Name: Arpita Dinesh Singh
2nd-year B.tech
Computer engineering

Roll No:231071005

DESIGN ANALYSIS OF ALGORITHM

LINEAR SEARCH AND BINARY SEARCH

Theory :

LINEAR SEARCH:

Linear search is a straightforward algorithm for finding a target value in a list. It works by sequentially checking each element of the list until the target value is found or the end of the list is reached.

BINARY SEARCH:

Binary search is an efficient algorithm for finding a target value in a sorted list. It repeatedly divides the search interval in half, comparing the target value to the middle element of the interval, until the target is found or the interval is empty.

CODE with proper coding style:

FOR LINEAR SEARCH:

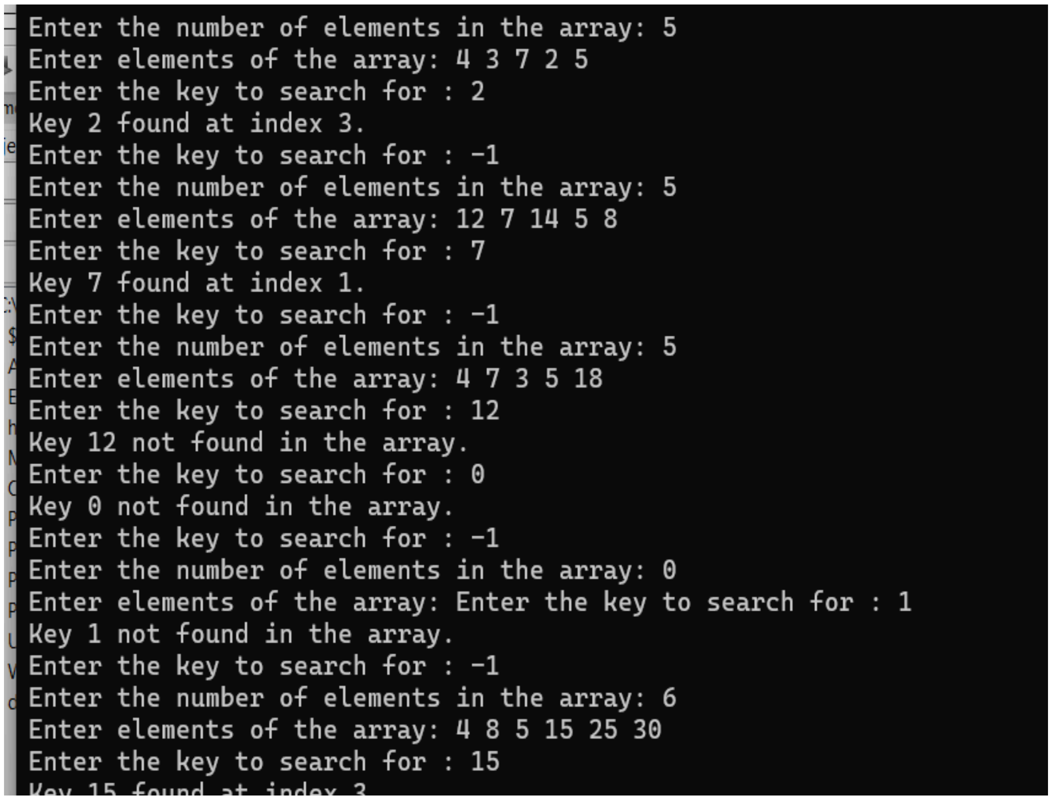
```
#include <iostream>
using namespace std;

int linearSearch(const int arr[], int size, int
key) {
    for (int i = 0; i < size; ++i) {
        if (arr[i] == key) {
            return i;
        }
    }
    return -1;
}
int main() {
    while (true) {
        int size;
        cout << "Enter the number of elements in
the array (or -1 to exit): ";
        cin >> size;

        if (size == -1) {
            break;
        }
        int* array = new int[size];

        cout << "Enter elements of the array: ";
        for (int i = 0; i < size; ++i) {
            cin >> array[i];
        }
        while (true) {
           int key;
            cout << "Enter the key to search for
(or -1 to enter a new array): ";
            cin >> key;

            if (key == -1) {
                delete[] array;
                break;
            }
            int index = linearSearch(array,
size, key);

            if (index != -1) {
                cout << "Key " << key << " found
at index " << index << "." << endl;
            } else {
                cout << "Key " << key << " not
found in the array." << endl;
            }
        }
    }
   cout << "Exiting program." << endl;
    return 0;
}
```

FOR BINARY SEARCH:

```
#include <iostream>
#include <algorithm>
using namespace std;

int binarySearch(int arr[], int size, int key) {
    int low = 0;
    int high = size - 1;

    while (low <= high) {
        int mid = low + (high - low) / 2;

        if (arr[mid] == key) {
            return mid;
        } else if (arr[mid] < key) {
            low = mid + 1;
        } else {
            high = mid - 1;
        }
    }
    return -1;
}
```

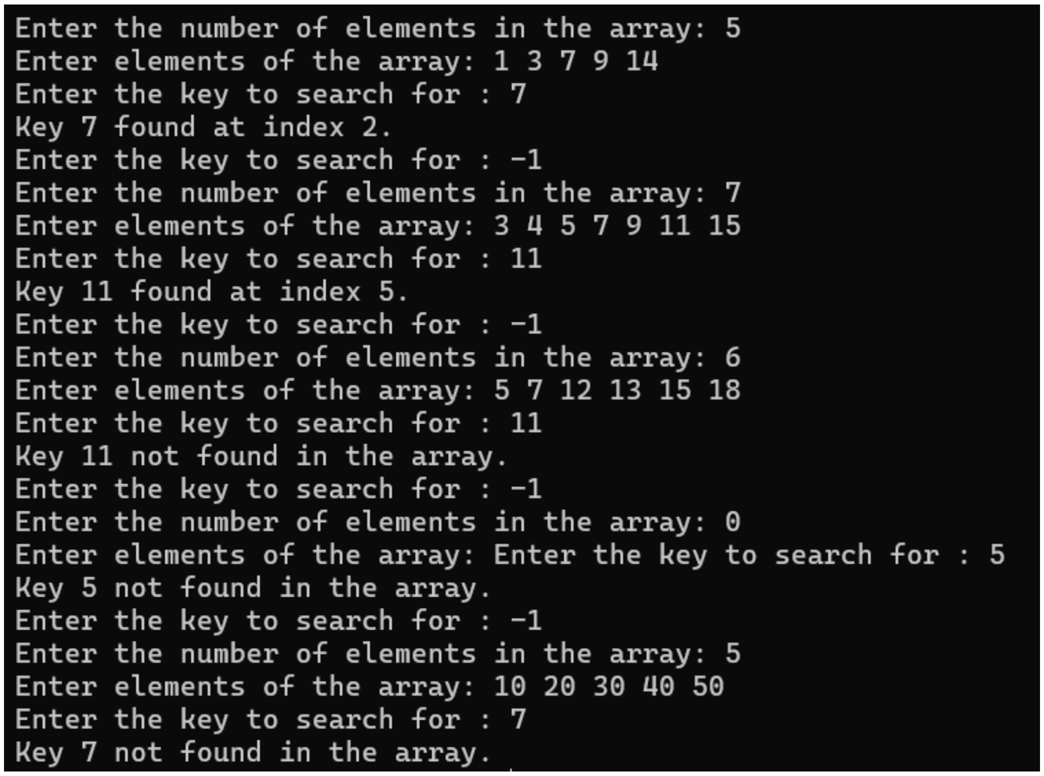
```
int main() {
    while (true) {
        int size;
        cout << "Enter the number of elements in
the array (or -1 to exit): ";
        cin >> size;

        if (size == -1) {
            break;
        }
        int* array = new int[size];
        cout << "Enter elements of the array: ";
        for (int i = 0; i < size; ++i) {
            cin >> array[i];
        }
        sort(array, array + size);

        while (true) {
            int key;
            cout << "Enter the key to search for
(or -1 to enter a new array): ";
            cin >> key;

            if (key == -1) {
                delete[] array;
                break;
            }
            int index = binarySearch(array,
size, key);
            if (index != -1) {
                cout << "Key " << key << " found
at index " << index << "." << endl;
            } else {
                cout << "Key " << key << " not
found in the array." << endl;
            }
        }
    }
    cout << "Exiting program." << endl;
    return 0;
}
```

TEST CASES FOR LINEAR SEARCH:



```
Enter the number of elements in the array: 5
Enter elements of the array: 4 3 7 2 5
Enter the key to search for : 2
Key 2 found at index 3.
Enter the key to search for : -1
Enter the number of elements in the array: 5
Enter elements of the array: 12 7 14 5 8
Enter the key to search for : 7
Key 7 found at index 1.
Enter the key to search for : -1
Enter the number of elements in the array: 5
Enter elements of the array: 4 7 3 5 18
Enter the key to search for : 12
Key 12 not found in the array.
Enter the key to search for : 0
Key 0 not found in the array.
Enter the key to search for : -1
Enter the number of elements in the array: 0
Enter elements of the array: Enter the key to search for : 1
Key 1 not found in the array.
Enter the key to search for : -1
Enter the number of elements in the array: 6
Enter elements of the array: 4 8 5 15 25 30
Enter the key to search for : 15
Key 15 found at index 3
```

TEST CASES FOR BINARY SEARCH:



```
Enter the number of elements in the array: 5
Enter elements of the array: 1 3 7 9 14
Enter the key to search for : 7
Key 7 found at index 2.
Enter the key to search for : -1
Enter the number of elements in the array: 7
Enter elements of the array: 3 4 5 7 9 11 15
Enter the key to search for : 11
Key 11 found at index 5.
Enter the key to search for : -1
Enter the number of elements in the array: 6
Enter elements of the array: 5 7 12 13 15 18
Enter the key to search for : 11
Key 11 not found in the array.
Enter the key to search for : -1
Enter the number of elements in the array: 0
Enter elements of the array: Enter the key to search for : 5
Key 5 not found in the array.
Enter the key to search for : -1
Enter the number of elements in the array: 5
Enter elements of the array: 10 20 30 40 50
Enter the key to search for : 7
Key 7 not found in the array.
```

CONCLUSION:

We learnt about the following:

- **Linear Search:** Sequentially checks each element in a list until the key is found or the list ends. Time Complexity: O(n).

- **Binary Search:**Searches a sorted list by repeatedly dividing the search interval in half. Time Complexity: O(log n).

- **Correct Coding Style:** Use consistent naming, formatting,indentation and simplicity to improve code readability and maintainability.

- **Calculating Time Complexity:** Measures how the runtime of an algorithm grows with input size, expressed in Big O notation.