Student Information

- **Name**: [ATUL KUMAR]
- **Student ID**: [590011190]
- **Branch**: [MCA]
- **Batch**: [B1]

Instructor: [Dr. Sourbh Kumar]

Lab Experiment: 07 **Batch: 1 & 2**
Subject: Data Structures Lab **MCA**
Semester: 1st

Assignment Tasks

Assignment 1st: Linear Search Implementation

Tasks:

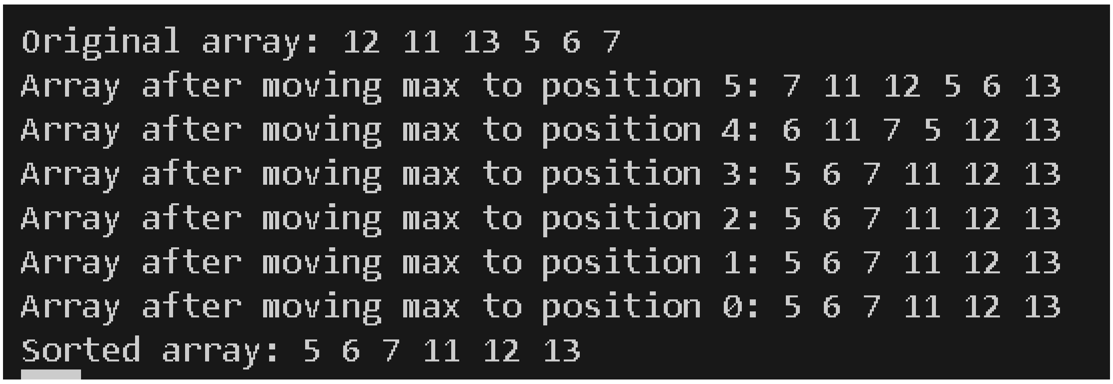
- Write a C program to implement linear search.
- The program should take an array and a target value as inputs and search for the target within the array.
- Display the index where the target value is found, or indicate if it is not present.

Testing: Use an example array of unsorted elements to demonstrate the search process.

Solution:--

```
#include <stdio.h>

// Function to perform linear search
int linearSearch(int arr[], int size, int target) {
for (int i = 0; i < size; i++) {
if (arr[i] == target) {
return i;  // Return the index if the target is found
}
}
return -1;  // Return -1 if the target is not found
}

int main() {
int n, target;

// Take array size as input
printf("Enter the size of the array: ");
scanf("%d", &n);

int arr[n];

// Take array elements as input
printf("Enter %d elements of the array: ", n);
for (int i = 0; i < n; i++) {
scanf("%d", &arr[i]);
}

// Take target value as input
printf("Enter the target value to search for: ");
scanf("%d", &target);

// Perform linear search
int result = linearSearch(arr, n, target);

// Display the result
if (result != -1) {
printf("Target found at index %d.\n", result);
} else {
printf("Target not found in the array.\n");
}

return 0;
}
```

Output:--



```
Original array: 12 11 13 5 6 7
Array after moving max to position 5: 7 11 12 5 6 13
Array after moving max to position 4: 6 11 7 5 12 13
Array after moving max to position 3: 5 6 7 11 12 13
Array after moving max to position 2: 5 6 7 11 12 13
Array after moving max to position 1: 5 6 7 11 12 13
Array after moving max to position 0: 5 6 7 11 12 13
Sorted array: 5 6 7 11 12 13
```

Assignment 2nd: Binary Search Implementation

Tasks:

- Write a C program to implement binary search.
- The program should prompt the user to enter a sorted array and a target value.
- Display the index where the target value is found, or indicate if it is not present.

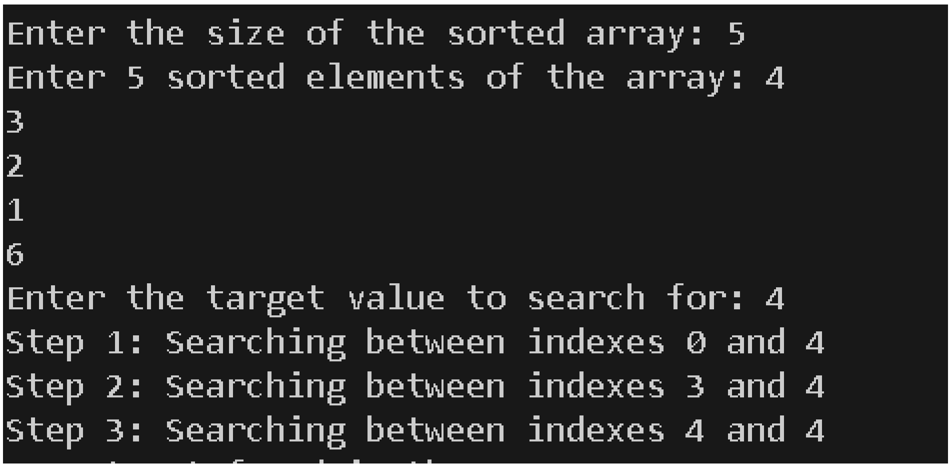
Testing: Use a sorted example array to demonstrate the search process and show each step as the interval is divided.

Solution:--

```
#include <stdio.h>

// Function to perform binary search
int binarySearch(int arr[], int left, int right, int target) {
int step = 1;  // To track each step in the search process

while (left <= right) {
int mid = left + (right - left) / 2;
// Display current step and search interval
printf("Step %d: Searching between indexes %d and %d\n", step++, left, right);

// Check if target is at mid
if (arr[mid] == target) {
return mid;
}
// If target is greater, ignore left half
if (arr[mid] < target) {
left = mid + 1;
}
// If target is smaller, ignore right half
else {
right = mid - 1;
}
}
return -1;  // Target not found
}

int main() {
int n, target;

// Take array size as input
printf("Enter the size of the sorted array: ");
scanf("%d", &n);

int arr[n];

// Take sorted array elements as input
printf("Enter %d sorted elements of the array: ", n);
for (int i = 0; i < n; i++) {
scanf("%d", &arr[i]);
}

// Take target value as input
printf("Enter the target value to search for: ");
scanf("%d", &target);

// Perform binary search
int result = binarySearch(arr, 0, n - 1, target);

// Display the result
if (result != -1) {
printf("Target found at index %d.\n", result);
} else {
printf("Target not found in the array.\n");
}

return 0;
```

```
}
```

Output:--



```
Enter the size of the sorted array: 5
Enter 5 sorted elements of the array: 4
3
2
1
6
Enter the target value to search for: 4
Step 1: Searching between indexes 0 and 4
Step 2: Searching between indexes 3 and 4
Step 3: Searching between indexes 4 and 4
```