime(int n)

{

int i, flag = 1;

if(n == 1){

cout << n << " is not a prime\n";

return;

}

for(i = 0; ; i++){

if(n == prime[i]){

flag = 1;

break;

}

if(prime[i] > sqrt(n))

break;

else if(n % prime[i] == 0){

flag = 0;

break;

}

else

flag = 1;

}

if(flag == 1)

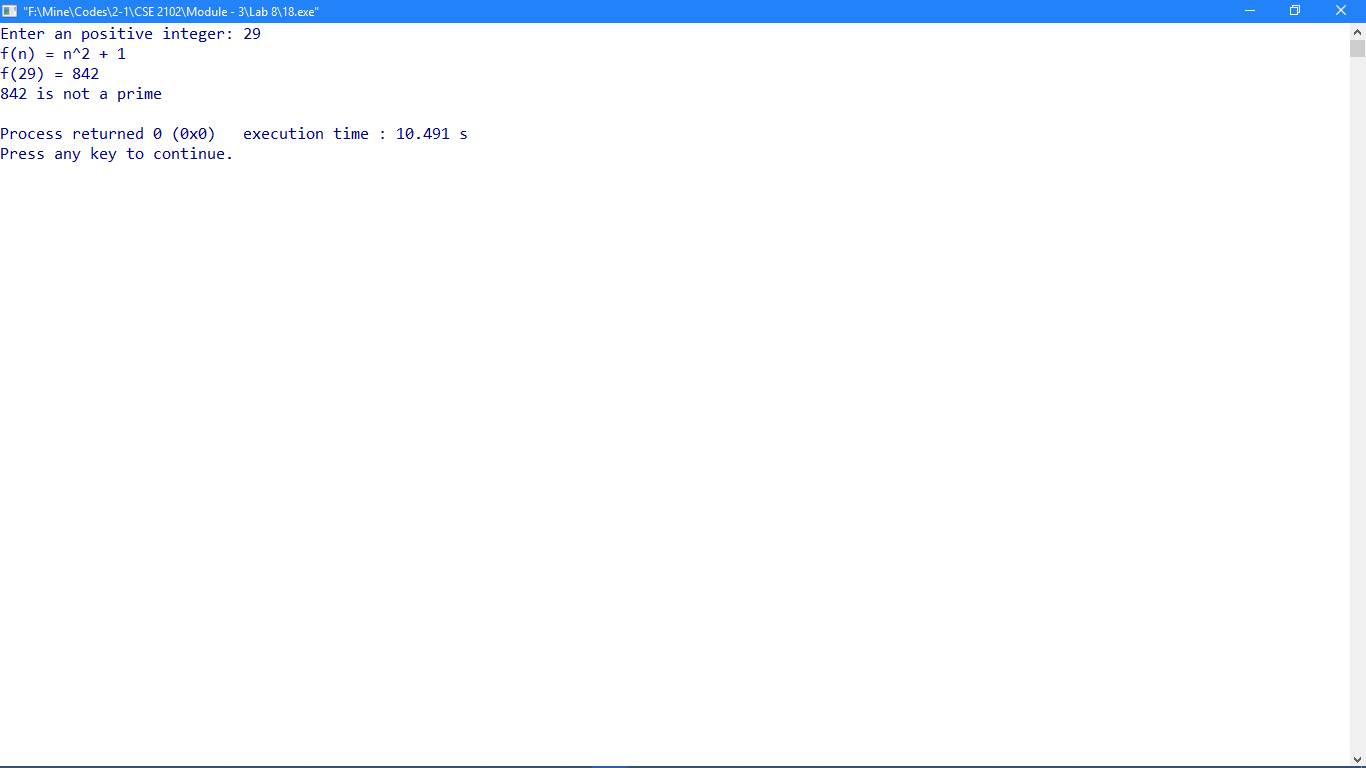
cout << n << " is a prime\n";

else

cout << n << " is not a prime\n";

}

**Output:**



**Problem No: 14**

**Topic: Number Theory**

**Problem Title:**

Given a positive number n, whether it is prime or not. If n is prime, check whether n and n+2 are Twin primes or not.

**Objectives:**

To determine twin prime conjecture.

**Source Code:**

#include <iostream>

#include <cmath>

using namespace std;

bool checkPrime(int n);

bool twinPrime(int n);

int prime[] = {2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97, 101};

int main()

{

int n;

cout << "Enter an positive integer: ";

cin >> n;

if(checkPrime(n) == 1){

if(twinPrime(n) == 1)

cout << n << " and " << n + 2 << " are twin primes.\n";

else

cout << n << " and " << n + 2 << " are not twin primes.\n";

}

else

cout << n << " is not a prime.\n";

return 0;

}

bool checkPrime(int n)

{

int i, flag = 1;

if(n == 1)

return 0;

for(i = 0; ; i++){

if(n == prime[i]){

flag = 1;

break;

}

if(prime[i] > sqrt(n))

break;

else if(n % prime[i] == 0){

flag = 0;

break;

}

else

flag = 1;

}

if(flag == 1)

return 1;

else

return 0;

}

bool twinPrime(int n)

{

if(checkPrime(n + 2) == 1)

return 1;

else

return 0;

}

**Output:**



**Problem No: 15**

**Topic: Number Theory**

**Problem Title:**

Given two positive integers, find their greatest common divisor using the Euclidean algorithm.

**Objectives:**

To implement Euclidean Algorithm.

**Source Code:**

#include <iostream>

using namespace std;

int gcdEucledean(int a, int b);

int main()

{

int a, b;

cout << "Enter two positive integer: ";

cin >> a >> b;

cout << "Greatest common divisor of " << a << " and " << b << " is " << gcdEucledean(a, b) << endl;

return 0;

}

int gcdEucledean(int a, int b)

{

int x = a;

int y = b;

int r;

while(y != 0){

r = x % y;

x = y;

y = r;

}

return x;

}

**Output:**

