Experiment-1.1

Aim of the Experiment :

Write a program for implementation of following searching techniques on linear data structures:

A) Linear Search

B) Binary Search

1. Problem Description :

Linear Search: Linear Search is a sequential search algorithm that starts at one end and goes through each element of the data structure until the desired element is found, otherwise the search continues till the end of the data structure.

2. Algorithm :

```
Linear Search ( Array A, Value x)

Step 1: Set i to 1

Step 2: if i > n then go to step 7

Step 3: if A[i] = x then go to step 6

Step 4: Set i to i + 1

Step 5: Go to Step 2

Step 6: Print Element x Found at index i and go to step 8

Step 7: Print element not found

Step 8: Exit
```

3. Complexity Analysis:

Time Complexity:

1) Best Case: O(1). When the element is present at the first index.

2) Worst Case: O(N). When the element is present at the last index or not present in the array.

3) Average Case: O(N)

Space Complexity: O(1). As no extra space is used to search the element.

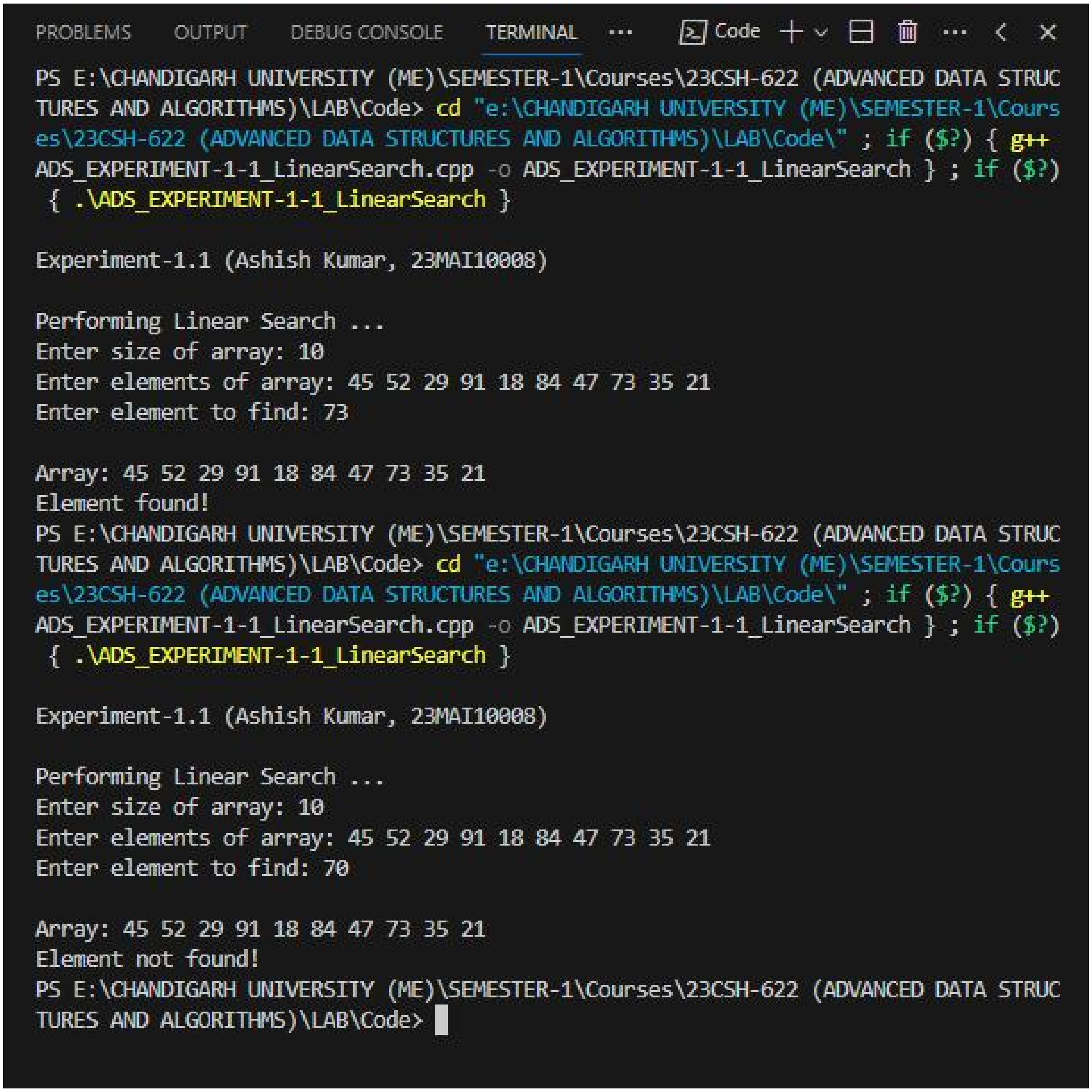
4. Pseudo Code :

```
procedure linear_search (list, value)

    for each item in the list

        if match item == value

        return the item's location

        end if

    end for

end procedure
```

5. Source Code for Experiment :

```
#include<iostream>
using namespace std;

void printArray(int arr[], int size){
    for(int i=0;i<size;i++){
        cout<<arr[i]<<" ";
    }
    cout<<endl;
}

bool linearsearch(int arr[], int size, int element){
    for(int i=0;i<size;i++){
        if(arr[i]==element){
            return true;
        }
    }
    return false;
}

int main(){
    cout<<"\nExperiment-1.1 (Ashish Kumar, 23MAI10008)"<<endl<<endl;
    cout<<"Performing Linear Search ..."<<endl;
    int n;
    int arr[1000];
    cout<<"Enter size of array: ";
    cin>>n;
    cout<<"Enter elements of array: ";
    for(int i=0; i<n; i++){
        cin>>arr[i];
    }

    int key;
    cout<<"Enter element to find: ";
    cin>>key;
    cout<<"\nArray: ";
    printArray(arr,n);

    // Linear Search
    if(linearsearch(arr,n,key)){
        cout<<"Element found!"<<endl;
    }
    else{
        cout<<"Element not found!"<<endl;
    }

}
```

6. Result/Output :



```
PS E:\CHANDIGARH UNIVERSITY (ME)\SEMESTER-1\Courses\23CSH-622 (ADVANCED DATA STRUC
TURES AND ALGORITHMS)\LAB\Code> cd "e:\CHANDIGARH UNIVERSITY (ME)\SEMESTER-1\Cours
es\23CSH-622 (ADVANCED DATA STRUCTURES AND ALGORITHMS)\LAB\Code\" ; if ($?) { g++
ADS_EXPERIMENT-1-1_LinearSearch.cpp -o ADS_EXPERIMENT-1-1_LinearSearch } ; if ($?)
 { .\ADS_EXPERIMENT-1-1_LinearSearch }

Experiment-1.1 (Ashish Kumar, 23MAI10008)

Performing Linear Search ...
Enter size of array: 10
Enter elements of array: 45 52 29 91 18 84 47 73 35 21
Enter element to find: 73

Array: 45 52 29 91 18 84 47 73 35 21
Element found!
PS E:\CHANDIGARH UNIVERSITY (ME)\SEMESTER-1\Courses\23CSH-622 (ADVANCED DATA STRUC
TURES AND ALGORITHMS)\LAB\Code> cd "e:\CHANDIGARH UNIVERSITY (ME)\SEMESTER-1\Cours
es\23CSH-622 (ADVANCED DATA STRUCTURES AND ALGORITHMS)\LAB\Code\" ; if ($?) { g++
ADS_EXPERIMENT-1-1_LinearSearch.cpp -o ADS_EXPERIMENT-1-1_LinearSearch } ; if ($?)
 { .\ADS_EXPERIMENT-1-1_LinearSearch }

Experiment-1.1 (Ashish Kumar, 23MAI10008)

Performing Linear Search ...
Enter size of array: 10
Enter elements of array: 45 52 29 91 18 84 47 73 35 21
Enter element to find: 70

Array: 45 52 29 91 18 84 47 73 35 21
Element not found!
PS E:\CHANDIGARH UNIVERSITY (ME)\SEMESTER-1\Courses\23CSH-622 (ADVANCED DATA STRUC
TURES AND ALGORITHMS)\LAB\Code> 
```

1. Problem Description :

Binary Search: Binary search is a search algorithm that finds position of a target value within a sorted array. Binary search compares target value to middle element of the array. If they are not equal, search continues on the remaining half where element lie.This process keeps on repeating until the target value is found.

2. Algorithm :

```
Binary_Search(a, lower_bound, upper_bound, val)
Step 1: Set beg = lower_bound, end = upper_bound, pos = - 1
Step 2: Repeat steps 3 and 4 while beg <=end
Step 3: Set mid = (beg + end)/2
Step 4: if a[mid] = val
        set pos = mid
        print pos
        go to step 6
        else if a[mid] > val
        set end = mid - 1
        else
        set beg = mid + 1
        [end of if]
        [end of loop]
Step 5: if pos = -1
print "value is not present in the array"
[end of if]
Step 6: Exit
```

3. Complexity Analysis:

Time Complexity:

1) Best Case: O(1). When the element is at the middle index of the array.

2) Worst Case: O(logN). When the element is present in the first position or not present in the array.

3) Average Case: O(logN).

Space Complexity: O(1). As no extra space is used to search the element.

4. Pseudo Code :

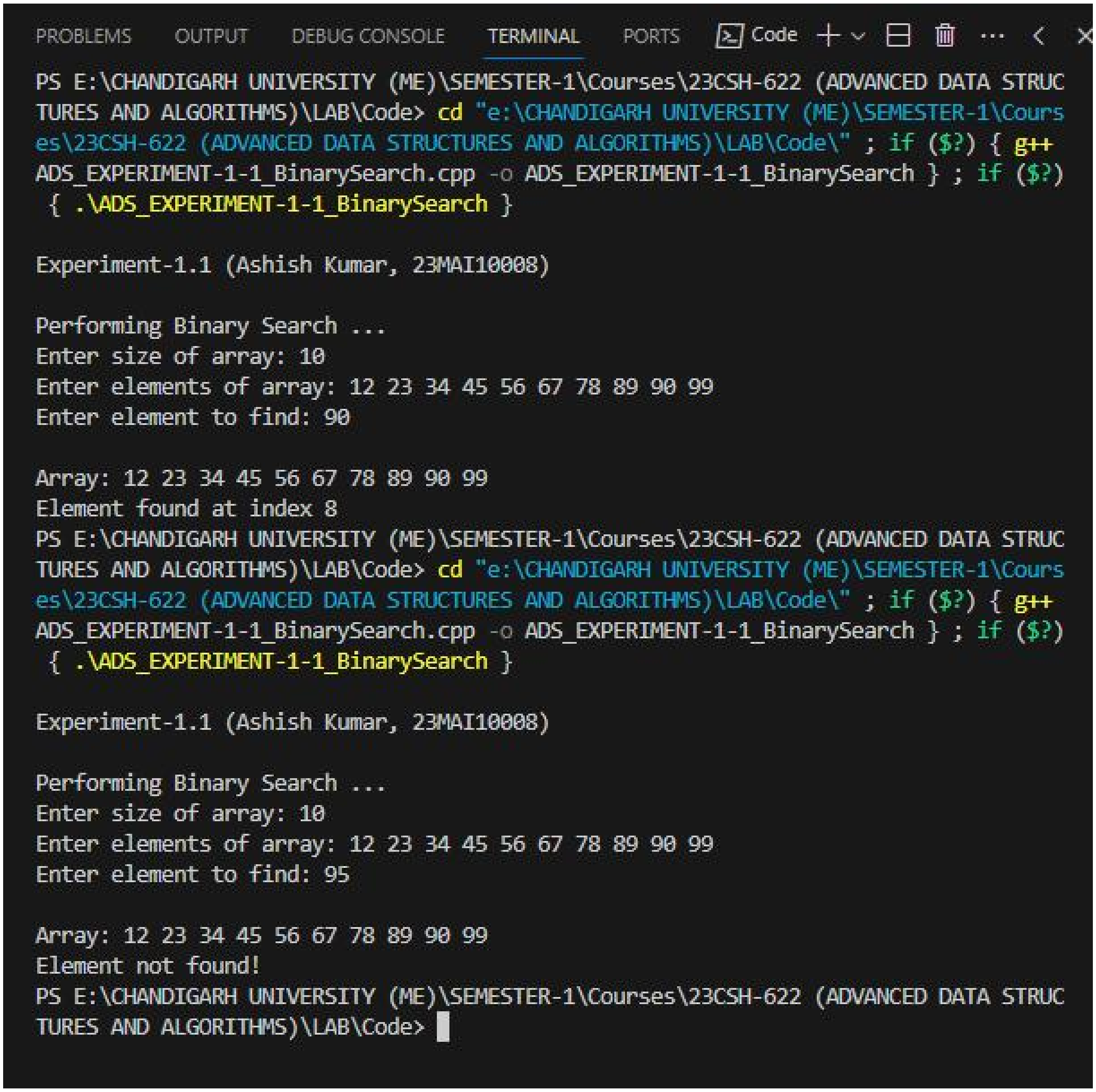
```
Procedure binary_search
    A ← sorted array
    n ← size of array
    x ← value to be searched
    Set lowerBound = 1
    Set upperBound = n
    while x not found
        if upperBound < lowerBound
            EXIT: x does not exists.
        set midPoint = lowerBound &plus; ( upperBound - lowerBound ) / 2
        if A[midPoint] < x
            set lowerBound = midPoint &plus; 1
        if A[midPoint] > x
            set upperBound = midPoint - 1
            if A[midPoint] = x
                  EXIT: x found at location midPoint
      end while
end procedure
```

5. Source Code for Experiment :

```
#include<iostream>
using namespace std;

void printArray(int arr[], int size){
  for(int i=0;i<size;i++){
    cout<<arr[i]<<" ";
  }
  cout<<endl;
}

int binarysearch(int arr[], int n, int key){
  int s=0;
  int e=n-1;
  int mid=s+(e-s)/2;

  while(s<=e){
    if(arr[mid]==key){
      return mid;
    }
```

```
        if(arr[mid]<key){
            s=mid+1;
        }
        else{
            e=mid-1;
        }
        mid=s+(e-s)/2;
    }
    return -1;
}

int main(){

    cout<<"\nExperiment-1.1 (Ashish Kumar, 23MAI10008)"<<endl<<endl;
    cout<<"Performing Binary Search ..."<<endl;

    int n;
    int arr[1000];
    cout<<"Enter size of array: ";
    cin>>n;
    cout<<"Enter elements of array: ";
    for(int i=0; i<n; i++){
        cin>>arr[i];
    }
    int key;
    cout<<"Enter element to find: ";
    cin>>key;

    cout<<"\nArray: ";
    printArray(arr,n);

    // Binary Search
    int index= binarysearch(arr,n,key);
    if(index == -1){
        cout<<"Element not found!"<<endl;
    }
    else{
        cout<<"Element found at index "<<index<<endl;
    }

}
```

6. Result/Output :



```
PS E:\CHANDIGARH UNIVERSITY (ME)\SEMESTER-1\Courses\23CSH-622 (ADVANCED DATA STRUC
TURES AND ALGORITHMS)\LAB\Code> cd "e:\CHANDIGARH UNIVERSITY (ME)\SEMESTER-1\Cours
es\23CSH-622 (ADVANCED DATA STRUCTURES AND ALGORITHMS)\LAB\Code\" ; if ($?) { g++
ADS_EXPERIMENT-1-1_BinarySearch.cpp -o ADS_EXPERIMENT-1-1_BinarySearch } ; if ($?)
 { .\ADS_EXPERIMENT-1-1_BinarySearch }

Experiment-1.1 (Ashish Kumar, 23MAI10008)

Performing Binary Search ...
Enter size of array: 10
Enter elements of array: 12 23 34 45 56 67 78 89 90 99
Enter element to find: 90

Array: 12 23 34 45 56 67 78 89 90 99
Element found at index 8
PS E:\CHANDIGARH UNIVERSITY (ME)\SEMESTER-1\Courses\23CSH-622 (ADVANCED DATA STRUC
TURES AND ALGORITHMS)\LAB\Code> cd "e:\CHANDIGARH UNIVERSITY (ME)\SEMESTER-1\Cours
es\23CSH-622 (ADVANCED DATA STRUCTURES AND ALGORITHMS)\LAB\Code\" ; if ($?) { g++
ADS_EXPERIMENT-1-1_BinarySearch.cpp -o ADS_EXPERIMENT-1-1_BinarySearch } ; if ($?)
 { .\ADS_EXPERIMENT-1-1_BinarySearch }

Experiment-1.1 (Ashish Kumar, 23MAI10008)

Performing Binary Search ...
Enter size of array: 10
Enter elements of array: 12 23 34 45 56 67 78 89 90 99
Enter element to find: 95

Array: 12 23 34 45 56 67 78 89 90 99
Element not found!
PS E:\CHANDIGARH UNIVERSITY (ME)\SEMESTER-1\Courses\23CSH-622 (ADVANCED DATA STRUC
TURES AND ALGORITHMS)\LAB\Code>
```

Learning outcomes (What I have learnt):

1. I learnt about how to search an element in an array.

2. I learnt about how to perform linear search in an array.

3. I learnt about how to perform binary serach in an array.

4. I learnt about comparison between linear and binary search.

5. I learnt about time and space complexity of linear and binary search.