<u>Previous Lab Discussion</u>

Linear Search

Linear search is a simple search algorithm that works by sequentially searching through a list of elements until the target element is found or the end of the list is reached. Linear search is inefficient for large lists, but it is simple to implement and can be used for any type of data structure.

Binary Search

Binary search is a more efficient search algorithm that works by repeatedly dividing a sorted list in half until the target element is found or the list is empty. Binary search is only efficient for sorted lists, but it is much faster than linear search for large lists.

<u>Current Lab Discussion</u>

Bubble Sorting

Bubble sort is a simple sorting algorithm that works by repeatedly comparing adjacent elements and swapping them if they are in the wrong order. The algorithm starts at the beginning of the array and compares the first two elements. If the first element is greater than the second element, the two elements are swapped. The algorithm then moves on to the next two elements and repeats the process. This process continues until the end of the array is reached. The algorithm then repeats the entire process again, starting at the beginning of the array. This continues until no more swaps are needed, indicating that the array is sorted.

Selection Sorting

Selection sort is another simple sorting algorithm that works by finding the smallest element in the unsorted array and swapping it with the first element in the array. The algorithm then repeats this process for the remaining elements in the array, finding the smallest element in the unsorted subarray and swapping it with the next element in the sorted array. This process continues until the entire array is sorted.

Insertion Sorting

Insertion sort is a sorting algorithm that works by inserting each element of the unsorted array into its correct position in the sorted array. The algorithm starts at the second element in the unsorted array and compares it to the first element in the sorted array. If the second element is smaller than the first element, the second element is inserted before the first element. The algorithm then moves on to the next element in the unsorted array and repeats the process. This process continues until the end of the unsorted array is reached.

TASK-02

- Create a Menu Driven program for Search in the Array. If User press 1 then linear search will be applied and if user press 2 then binary search will be applied. Take input according to the search algorithm requirement**.**

Code:

```
#include<iostream>

using namespace std;

int main()

{

    int size;
    int NumtoFind;

    bool flag = false;

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

    cin>>size;

    int start = 0;

    int end=size-1;

    int array[size];

    cout<<"Enter the values in an array (in ascending order)"<<endl;

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

    {

       cout<<"Enter value "<<i+1<<": ";

       cin>>array[i];

    }

    cout<<"Enter the number you want to find in array: "<<endl;

    cin>>NumtoFind;

    int choice;

         cout<<"Please choose searching method \nPress 1 for linear search and
press 2 for binary search: "<<endl;

         cin>>choice;

         while(start <= end)

         {

         if(choice == 1)

                 {
                        if(array[start] == NumtoFind)
                                {
                         flag = true;
                         break;
                        }
                        else
                                {
                         start++;
                        }
                    }
              else if(choice == 2)
                        {
                                int mid;
                      mid = (start+end)/2;
                        if(array[mid] == NumtoFind)
                                {
                         flag = true;
                         break;
                        }
                        if (array[mid]<NumtoFind)
                        {
                          start=mid+1;
                        }
                    if (array[mid]>NumtoFind)
                    {
                     end=mid-1;
                    }
                   }
        else
                   {
                           cout<<"Invalid choice.";
                           exit(0);
                   }
        }
           if(flag)
           {
              cout<<NumtoFind<<" is founded in an array!"<<endl;
           }
           else{
              cout<<NumtoFind<<" is not found in an array!";
           }
}
```