EXPERIMENT NO:-01

Aim Of Experiment:- Write a program to implement the linear search and binary search.

A. Linear Search

```
import java.util.Scanner;

public class LinearSearch {

    public static int linearSearch(int[] array, int key)
{
        for (int i = 0; i < array.length; i++) {
            if (array[i] == key) {
                return i;
            }
        }
        return -1;
    }

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);

        int[] array = {10, 20, 30, 40, 50};

        System.out.print("Enter the number to search: ");
        int key = scanner.nextInt();

        int result = linearSearch(array, key);

        if (result != -1) {
            System.out.println("Element found at index: "
+ result);
        } else {
            System.out.println("Element not found in the
array.");
        }

        scanner.close();
    }
}
```

OUTPUT:-

```
"C:\Program Files\Java\jdk-22\bin\java.exe" "-javaagent:C:\Users\Ankit raj\IntelliJ IDEA Community Edition 2023.2.4
Enter the number to search: 40
Element found at index: 3
```

BEST CASE TIME COMPLEXITY :- O(1).

WORST CASE TIME COMPLEXITY :- O(n).

B. Binary Search

```
import java.util.Scanner;

public class BinarySearch {
    public static void main(String[] args) {
        int[] arr = {2, 4, 6, 8, 10, 12, 14, 16};
        Scanner scanner = new Scanner(System.in);

        System.out.println("Array:");
        printArray(arr);

        System.out.print("Enter the number to search: ");
        int target = scanner.nextInt();

        int result = binarySearch(arr, target);

        if (result == -1) {
            System.out.println("Element not found in the
array.");
        } else {
            System.out.println("Element found at index: "
+ result);
        }

        scanner.close();
    }

    public static int binarySearch(int[] arr, int target)
{
        int left = 0, right = arr.length - 1;

        while (left <= right) {
            int mid = left + (right - left) / 2;

            if (arr[mid] == target) {
                return mid;
            }

            if (arr[mid] < target) {
                left = mid + 1;
            }

            else {
                right = mid - 1;
            }
        }

        return -1;
    }
```

```
    public static void printArray(int[] arr) {
        for (int value : arr) {
            System.out.print(value + " ");
        }
        System.out.println();
    }
}
```

OUTPUT:-

```
"C:\Program Files\Java\jdk-22\bin\java.exe" "-javaagent:C:\Users\Ankit raj\IntelliJ IDEA Community Edition 2023.2.4
Array:
2 4 6 8 10 12 14 16
Enter the number to search: 14
Element found at index: 6
```

BEST CASE TIME COMPLEXITY :- O(1).

WORST CASE TIME COMPLEXITY :- O(log n).

EXPERIMENT NO:-02

Aim Of Experiment:- Write a program to implement the insertion sort.

```
public class InsertionSort {
   public static void main(String[] args) {
      int[] arr = {2,13,5,18,14};

      System.out.println("Original array:");
      printArray(arr);

      insertionSort(arr);

      System.out.println("Sorted array:");
      printArray(arr);
   }

   public static void insertionSort(int[] arr) {
      int n = arr.length;
      for (int i = 1; i < n; i++) {
         int key = arr[i];
         int j = i - 1;


         while (j >= 0 && arr[j] > key) {
            arr[j + 1] = arr[j];
            j = j - 1;
         }
         arr[j + 1] = key;
      }
   }

   public static void printArray(int[] arr) {
      for (int value : arr) {
         System.out.print(value + " ");
      }
      System.out.println();
   }
}
```

OUTPUT:-

```
"C:\Program Files\Java\jdk-22\bin\java.exe" "-javaagent:C:\Users\Ankit raj\IntelliJ IDEA Community Edition 2023.2.4
Original array:
2 13 5 18 14
Sorted array:
2 5 13 14 18
```

OUTPUT:-

Best Case Time Complexity: O(n).

Worst Case Time Complexity: O(n2).