**Experiment No. 10**

**Title :** Implementation of Binary Search in C++

**Problem Statement :** Implementing Binary Search algorithm in C++

**Algorithm :**

**S1 :** Start

**S2 :** Declare an array and loop control variables and binarysearch().

**S3 :** Ask for the array input and element to be searched from user.

**S4 :** with the array, first, last indexes and element to be searched as arguments to be sent to binarysearch function.

**S5 :** Calculate mid and check if key value is equal to mid value in array else if it is smaller the call binarysearch function with last as mid-1 if it is greater the call function with first as mid+1

**S6 :** If element is found print position else print element not found

**S7 :** Stop

**Code :**

#include <iostream>

using namespace std;

// If found, location of x in the array is returned.

// otherwise -1 is returned.

int binarySearch(int arr[], int first, int last, int x)

{

if (last >= first) {

int mid = first + (last- first) / 2;

// If the element is present at the middle of the array

if (arr[mid] == x)

return mid;

// If element is smaller than mid, then

// it can only be present in left subarray

if (arr[mid] > x)

return binarySearch(arr, first, mid - 1, x);

// Else the element can only be present

// in right subarray

return binarySearch(arr, mid + 1, last, x);

}

// If the element is not found.

return -1;

}

int main()

{

int arr[20],n,x,i;

cout<<"How many elements?";

cin>>n;

cout<<"\nEnter elements of the array\n";

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

cin>>arr[i];

cout<<"\nEnter element to search:";

cin>>x;

int result = binarySearch(arr, 0, n - 1, x);

if (result == -1)

cout << "Element is not present in array.";

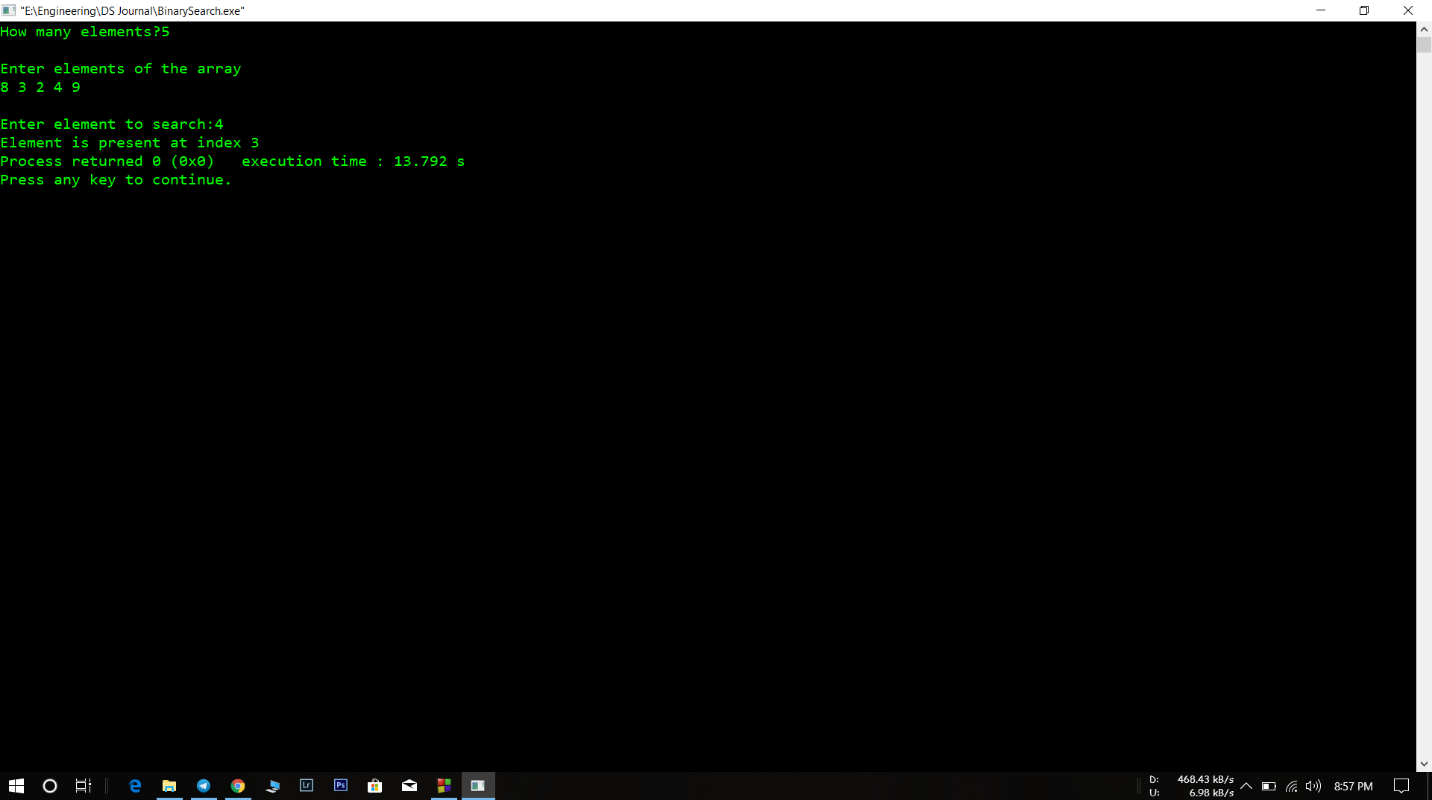
else

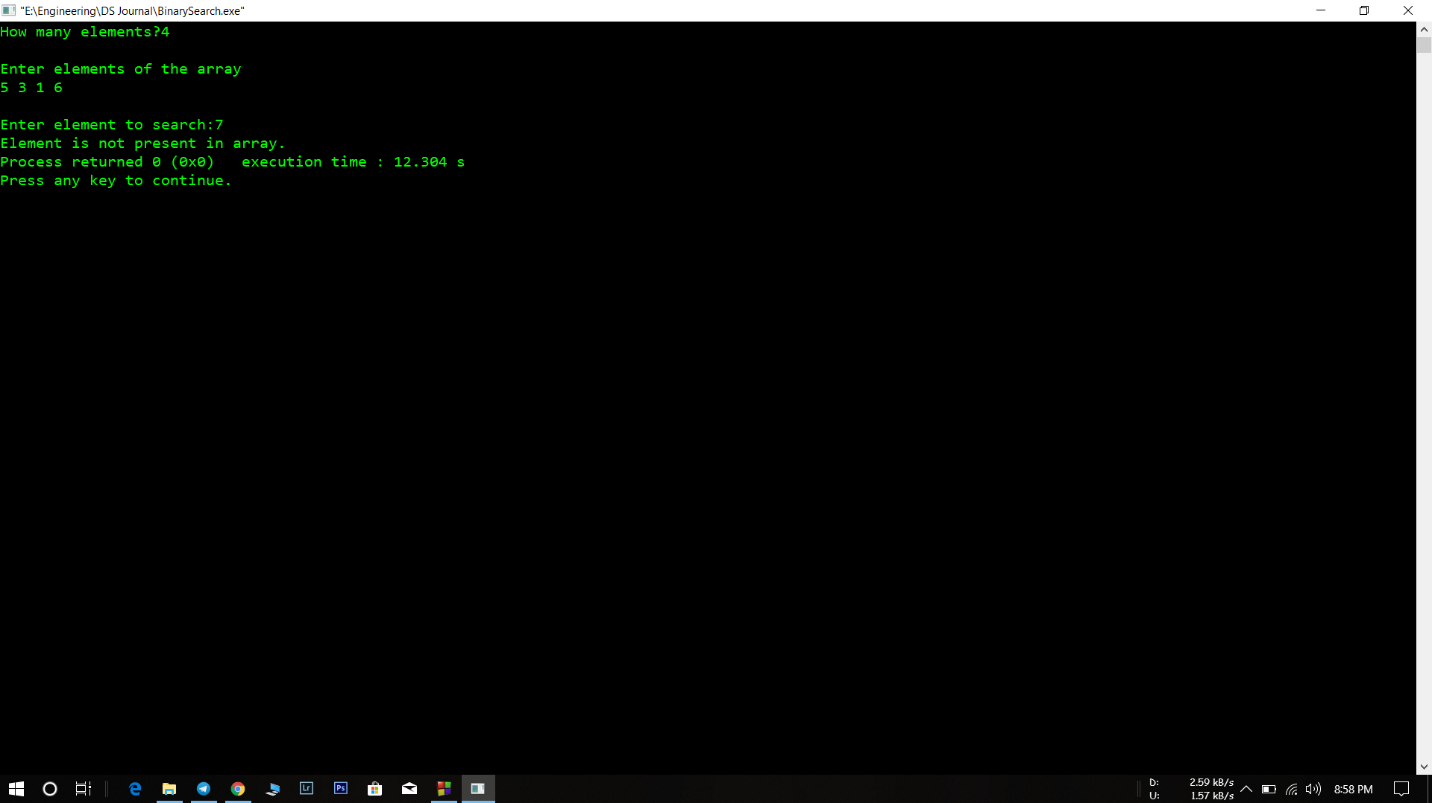
cout << "Element is present at index " <<result;

return 0;

}

**Output :**

****

****

**Analysis :**

* The array in which we search has to be sorted. If the array is not sorted the algorithm would not work and be totally inefficient.
* It is more complicated than linear search but as compared to efficiency for smaller numbers in array it might increase time complexity but for arrays with huge values the algorithm is efficient.