Lab - Experiment 7

Objective:

- To understand and implement basic searching algorithms.

- To analyze the differences between linear and binary search in terms of efficiency.

- To understand how the data structure (sorted vs. unsorted) affects search performance.

Assignment 1st: Linear Search Implementation

Tasks:

- Write a C program to implement linear search.

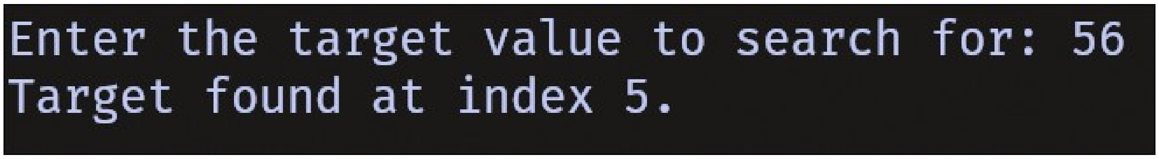
- The program should take an array and a target value as inputs and search for the target within the array.

- 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.

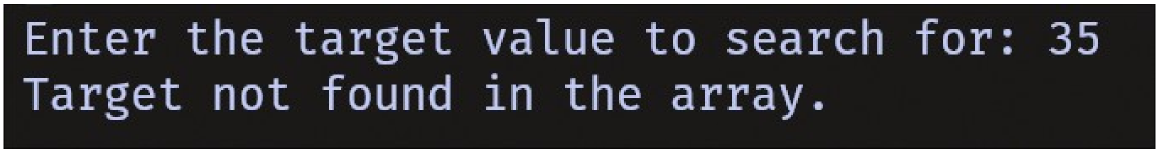
Code:-

```
#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 index if target is found
    }
  }
  return -1; // Return -1 if target is not found
}
int main() {
  int arr[] = {34, 78, 19, 5, 102, 56, 89};
  int size = sizeof(arr) / sizeof(arr[0]);
  int target;
  // Prompt user to enter the target value
  printf("Enter the target value to search for: ");
  scanf("%d", &target);
  // Perform linear search
  int result = linearSearch(arr, size, target);
  // Display results
  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 target value to search for: 56
Target found at index 5.
```



```
Enter the target value to search for: 35
Target not found in the array.
```

Assignment 2nd: Binary Search Implementation

Tasks:

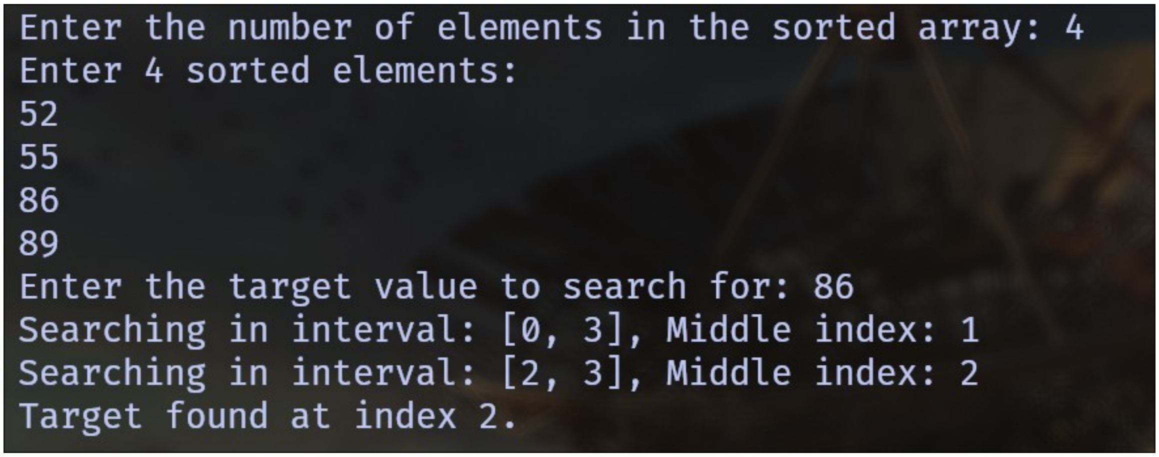
- Write a C program to implement binary search.
- The program should prompt the user to enter a sorted array and a target value.
- Display the index where the target value is found, or indicate if it is not present.

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

Code:-

```
#include <stdio.h>
// Function to perform binary search
int binarySearch(int arr[], int left, int right, int target) {
  while (left <= right) {
    int mid = left + (right - left) / 2; // Find the middle index
    // Display the current interval being checked
    printf("Searching in interval: [%d, %d], Middle index: %d\n", left, right,mid);
    // Check if the target is present at mid
    if (arr[mid] == target) {
      return mid; // Return index if target is found
    }
    // If target is greater, ignore the left half
    else if (arr[mid] < target) {
      left = mid + 1;
    }
    // If target is smaller, ignore the right half
    else {
      right = mid - 1;
    }
  }
  return -1; // Return -1 if target is not found
}
int main() {
  int size, target;
  // Prompt user to enter the array size
  printf("Enter the number of elements in the sorted array: ");
  scanf("%d", &size);
  int arr[size];
  // Prompt user to enter the sorted array elements
  printf("Enter %d sorted elements:\n", size);
  for (int i = 0; i < size; i++) {
    scanf("%d", &arr[i]);
  }
  // Prompt user to enter the target value
  printf("Enter the target value to search for: ");
  scanf("%d", &target);
  // Perform binary search
  int result = binarySearch(arr, 0, size - 1, target);
  // Display results
  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 number of elements in the sorted array: 4
Enter 4 sorted elements:
52
55
86
89
Enter the target value to search for: 86
Searching in interval: [0, 3], Middle index: 1
Searching in interval: [2, 3], Middle index: 2
Target found at index 2.
```