**TITLE OF LAB: (SEARCHING ALGORITHMS)**

**LAB REPORT NO.04**



**Spring 2022**

**CSE-210L Data Structures and Algorithm Lab**

Submitted by

Name:: **Safi Ullah Khan**

Registration No. **20PWCSE1943**

Class Section: B

“On my honor, as student of University of Engineering and Technology, I have neither given nor received unauthorized assistance on this academic work.”

Student Signature: \_\_\_\_\_\_\_\_\_\_\_\_\_\_

Submitted to:

**Dr. Khurram Shehzad Khattak**

(Friday, July 29th, 2022)

**Department of Computer Systems Engineering**

**University of Engineering and Technology, Peshawar**

**OBJECTIVES OF THE LAB**

‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐

In this lab, we will learn about some basic searching techniques and algorithms.

* Linear Search
* Binary Search

‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐‐

## **Task 01**

Implement Linear Search and analyze its worst, best and average case complexity.

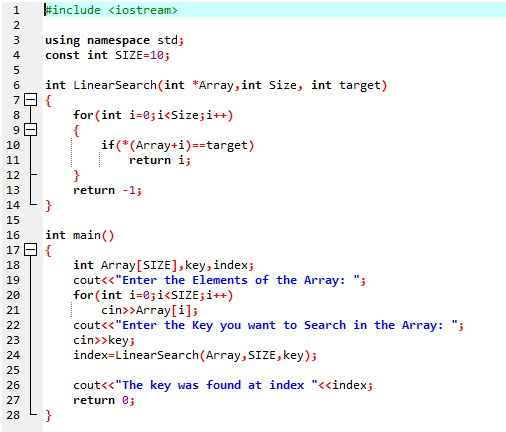
**Pseudo-Code/Explanation:**

* Ask the user to enter size of the Array
* Ask the user to enter elements of the Array
* Ask the user to enter a key 2
* Call the linear search function and pass the array and its size to it.

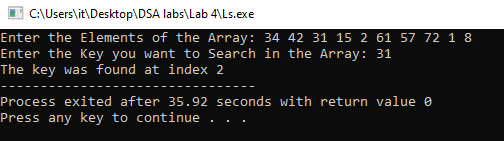
• Take a for loop from 0 to size

• If Arr[i] is equal to key, Return the index value.

**Screenshot of Input:**



**Screenshot of Output:**



**Complexity:**

**Best case:** For linear search algorithm best case complexity is O [1] since in this algorithm the if the key is found at the first index of the array, the loop will only have to transverse one time.

**Worst case:** For linear search algorithm worst case complexity is O [N] since in this algorithm if the key is at any position other than the first index, the loop will have to transverse n times.

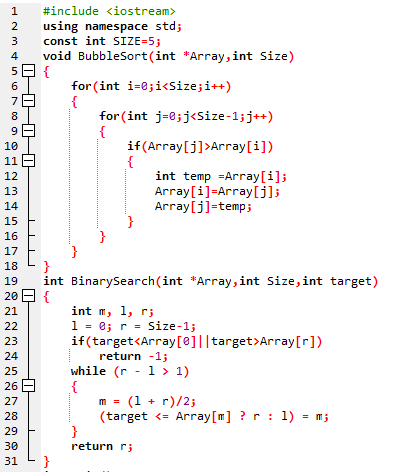
## **Task 02**

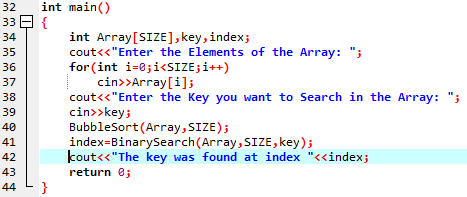
Implement Binary Search and analyze its worst, best and average case complexity.

**Pseudo-Code/Explanation:**

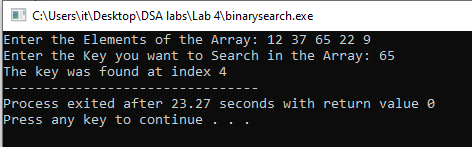
* Ask the user to enter size of the Array
* Ask the user to enter elements of the Array
* Ask the user to enter a key.
* Call the binary search function and pass the array and its size to it.
* Calculate the midpoint of the array.
* While l – r is greater than 1.
* If Arr[m] is greater than or equal to the key, put r equal to m.
* Else if Arr[m] is less than key, put l equal to m.
* Return the r value.

**Screenshot of Input:**





**Screenshot of Output:**



**Complexity:**

**Best case:** For binary search algorithm best case complexity is O [1] since in this algorithm the if the key is found at the middle index of the array, the loop will only have to transverse one time.

**Worst case:** For binary search algorithm worst case complexity is O [log2n] since in this algorithm the if the key is not found at the first middle index of the array, binary search begins comparing the middle element of the sub-array with the key. If the key is less than or greater than the middle element, the search continues in the lower or upper half of the array.