Green University of Bangladesh
Department of Computer Science and Engineering (CSE)
Faculty of Sciences and Engineering
Semester: Spring, Year: 2024, B.Sc. in CSE (Day)

LAB REPORT NO: 01
Course Title: Data Structure Lab
Course Code: CSE 106 **Section:** 223(D5)
Lab Experiment Name: Implement Linear Search Algorithm

Student Details

Name	ID
Promod Chandra Das	231002005

Lab Date : _
Submission Date : _
Course Teacher's Name : Md. Shihab Hossain

[For Teachers use only: Don't Write Anything inside this box]

Lab Report Status

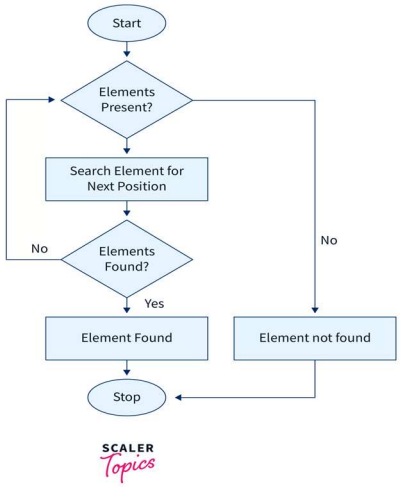
Marks: …… **Signature:**.....
Comments:..... **Date:**.....

❖**Title of The Lab Experiment:** Implement Linear Search Algorithm.

❖**Objectives:-**

- To gather knowledge of different types of searching algorithms.
- To implement linear and binary search algorithms.

❖**Problem analysis:**



❖Algorithm

Algorithm 1: Linear Search
Input: Array A, Value x
/* Linear Search (Array A, Value x) */
1 Step 1: Set i to 1
2 Step 2: if i > n then
3 then go to step 7
4 end
5 Step 3: if A[i] = x then
6 then go to step 6
7 end
8 Step 4: Set i to i + 1
9 Step 5: Go to Step 2
10 Step 6: Print Element x Found at index i and go to step 8
11 Step 7: Print element not found
12 Step 8: Exit

❖Implementation in C

```
1 /* Linear Search code */
2 #include<stdio.h>
3 int main()
4 {
5 int list[20],size,i,sElement;
6
7 printf("Enter size of the list: ");
8 scanf("%d",&size);
9
10 printf("Enter any %d integer values: ",size);
11 for(i = 0; i < size; i++)
12 scanf("%d",&list[i]);
13
14 printf("Enter the element to be Search: ");
15 scanf("%d",&sElement);
16
17 // Linear Search Logic
18 for(i = 0; i < size; i++)
```

```
19 {
20 if(sElement == list[i])
21 {
22 printf("Element is found at %d index", i);
23 break;
24 }
25 }
26 if(i == size)
27 printf("Given element is not found in the list!!!");
28
29 return 0;
30 }
```

❖Input/Output

```
Output of the program is given below.
Enter size of the list: 5
Enter any 5 integer values: 3 1 5 7 4
Enter the element to be Search: 7
Element is found at 3 index
```

- **Title of The Lab Experiment:** Implement Binary Search Algorithm.

❖Problem analysis

Search a sorted array by repeatedly dividing the search interval in half. Begin with an interval covering the
whole array. If the value of the search key is less than the item in the middle of the interval, narrow the interval
to the lower half. Otherwise, narrow it to the upper half. Repeatedly check until the value is found or the
interval is empty. Binary search runs in logarithmic time in the worst case, making O (log n) comparisons,
where n is the number of elements in the array. Binary search is faster than linear search except for small
arrays. However, the array must be sorted first to be able to apply binary search.

❖Algorithm :

Algorithm 2: Binary Search
Input: a, lower_bound, upper_bound, val
/* Binary_Search (a, lower_bound, upper_bound, val) */
1 step 1: set beg = lower_bound end = upper_bound, pos = - 1
2 step 2: repeat steps 3 and 4 while beg<=end
3 step 3: set mid = (beg + end)/2
4 step 4: if a [mid] = val then
5 set pos = mid
6 print pos
7 go to step 6
8 else
9 if a[mid] > val then
10 set end = mid - 1
11 else
12 set beg = mid + 1
13 end
14 end
15 step 5: if pos = -1 then
16 print "value is not present in the array"
17 end
18 step 6: exit

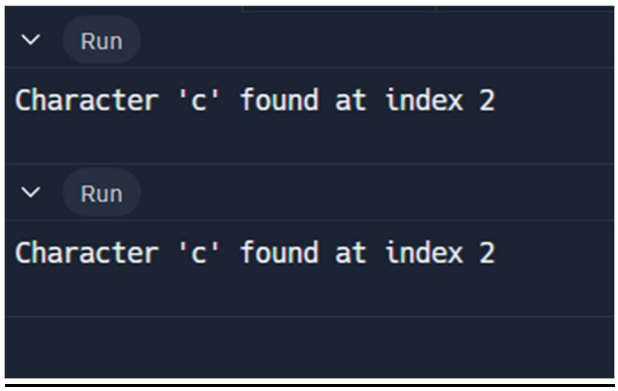
❖Lab Exercise (Submit as a report):

- Implement Linear Search for an array with character data.

main.c

```c
#include <stdio.h>

int linear_search(char arr[], int n, char target) {
  for (int i = 0; i < n; i++) {
    if (arr[i] == target) {
      return i;
    }
  }
  return -1; // Target not found
}

int main() {
  char arr[] = {'a', 'b', 'c', 'd', 'e'};
  int n = sizeof(arr) / sizeof(arr[0]); // Get array size (excluding null terminator)

  char target = 'c';

  int index = linear_search(arr, n, target);
  if (index != -1) {
    printf("Character '%c' found at index %d\n", target, index);
  } else {
    printf("Character '%c' not found\n", target);
  }

  return 0;
}
```

❖Input/Output:



❖ Implement Binary Search for an array with character data.

main.c

```c
#include <stdio.h>

int binary_search(char arr[], int size, char target) {
    int low = 0;
    int high = size - 1;
while (low <= high) {
        int mid = (low + high) / 2;
        if (arr[mid] == target) {
            return mid;
        } else if (arr[mid] < target) {
            low = mid + 1;
        } else {
            high = mid - 1;
        }
    }
 return -1;
}
int main() {
    char arr[] = {'a', 'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j'};
    int size = sizeof(arr) / sizeof(arr[0]);
    char target = 'e';
    int result = binary_search(arr, size, target);

    if (result != -1) {
        printf("Element %c is present at index %d\n", target, result);
    } else {
        printf("Element %c is not present in the array\n", target);
    }
  return 0;
}
```

❖Input/Output:

❖Discussion & Conclusion:

Linear Search and Binary Search in C

This program demonstrates two search algorithms: Linear Search and Binary Search.

Linear Search:

- Iterates through each element in the array, comparing it to the target value.
- Simpler to implement but slower for large datasets.

Binary Search:

- Works on sorted arrays only.
- Divides the search space in half repeatedly until the target is found or eliminated.
- Significantly faster for large sorted datasets.