Date : 20.01.2025
Lakshana Baskaran
24011103026

LAB - 1
Searching and Sorting

QUESTION 1:

Write a C++ menu-driven program to sort a given array in ascending order. Design proper functions, maintain boundary conditions and follow coding best practices. The menus are as follows:

a. Bubble Sort
b. Selection Sort
c. Insertion Sort
d. Exit

ALGORITHMS:

BUBBLE SORT
Input: A - array of size n
Output: sorted array in ascending order.

```
FOR i = 0 to n-1, do
    FOR j = 1 to n - 1, do
        IF A[j] > A[j + 1], then
            SWAP A[j] and A[j + 1]
RETURN void
```

SELECTION SORT
Input: A - array of size n
Output: sorted array in ascending order.

```
FOR i = 0 to n-2, do
    Min_index ← i
    FOR j = 1 to n - 1, do
        IF A[j] < A[min_index], then
            Min_index ← j
    SWAP A[j] and A[j + 1]
RETURN void
```

Date : 20.01.2025
Lakshana Baskaran
24011103026

INSERTION SORT

Input: A - array of size n

Output: sorted array in ascending order.

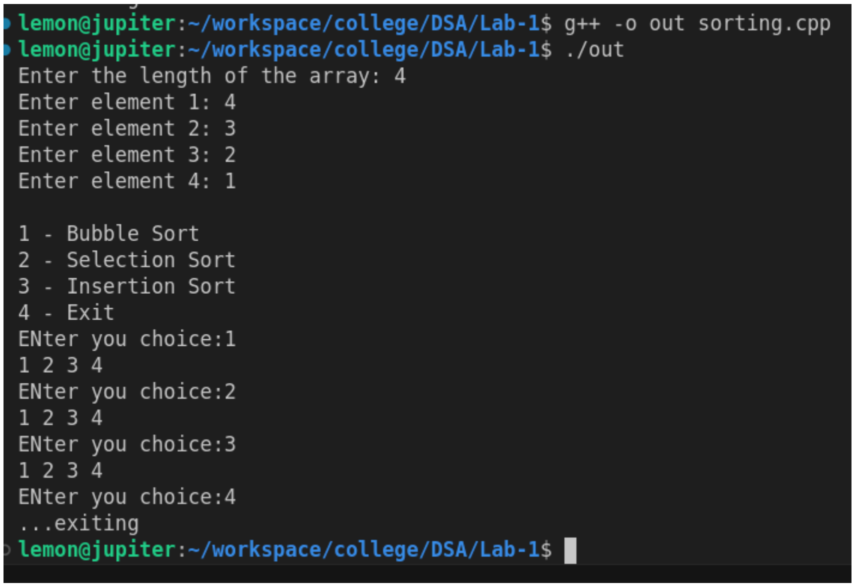
```
FOR i = 1 to n-1, do
    key ← A[i]
    j ← i - 1
    WHILE j  >= 0 and A[i] > key, repeat
        A[j+1] = A[j]
        j ← j - 1
    A[j+1] ← key
RETURN void
```

SOURCE CODE:

```
//Menu Driven program for sorting algorithms
#include <stdio.h>

//swaps the values of two variables
void swap(int* a, int* b) {
    int temp = *a;
    *a = *b;
    *b = temp;
}

//prints the elements of the input array in terminal
void print_array(int arr[], int n) {
    for (int i = 0; i < n; i++) {
            printf("%d ", arr[i]);
    }
    printf("\n");
}

//sorts the input array of length n using the bubble sorting method
void bubble_sort(int arr[], int n) {
    for (int i = 0; i < n; i++) {
        for (int j = 0; j < n; j++) {
            if (arr[i] < arr[j]) {
                swap(arr+i, arr+j);
            }
        }
    }
}

//sorts the input array of length n using the insertion sorting
method
void insertion_sort(int arr[], int n) {
    for (int i = 1; i < n; i++) {
        int key = arr[i];
        int j = i-1;

        while(j >= 0 && arr[j] > key) {
            arr[j+1] = arr[j];
            j = j-1;
        }
        arr[j+1] = key;

    }
}

//sorts the input array of length n using the bubble sorting method
void selection_sort(int arr[], int n) {
    for (int i = 0; i < n-1; i++) {
        int min_index = i;

        for (int j = i+1; j < n; j++) {
            if (arr[j] < arr[min_index]) {
                min_index = j;
            }
        }
        swap(arr+min_index, arr+i);
   }
}


int main() {
    int n = 5;
//    int a[n] = {15,2,13,4,5};
    printf("Enter the length of the array: ");
    scanf("%d", &n);

    int a[n];
    for (int i = 0; i < n; i++) {
        printf("Enter element %d: ", i+1);
        scanf("%d", a+i);
    }
    printf("\n");

    printf("1 - Bubble Sort\n2 - Selection Sort\n3 - Insertion
Sort\n4 - Exit\n");
    int choice;
    while (choice != 4) { //menu
        printf("ENter you choice:");
        scanf("%d", &choice);

        switch(choice) {
            case 1:
                bubble_sort(a, n);
                print_array(a, n);
                break;
            case 2:
                selection_sort(a, n);
                print_array(a, n);
                break;
            case 3:
                insertion_sort(a, n);
                print_array(a, n);
                break;
            case 4:
                printf("...exiting\n");
                break;
            default:
                printf("Invalid Choice\n");
                break;
        }
    }
    return 0;
}
```

Date : 20.01.2025
Lakshana Baskaran
24011103026

OUTPUT:



```
lemon@jupiter:~/workspace/college/DSA/Lab-1$ g++ -o out sorting.cpp
lemon@jupiter:~/workspace/college/DSA/Lab-1$ ./out
Enter the length of the array: 4
Enter element 1: 4
Enter element 2: 3
Enter element 3: 2
Enter element 4: 1

1 - Bubble Sort
2 - Selection Sort
3 - Insertion Sort
4 - Exit
ENter you choice:1
1 2 3 4
ENter you choice:2
1 2 3 4
ENter you choice:3
1 2 3 4
ENter you choice:4
...exiting
lemon@jupiter:~/workspace/college/DSA/Lab-1$ 
```

Date : 20.01.2025
Lakshana Baskaran
24011103026

QUESTION 2:

Convert the sorting program into a header file and include it into a new cpp file. Write a C++ menu-driven program for linear and binary search in this new cpp file. Utilize any of the sorting functions in the included header file to sort the input array before performing a binary search. Design proper functions, maintain boundary conditions and follow coding best practices. The menu-driven program supports:

a. Linear Search
b. Binary Search
c. Exit

ALGORITHMS:

LINEAR SEARCH
Input: A - array of size n, x - element to be searched
Output: True/False

```
FOR i = 0 to n-1, repeat
    IF A[i] = x, then
        RETURN True
RETURN False
```

BINARY SEARCH
Input: A - array of size n, low - index of the first element of A, high - index of the last element of A, x - element to be searched
Output: True/False

```
WHILE low <= high, repeat
    mid ← low + (high-low)/2
    IF A[mid] = x, then
        RETURN true
    IF x > A[mid], then
        low = mid + 1
    IF x < A[mid], then
        high = mid - 1
RETURN false
```

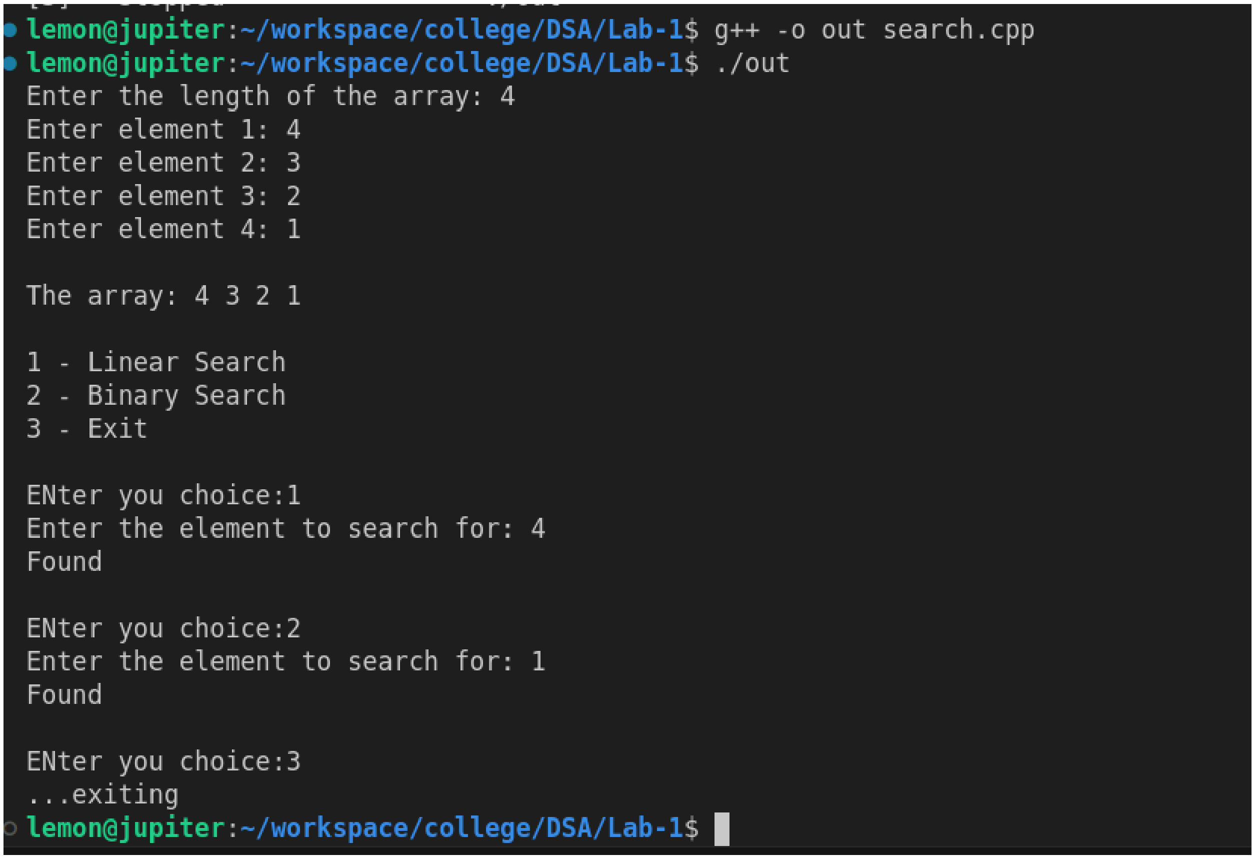
SOURCE CODE:

```
#include <stdio.h>
#include <stdbool.h>
```

Date : 20.01.2025
Lakshana Baskaran
24011103026

```
#include "sort.h"

//searches input array of length n for element x through linear
search method
bool linear_search(int arr[], int n, int x) {
    for (int i = 0; i < n; i++) {
        if (arr[i] == x) {
            return true;
        }
    }
    return false;
}

//searches input array of length n for element x through binary
search method
bool binary_search(int arr[], int low, int high, int x) {

    while (low <= high) {
        int mid = low + (high-low)/2;

        if (arr[mid] == x) {
            return true;
        }

        if (x > arr[mid]) {
            low = mid + 1;
        }

        if (x < arr[mid]) {
            high = mid - 1;
        }

    }
    return false;
}

int main() {
    int n, element;
    printf("Enter the length of the array: ");
    scanf("%d", &n);

    int arr[n];
    for (int i = 0; i < n; i++) {
        printf("Enter element %d: ", i+1);
        scanf("%d", arr+i);
    }
    printf("\nThe array: ");

    print_array(arr, n);


    printf("\n1 - Linear Search\n2 - Binary Search\n3 - Exit\n");
    int choice;
    while (choice != 3) { //menu
        printf("\nENter you choice:");
        scanf("%d", &choice);

        switch(choice) {
            case 1:
                printf("Enter the element to search for: ");
                scanf("%d", &element);

                if (linear_search(arr, n, element)){
                    printf("Found\n");
                }
                else {
                    printf("not found\n");
                }
                break;

            case 2:
                printf("Enter the element to search for: ");
                scanf("%d", &element);

                selection_sort(arr, n);
                if (binary_search(arr, 0, n-1, element)){
                    printf("Found\n");
                }
                else {
                    printf("not found\n");
                }
                break;

            case 3:
                printf("...exiting\n");
                break;
            default:
                printf("Invalid Choice\n");
                break;
        }
    }
    return 0;
}
```

OUTPUT:



```
lemon@jupiter:~/workspace/college/DSA/Lab-1$ g++ -o out search.cpp
lemon@jupiter:~/workspace/college/DSA/Lab-1$ ./out
Enter the length of the array: 4
Enter element 1: 4
Enter element 2: 3
Enter element 3: 2
Enter element 4: 1

The array: 4 3 2 1

1 - Linear Search
2 - Binary Search
3 - Exit

ENter you choice:1
Enter the element to search for: 4
Found

ENter you choice:2
Enter the element to search for: 1
Found

ENter you choice:3
...exiting
lemon@jupiter:~/workspace/college/DSA/Lab-1$ 
```