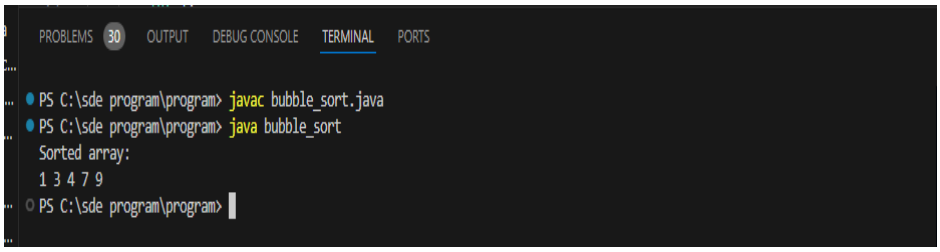
**Q 1) Bubble Sort**

```
import java.io.*;

class bubble_sort {

   static void bubbleSort(int arr[], int n){

      int i, j, temp;

      boolean swapped;

      for (i = 0; i < n - 1; i++) {

         swapped = false;

         for (j = 0; j < n - i - 1; j++) {

            if (arr[j] > arr[j + 1]) {

               temp = arr[j];

               arr[j] = arr[j + 1];

               arr[j + 1] = temp;

               swapped = true;

            }

         }

         if (swapped == false)

            break;

      }

   }

   static void printArray(int arr[], int size){

      int i;

      for (i = 0; i < size; i++)

         System.out.print(arr[i] + " ");

      System.out.println();

   }

   public static void main(String args[]){
```

```
        int arr[] = { 4, 1, 3, 9, 7 };

        int n = arr.length;

        bubbleSort(arr, n);

        System.out.println("Sorted array: ");

        printArray(arr, n);

    }

}
```

**Time Complexity**: O(n2)

**Output:**




**Q 2) Quick Sort**

```
import java.util.Arrays;

class quick_sort {

    static int partition(int[] arr, int low, int high) {

        int pivot = arr[high];

        int i = low - 1;

        for (int j = low; j <= high - 1; j++) {

            if (arr[j] < pivot) {

                i++;

                swap(arr, i, j);

            }

        }

        swap(arr, i + 1, high);

        return i + 1;
```

```
    }
    static void swap(int[] arr, int i, int j) {
        int temp = arr[i];
        arr[i] = arr[j];
        arr[j] = temp;
    }
    static void quickSort(int[] arr, int low, int high) {
        if (low < high) {
            int pi = partition(arr, low, high);
            quickSort(arr, low, pi - 1);
            quickSort(arr, pi + 1, high);
        }
    }
    public static void main(String[] args) {
        int[] arr = {10, 7, 8, 9, 1, 5};
        int n = arr.length;
        quickSort(arr, 0, n - 1);
        for (int val : arr) {
            System.out.print(val + " ");
        }
    }
}
```

**Output:**

```
PS C:\sde program\program> javac quick_sort.java
PS C:\sde program\program> java quick_sort
1 5 7 8 9 10
PS C:\sde program\program>
```

**Q 3) Non Repeating Character**

```
class non_repeating_element {

   static char nonRepeatingChar(String s) {

      int n = s.length();

      for (int i = 0; i < n; ++i) {

         boolean found = false;

         for (int j = 0; j < n; ++j) {

            if (i != j && s.charAt(i) == s.charAt(j)) {

               found = true;

               break;

            }

         }

         if (found == false)

            return s.charAt(i);

      }

      return '$';

   }
```

**Time Complexity:** O(n)

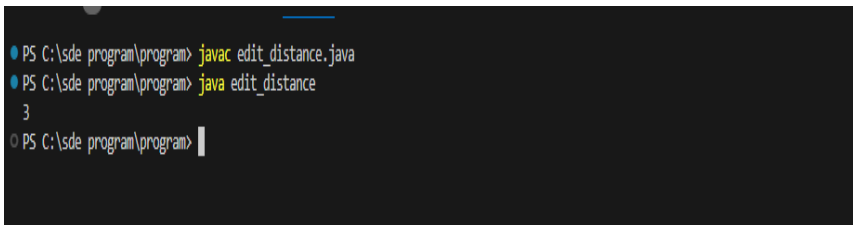
**Output:**


**Q 4) Edit Distance**

```
public class edit_distance {

   public static int edit_distance(String s1,String s2,int m, int n){

      if(m==0) return n;
```

```
        if(n==0) return m;

        if(s1.charAt(m-1)==s2.charAt(n-1))

            return edit_distance(s1, s2, m-1, n-1);

        return 1+Math.min(Math.min(edit_distance(s1, s2, m, n-1),edit_distance(s1, s2, m-1,
n)),edit_distance(s1, s2, m-1, n-1));

    }

    public static int edit_distance(String s1,String s2){

        return edit_distance(s1, s2,s1.length(),s2.length());



    }

    public static void main(String[] args){

        String s1="SUNDAY";

        String s2="SATURDAY";

        System.out.println(edit_distance(s1, s2));

    }

}
```

**Output:**



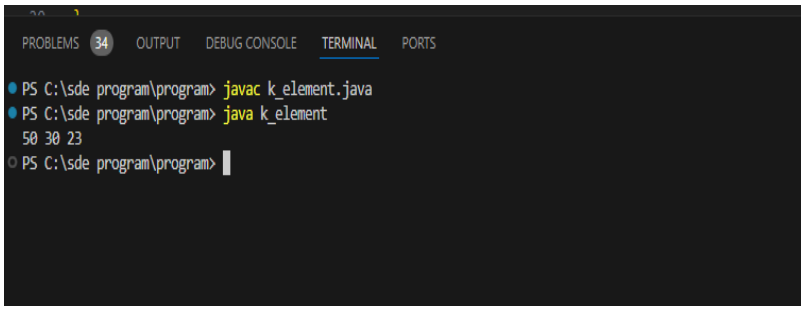

Q 5) K Largest elements

```
import java.util.*;

class k_element {

    static ArrayList<Integer> kLargest(int[] arr, int k) {

        int n = arr.length;

        Integer[] arrInteger =

            Arrays.stream(arr).boxed().toArray(Integer[]::new);
```

```
        Arrays.sort(arrInteger, Collections.reverseOrder());
        ArrayList<Integer> res = new ArrayList<>();
        for (int i = 0; i < k; i++)
            res.add(arrInteger[i]);
        return res;
    }
    public static void main(String[] args) {
        int[] arr = {1, 23, 12, 9, 30, 2, 50};
        int k = 3;
        ArrayList<Integer> res = kLargest(arr, k);
        for (int ele : res)
            System.out.print(ele + " ");
    }
}
```

**Time Complexity:** O(n2)

**Output:**




**Q 6) Form the largest number**

```
class Solution {
    public String largestNumber(int[] nums) {
        String[] numStrs = new String[nums.length];
        for (int i = 0; i < nums.length; i++) {
            numStrs[i] = String.valueOf(nums[i]);
        }
```

```
        Arrays.sort(numStrs, (a, b) -> (b + a).compareTo(a + b));
        if (numStrs[0].equals("0")) {
            return "0";
        }
        StringBuilder result = new StringBuilder();
        for (String numStr : numStrs) {
            result.append(numStr);
        }
        return result.toString();
    }
}
```

**Time Complexity:** O(n)

**Output:**