NAME: MIHIR KATAKDHOND
ROLL NO: 231070025
BRANCH: SY BTECH CE
ASSIGNMENT NO. 2

Page No.
Date

Aim: To implement linear and binary search

=> pseudo-code for linear search

```
linearsearch (arr, K)
// searches for a given value in a given array by iterating
// through the array by linear search
// input: An array arr[] and a search key K
// output: The index of the first element in arr that
//          matches K or -1 if no such element exists
for i in arr [i]
   if arr [i] == K  return i
     else return -1
```

=> pseudo-code for recursive binary search

```
binarysearch (arr, K, high, low)
// searches for a given value in a given array by binary
// search
// input: An array arr[], a search key K and three
//          datatypes of high, low, mid where initially
//          high stores index of last element, low stores index
//          of starting element and mid stores index of middle element
// output: The index of the element in arr that matches K
//          or -1 if no such element is found
data_type low = 0
data_type high = arr.size() -1
data_type mid = low + (high - low)/2
if high >= low  return  mid = low + (high - low)/2
  if arr [mid] == K  return mid
    if arr [mid] > K  return binarysearch(arr, K, mid-1, low)
      else return binarysearch (arr, K, high, mid+1)
        return -1
```

Page No.
Date

⇒ Test cases

1] Test case 1

int arr [5] = { 1, 2, 3, 4, 5 }
int Key = 2

linearsearch ⇒ expected output: index = 1
binarysearch ⇒ expected output: index = 1

2] Test case 2

int arr [3] = { 10, 20, 30 }
int Key = 30

linearsearch ⇒ expected output: index = 2
binarysearch ⇒ expected output: index = 2

3] Test case 3

int arr [ ] = { }
int Key = 10

linearsearch ⇒ expected output: since array is empty Key value cannot be found

binary search ⇒ expected output: since array is empty Key value cannot be found

4] Test case 4

int arr [4] = {12, 13, 14, 15}
int Key = 18

linearsearch => expected output: The given Key does not exist in the given array heme cant be located

binarysearch => expected output: The given Key does not exist in the given array heme cant be located

5] Test case 5

int arr [4] = {12, 13, 14, 15}
int Key = mihir

linear search => expected output: you have inputted a string and array is of integer heme invalid datatype has been submitted

int arr [4] = {10, 5, 2, 7}
int Key = 2

binarysearch => expected output: ~~You have~~ The array is unsorted. Binary search requires a sorted array

→ Time complexity of linear search

best case: The key value is located at very first position in the array
Key value is found itself at first iteration so no of operations is constant i.e $\underline{O(1)}$

average case: Key value is located in the middle of the array
The algorithm will have to check half the elements from array so if array has n elements then will take approx n/2 comparison
time complexity = $\underline{O(n/2)}$

worst case: Key value is located at end of the array
The loop will have to check all the elements from array so if array has n elements then the time complexity will be $\underline{O(n)}$

→ Time complexity of binary search

$T(1) = O(1)$ { when array is of size 1 }

$T(n) = T\left(\frac{n}{2}\right) + O(1)$

$= \left(T\left(\frac{n}{4}\right) + O(1)\right) + O(1)$

$= \left(T\left(\frac{n}{8}\right) + O(1)\right) + 2O(1)$

$= T\left(\frac{n}{8}\right) + 3O(1)$

$\vdots$

$= T\left(\frac{n}{2^k}\right) + k \cdot O(1)$

The recursion continues until size of array is reduced to 1, this happens when:

$$\frac{n}{2^k} = 1$$

$$n = 2^k$$

$$k = \log_2(n)$$

Substituting $k = \log_2(n)$ into recurrence relation

$$T(n) = T(1) + \log_2(n) \cdot O(1)$$

but: $T(1) = O(1)$

$$T(n) = O(1) + \log_2(n) \cdot O(1)$$
$$= O(\log n)$$

Conclusion:

In conclusion, I have learned how to design and implement algorithms to find the index of the key element using both linear and binary search. The code generates appropriate and accurate results by handling a comprehensive set of test cases, which includes both positive and negative scenarios.