	Department of Computer Science and Engineering Faculty of Engineering, South Eastern University of Sri Lanka		
	Subject	**CS53003: Data Structure and Algorithms**	
	Batch	**E18**	**Semester**	**5**

Lab no and title : Lab 05: Searching
Name : G.W.P.R.R. Wijesinghe
Reg No : SEU-IS-18-EG-013
Submission Date : 18-01-2022

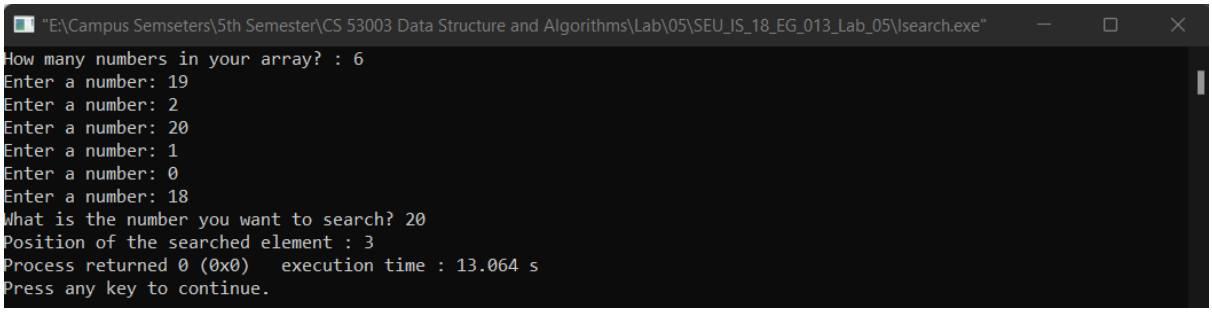
---

01. Given an integer and an array of integers, write a function named linear_search that returns the number of comparisons performed doing a linear search. The function should take 3 arguments:
    - The number searched for
    - The array of integers
    - The number of elements in the array

    If the number searched for is not in the array then the function should return -1.

Start here | lsearch.cpp | bsearch.cpp

```cpp
#include<iostream>

using namespace std;

int linear_search(int num, int arr[], int arrSize){
    for (int i=0; i<arrSize; i++){
        if (arr[i]==num){
            return i+1;
        }
    }
    return -1;
}

int main(){
    int aSize, searchNum;

    cout << "How many numbers in your array? : ";
    cin >>  aSize;
    int A[aSize]={};
    for (int j=0; j<aSize; j++){
        cout << "Enter a number: ";
        cin >> A[j];
    }

    cout << "What is the number you want to search? ";
    cin >> searchNum;
    cout << "Position of the searched element : "<<linear_search(searchNum,A,aSize);
}
```



02. Given an integer and a sorted array of integers, write a function named binary_search that prints the number of comparisons performed doing a binary search. The function should take 3 arguments:
    - The number searched for
    - The array of integers
    - The number of elements in the array

    If the number searched for is not in the array then the function should return -1.

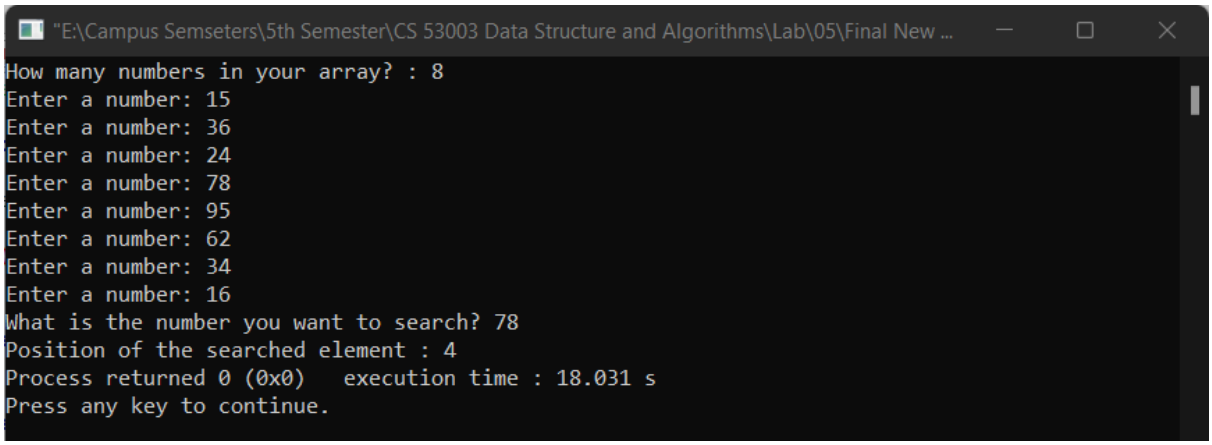
Start here | lsearch.cpp | bsearch.cpp

```cpp
#include<iostream>

using namespace std;

int binarySearch(int num, int arr[], int arrSize){
    int first = 0;
    int last = arrSize-1;
    int totalCount = 0;

    while (first <= last){
        int mid = (first+last)/2;
        totalCount = totalCount+1;
        if (num == arr[mid]){
            return totalCount;
        }
        else if (num < arr[mid]){
            last = mid-1;
        }
        else {
            first = mid+1;
        }

    }
    return -1;
}

int main(){
    int aSize, searchNum;

    cout << "How many numbers in your array? : ";
    cin >>  aSize;
    int A[aSize]={};
    for (int j=0; j<aSize; j++){
        cout << "Enter a number: ";
        cin >> A[j];
    }

    cout << "What is the number you want to search? ";
    cin >> searchNum;
    cout << "Position of the searched element : "<<binarySearch(searchNum,A,aSize);
}
```

3. Write a C++ program to implement binary search using recursion

Start here | RecusBsort.cpp | EX04.cpp

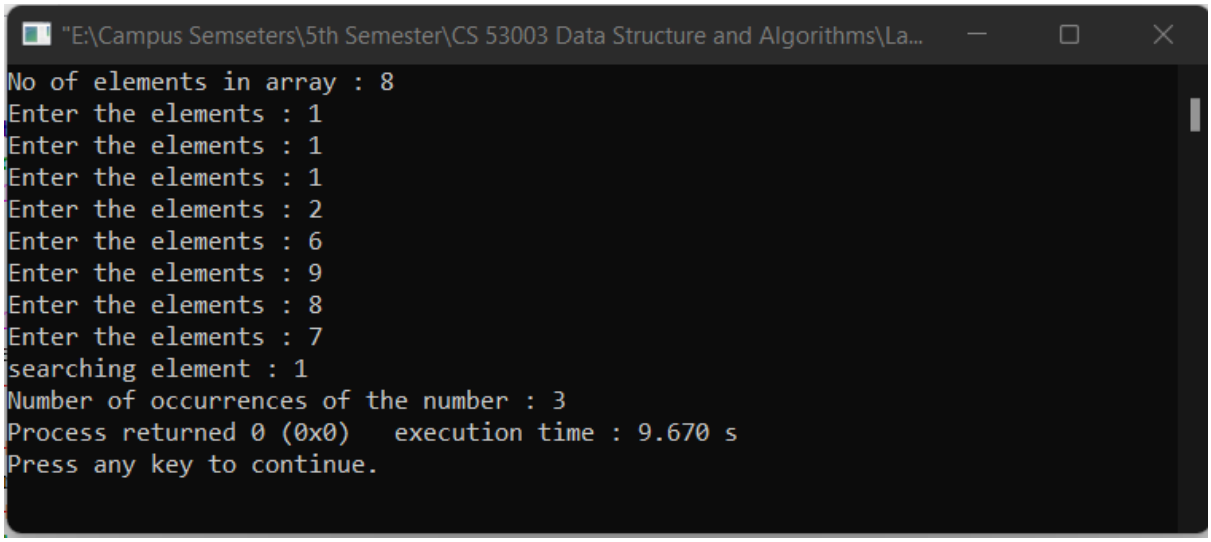
```cpp
#include<iostream>

using namespace std;

int binarySearch(int arr[],int first, int last, int num){
        int mid = (first+last)/2;
        if (num == arr[mid]){
            return mid+1;
        }
        else if (num < arr[mid]){
            binarySearch(arr,first,mid-1,num);
        }
        else {
            binarySearch(arr,first+1,last,num);
        }

}
int main(){
    int aSize, searchNum;
    int start = 0;

    cout << "How many numbers in your array? : ";
    cin >>  aSize;
    int A[aSize]={};
    for (int j=0; j<aSize; j++){
        cout << "Enter a number: ";
        cin >> A[j];
    }

    cout << "What is the number you want to search? ";
    cin >> searchNum;
    cout << "Position of the searched element : "<<binarySearch(A,start,aSize-1,searchNum);
}
```



4. Write a C++ program to find out the number of occurrences of a number in a sorted array using binary search.

Start here | RecusBsort.cpp | EX04.cpp

```cpp
#include<iostream>

using namespace std;

int binary_search(int arr[] , int first , int last , int key){
    if(first > last)
        return -1;
    else{
        int mid = (first + last)/2;
            if(key == arr[mid])
                return mid;
            else if (key<arr[mid])
                return binary_search(arr , first , mid-1 , key);
            else
                return binary_search(arr , mid+1 , last , key);
    }
}

int count_occur (int arr[] , int n , int key){
    int result = binary_search(arr , 0 , n-1 , key);
    if (result == -1)
        return 0;

    int count = 1;
    int i = result-1;
    while (i>=0 && arr[i] == key)
        count++ , i--;

    int j = result + 1;
    while (j<n && arr[j] == key)
        count++ , j++;
        return count;
}

int main(){
    int elements , key , n ;
    int start=0 ;

    cout << "No of elements in array : ";
    cin >>  elements ;

    int arr[elements]={};
    for (int i=0; i<elements; i++){
        cout << "Enter the elements : ";
        cin >> arr[i];
    }

    cout<< "searching element : ";
    cin>>key;

    int X = sizeof(arr)/sizeof(arr[0]);
    int results = count_occur (arr , X , key);

    cout << "Number of occurrences of the number : "<< results;
}
```


"E:\Campus Semseters\5th Semester\CS 53003 Data Structure and Algorithms\La...
No of elements in array : 8
Enter the elements : 1
Enter the elements : 1
Enter the elements : 1
Enter the elements : 2
Enter the elements : 6
Enter the elements : 9
Enter the elements : 8
Enter the elements : 7
searching element : 1
Number of occurrences of the number : 3
Process returned 0 (0x0)   execution time : 9.670 s
Press any key to continue.

***