LAB 6

Name : AZHAR ALI

CMS : 023-23-0314

SEC : A

1. **Implement selection sort and insertion sort.**

```java
import java.lang.Math;
public class Sorting {

    void SELECTION_SORTING(int arr[]) {
        for (int i = 0; i < arr.length - 1; i++) {
            int minIndex = i;
            for (int j = i + 1; j < arr.length; j++) {
                if (arr[j] < arr[minIndex]) {
                    minIndex = j;
                }
            }
            int temp = arr[i];
            arr[i] = arr[minIndex];
            arr[minIndex] = temp;
        }
    }

    void INSERTION_SORT(int arr[]) {
        for (int i = 1; i < arr.length; 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;
        }
    }



    void print(int arr[]){
        for (int i=0;i<arr.length;i++){
            System.out.print(arr[i]+" ");
        }
    }

}
    class Main{
        public static void main(String[] args) {
            Sorting s1 = new Sorting();
            int array[]={5,1,2,6,3,7,4,8};
            int array2[]={5,1,2,6,3,7,4,8};
            System.out.println("Original Array");
            s1.print(array);
            System.out.println("\nOriginal Array After Apply Selection Sort.");
            s1.SELECTION_SORTING(array);
            s1.print(array);
            System.out.println("\n\nOriginal Array");
            s1.print(array2);
            System.out.println("\nOriginal Array After Apply Insertion Sort.");
            s1.INSERTION_SORT(array2);
            s1.print(array);
            System.out.println();



        }}
```

```
Original Array
5 1 2 6 3 7 4 8
Original Array After Apply Selection Sort.
1 2 3 4 5 6 7 8

Original Array
5 1 2 6 3 7 4 8
Original Array After Apply Insertion Sort.
1 2 3 4 5 6 7 8
azharali@fedora:~/Semester 3/DSA LAB/LAB_6$
```

2. **(Solve in NlogN): We are given an array that contains N numbers. We want to determine if there are two numbers whose sum equals a given number K. For instance, if the input is 8, 4, 1, and 6, and K is 10, then the answer is yes (4 plus 6 is 10). A number n may appear more than once in the input array; in that case and only in that case the sum may have the form n + n.**

   Implement a function TwoSum() to solve this problem in O(N log N) time.
   Hint: Sort the items first!

```java
import java.util.Scanner;
import java.util.Arrays;
class FindingSum
{
    static void sumFinder(int arr[],int result)
    {
        Arrays.sort(arr);
        int left=0;
        int right=arr.length-1;
        while(left < right)
        {
            int currentSum=arr[left]+arr[right];
            if(currentSum==result)
            {
                System.out.println("I found two numbers "+arr[left]+" and "+arr[right]+" whose Sum equals to: "+result);
                return;
            }
            else if(currentSum < result)
                left++;
            else
                right--;
        }
        System.out.println("Did not found the numbers whose sum is equal to: "+result);
    }

    public static void main(String[] args)
    {
        Scanner in=new Scanner(System.in);
        int arr[]={6,5,4,3,2,1};
        System.out.print("Enter the number you want to Find: ");
        int sum=in.nextInt();
        sumFinder(arr,sum);
        in.close();
    }
}
```