Lab No: 19 **Date: 2081/12/05**

Title: Write a program to sort the user input data in ascending or descending order using Insertion sort

Insertion sort is a simple sorting algorithm used to sort a collection of elements in a given order. It is less efficient on large lists than more advanced algorithms such as quicksort, heapsort, or merge sort but it is simple to implement and is suitable to sort small data lists.

Insertion sort is one of the simple and comparison-based sorting algorithms. The basic idea behind the algorithm is to virtually divide the given list into two parts: a sorted part and an unsorted part, then pick an element from the unsorted part and insert it in it’s place in the sorted part. It does this till all the elements are placed in the sorted part.

IDE: Visual Studio Code

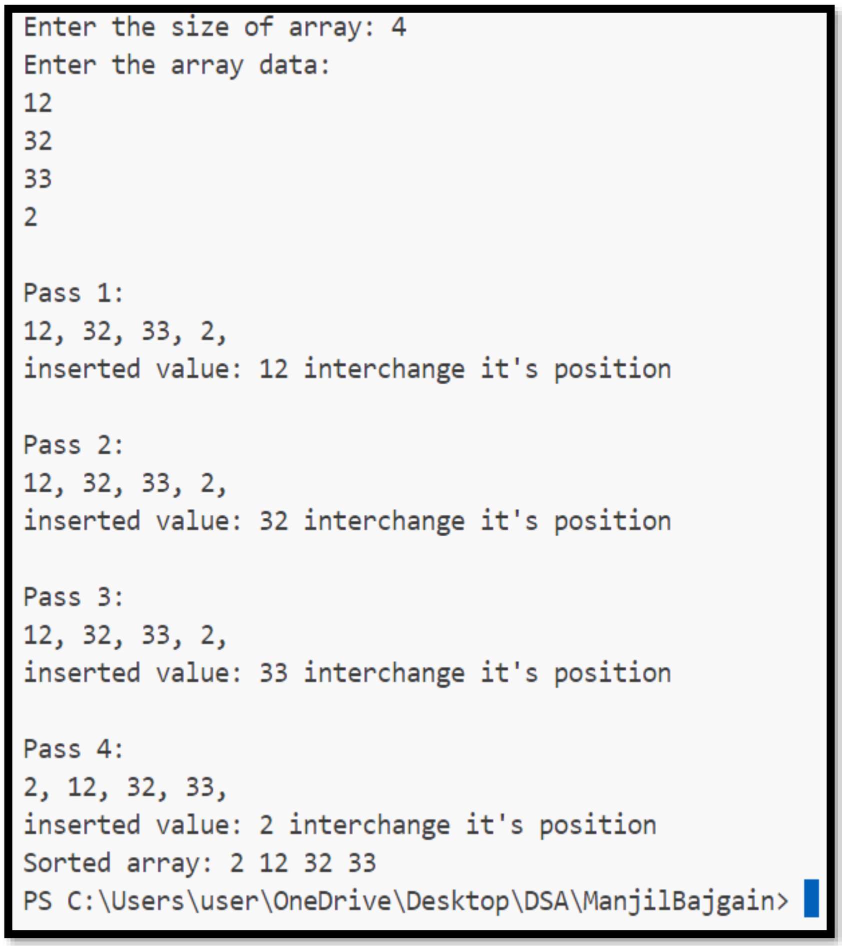
Language: C

Source code:

```c
#include <stdio.h>
#include <conio.h>

void insertionSort(int arr[], int n)
{
    int least, p, i, j, k, temp, pass = 1, key;
    for (i = 0; i < n; i++)
    {
        key = arr[i];
        j = i - 1;
        printf("\nPass %d: \n", pass++);
        while (j >= 0 && arr[j] > key)
        {
            arr[j + 1] = arr[j];
            j = j - 1;
        }
        arr[j + 1] = key;
        for (k = 0; k < n; k++)
        {
            printf("%d, ", arr[k]);
        }
        printf("\n");
        printf("inserted value: %d interchange it's position\n", key);
    }
}
int main()
{
    int n, i;
    printf("Enter the size of array: ");
    scanf("%d", &n);
    int arr[n];
    printf("Enter the array data:\n");//Taking Input
    for (i = 0; i < n; i++)
    {
        scanf("%d", &arr[i]);
    }
    insertionSort(arr, n);
    printf("Sorted array: ");
    for (int i = 0; i < n; i++)
        printf("%d ", arr[i]);
    return 0;
}
```

Output:



```
Enter the size of array: 4
Enter the array data:
12
32
33
2

Pass 1:
12, 32, 33, 2,
inserted value: 12 interchange it's position

Pass 2:
12, 32, 33, 2,
inserted value: 32 interchange it's position

Pass 3:
12, 32, 33, 2,
inserted value: 33 interchange it's position

Pass 4:
2, 12, 32, 33,
inserted value: 2 interchange it's position
Sorted array: 2 12 32 33
PS C:\Users\user\OneDrive\Desktop\DSA\ManjilBajgain>
```

Lab No: 15 **Date: 2081/12/06**

Title: Write a program to search the user input key in the list using Binary search

Binary search is a search algorithm used to find the position of a target value within a sorted array. It works by repeatedly dividing the search interval in half until the target value is found or the interval is empty. The search interval is halved by comparing the target element with the middle value of the search space.

Below is the step-by-step algorithm for Binary Search:

- Divide the search space into two halves by finding the middle index “mid”.
- Compare the middle element of the search space with the key.
- If the key is found at middle element, the process is terminated.
- If the key is not found at middle element, choose which half will be used as the next search space.
- If the key is smaller than the middle element, then the left side is used for next search.
- This process is continued until the key is found or the total search space is exhausted.

IDE: Visual Studio Code

Language: C

Source code:

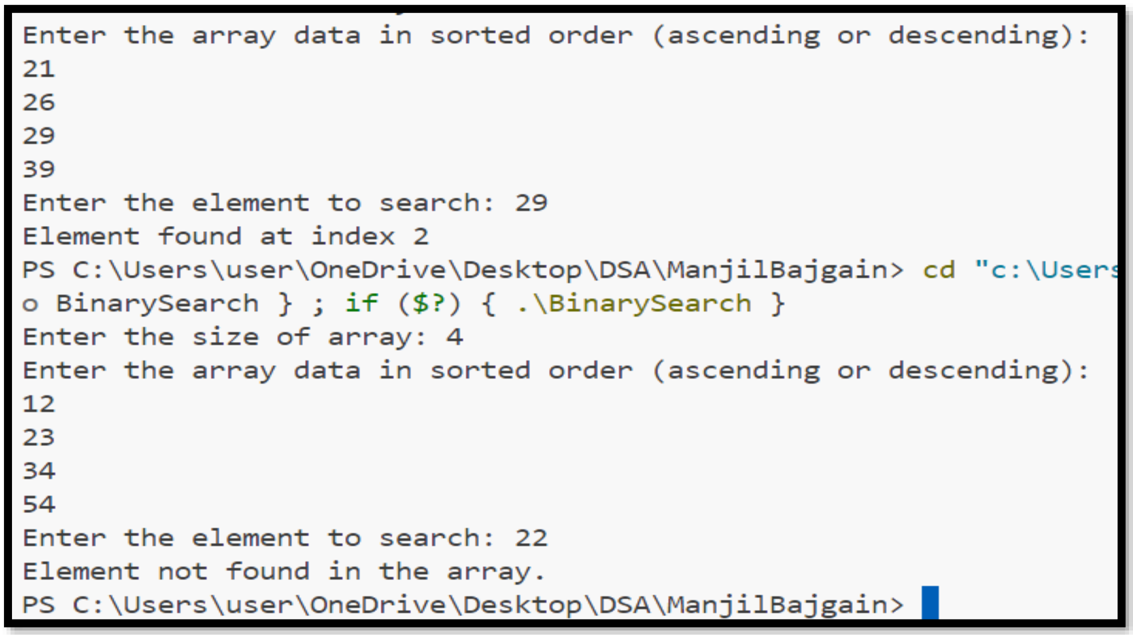
```
#include <stdio.h>
void binarySearch(int arr[], int n, int key)
{
    int low = 0, high = n - 1, mid, flag = 0;
    while (low <= high)
    {
        mid = (low + high) / 2;

        if (arr[mid] == key)
        {
            printf("Element found at index %d\n", mid);
            flag = 1;
            break;
        }
        else if (arr[mid] < key)
        {
            low = mid + 1;
        }
        else
        {
            high = mid - 1;
        }
    }
    if (!flag)
    {
        printf("Element not found in the array.\n");
    }
}
```

```
int main()
{
    int n, i, key;
    printf("Enter the size of array: ");
    scanf("%d", &n);
    int arr[n];
    printf("Enter the array data in sorted order (ascending or descending):\n"); // Taking
input from user
    for (i = 0; i < n; i++)
    {
        scanf("%d", &arr[i]);
    }
    printf("Enter the element to search: ");
    scanf("%d", &key);

    binarySearch(arr, n, key); // Calling Binary Search function
    return 0;
}
```

Output:



```
Enter the array data in sorted order (ascending or descending):
21
26
29
39
Enter the element to search: 29
Element found at index 2
PS C:\Users\user\OneDrive\Desktop\DSA\ManjilBajgain> cd "c:\Users
o BinarySearch } ; if ($?) { .\BinarySearch }
Enter the size of array: 4
Enter the array data in sorted order (ascending or descending):
12
23
34
54
Enter the element to search: 22
Element not found in the array.
PS C:\Users\user\OneDrive\Desktop\DSA\ManjilBajgain>
```

Lab No: 14 **Date: 2081/12/06**

Title: Write a program to search the user input key in the list using linear search

Linear Search is a sequential searching algorithm in C that is used to find an element in a list. Linear Search compares each element of the list with the key till the element is found or we reach the end of the list.

To search for the given element using linear search, follow the below approach:

- Start traversing from the start of the dataset.
- Compare the current element with the key (element to be searched).
- If the element is equal to the key, return index.
- Else, increment the index and repeat the step 2 and 3.
- If we reach the end of the list without finding the element equal to the key, return some value to represent that the element is not found.

IDE: Visual Studio Code

Language: C

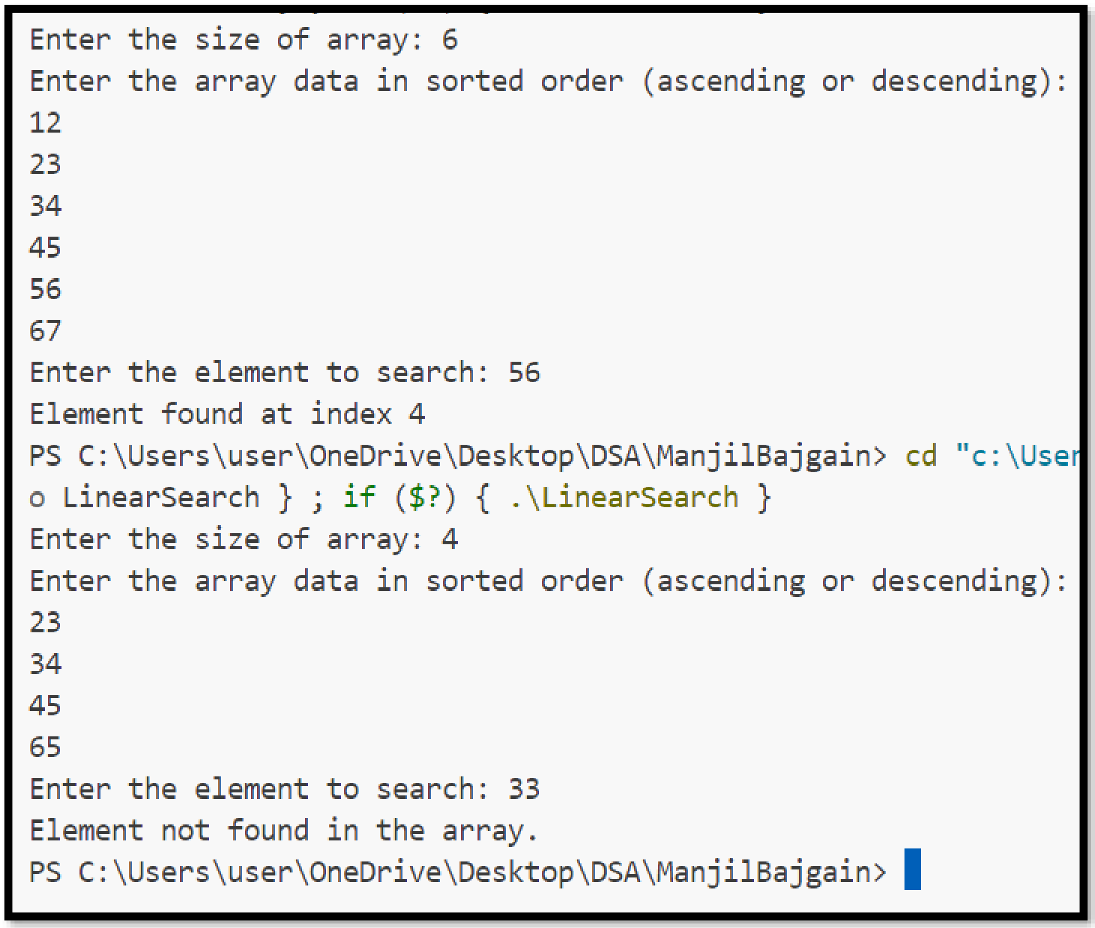
Source code:

```
#include <stdio.h>

void linearSearch(int arr[], int n, int key)
{
    int i, flag = 0;
    for (i = 0; i < n; i++)
    {
        if (arr[i] == key)
        {
            printf("Element found at index %d\n", i);
            flag = 1;
            break;
        }
    }
    if (!flag)
    {
        printf("Element not found in the array.\n");
    }
}
int main()
{
    int n, i, key;
    printf("Enter the size of array: ");
    scanf("%d", &n);
    int arr[n];
    printf("Enter the array data in sorted order (ascending or 
descending):\n"); // Taking input from user
    for (i = 0; i < n; i++)
    {
        scanf("%d", &arr[i]);
    }
    printf("Enter the element to search: ");
    scanf("%d", &key);

    linearSearch(arr, n, key); // Calling Linear Search function
    return 0;
}
```

Output:



```
Enter the size of array: 6
Enter the array data in sorted order (ascending or descending):
12
23
34
45
56
67
Enter the element to search: 56
Element found at index 4
PS C:\Users\user\OneDrive\Desktop\DSA\ManjilBajgain> cd "c:\User
o LinearSearch } ; if ($?) { .\LinearSearch }
Enter the size of array: 4
Enter the array data in sorted order (ascending or descending):
23
34
45
65
Enter the element to search: 33
Element not found in the array.
PS C:\Users\user\OneDrive\Desktop\DSA\ManjilBajgain>
```