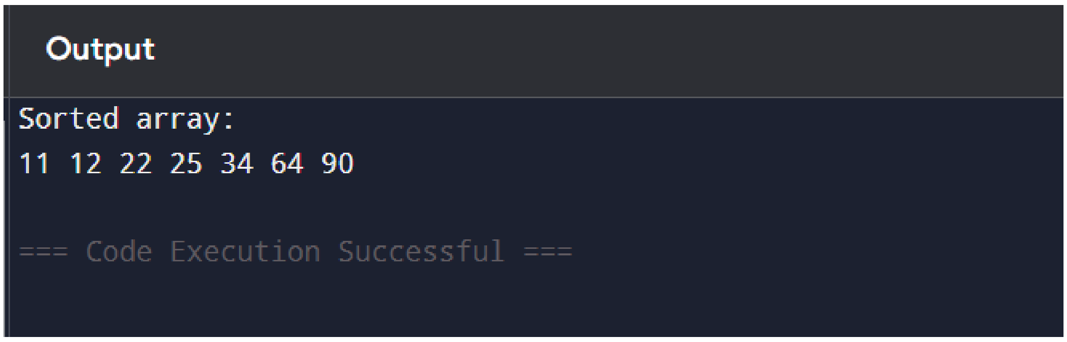
1)BUBBLE SORT

```
import java.io.*;
class bubblesort{
   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]) {

               // Swap arr[j] and 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[] = { 64, 34, 25, 12, 22, 11, 90 };
      int n = arr.length;
```

```
        bubbleSort(arr, n);

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

        printArray(arr, n);

    }

}
```

Time Complexity: O(n^2)
Auxiliary Space: O(1)

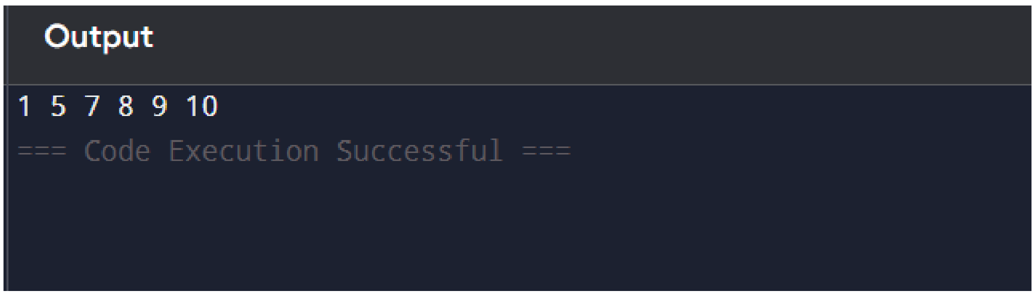


Output

```
Sorted array:
11 12 22 25 34 64 90

=== Code Execution Successful ===
```

2)QUICK SORT

```
import java.util.Arrays;

class quicksort{

    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

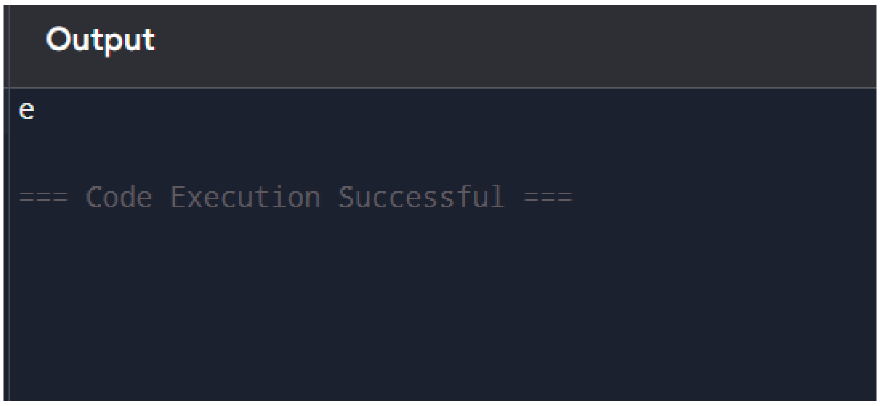
```
1 5 7 8 9 10
=== Code Execution Successful ===
```

3)Find first non-repeating character of given string

```
class nonrepeatingchar {
    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 '$';
    }
    public static void main(String[] args) {
        String s = "racecar";

        System.out.println(nonRepeatingChar(s));
    }
}
```

Time Complexity: O(n^2)

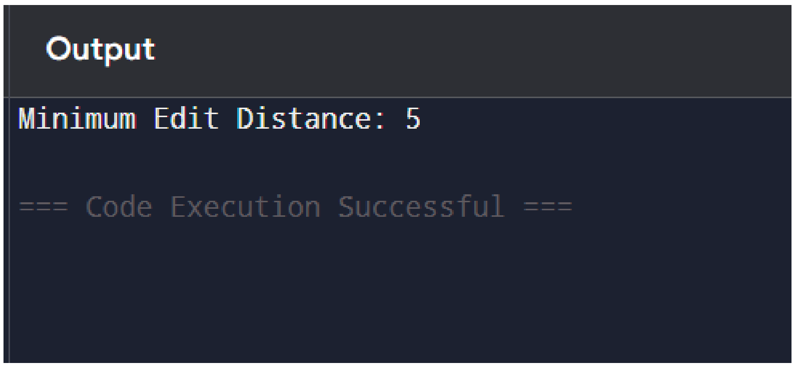
Auxiliary Space: O(1)



```
e

=== Code Execution Successful ===
```

4)EDIT DISTANCE

```
public class EditDistance {
    private static int minDisRec(String s1, String s2, int m, int n, int[][] memo) {
        if (m == 0) return n;
        if (n == 0) return m;
        if (memo[m][n] != -1) return memo[m][n];
        if (s1.charAt(m - 1) == s2.charAt(n - 1)) {
            memo[m][n] = minDisRec(s1, s2, m - 1, n - 1, memo);
        }
        else {
            int insert = minDisRec(s1, s2, m, n - 1, memo);
            int remove = minDisRec(s1, s2, m - 1, n, memo);
            int replace = minDisRec(s1, s2, m - 1, n - 1, memo);
            memo[m][n] = 1 + Math.min(insert, Math.min(remove, replace));
        }
        return memo[m][n];
    }
    public static int minDis(String s1, String s2) {
        int m = s1.length(), n = s2.length();
        int[][] memo = new int[m + 1][n + 1];
        for (int i = 0; i <= m; i++) {
            for (int j = 0; j <= n; j++) {
                memo[i][j] = -1;
            }
        }
        return minDisRec(s1, s2, m, n, memo);
    }
    public static void main(String[] args) {
        String s1 = "intention";
        String s2 = "execution";
        System.out.println("Minimum Edit Distance: " + minDis(s1, s2));
    }
}
```



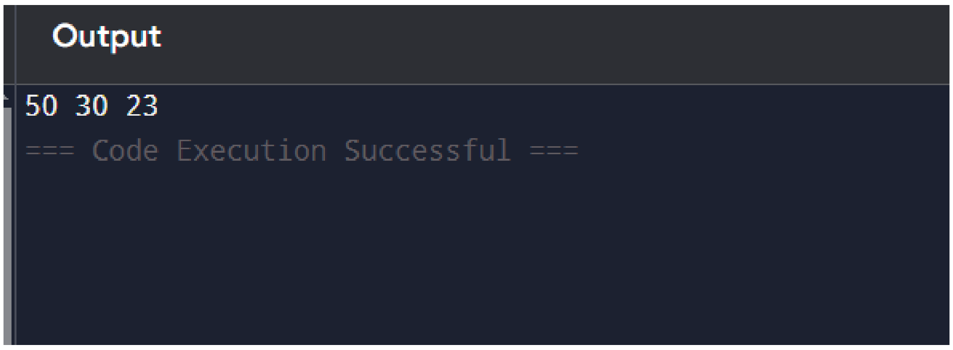
```
Output

Minimum Edit Distance: 5

=== Code Execution Successful ===
```

Time Complexity: O(m x n)
Auxiliary Space: O(m x n)

5)Find k largest elements in an array

```
import java.util.*;
class klarge{
    static int partition(int[] arr, int left, int right) {
        int pivot = arr[right];
        int i = left;

        for (int j = left; j < right; j++) {
            if (arr[j] >= pivot) {
                int temp = arr[i];
                arr[i] = arr[j];
                arr[j] = temp;
                i++;
            }
        }
        int temp = arr[i];
        arr[i] = arr[right];
        arr[right] = temp;
        return i;
    }
    static void quickSelect(int[] arr, int left, int right, int k) {
        if (left <= right) {
            int pivotIdx = partition(arr, left, right);
            int leftCnt = pivotIdx - left + 1;
            if (leftCnt == k)
                return;
            if (leftCnt > k)
                quickSelect(arr, left, pivotIdx - 1, k);
            else
                quickSelect(arr, pivotIdx + 1, right, k - leftCnt);
        }
    }
    static ArrayList<Integer> kLargest(int[] arr, int k) {
        quickSelect(arr, 0, arr.length - 1, k);
        ArrayList<Integer> res = new ArrayList<>();
          for(int i = 0; i < k; i++)
              res.add(arr[i]);
          Collections.sort(res, Collections.reverseOrder());
        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(n^2)
Auxiliary Space: O(n)



Output

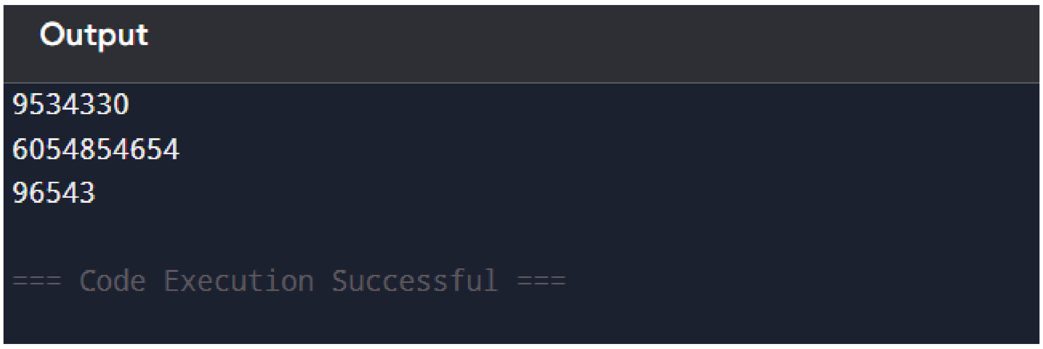
```
50 30 23
=== Code Execution Successful ===
```

6)FORM THE LARGEST NUMBER:

```
import java.util.*;

public class LargestNumber {

    public static String largestNumber(int[] arr) {

        String[] strArr = new String[arr.length];

        for (int i = 0; i < arr.length; i++) {

            strArr[i] = String.valueOf(arr[i]);

        }

        Arrays.sort(strArr, (a, b) -> (b + a).compareTo(a + b));

        if (strArr[0].equals("0")) {

            return "0";

        }

        StringBuilder result = new StringBuilder();

        for (String num : strArr) {

            result.append(num);

        }


        return result.toString();

    }


    public static void main(String[] args) {

        int[] arr1 = {3, 30, 34, 5, 9};
        int[] arr2 = {54, 546, 548, 60};

        int[] arr3 = {3, 4, 6, 5, 9};


        System.out.println(largestNumber(arr1));

        System.out.println(largestNumber(arr2));

        System.out.println(largestNumber(arr3));

    }

}
```

TIME COMPLEXITY:O(k.nlogn)

SPACE COMPLEXITY:O(kn)



Output

```
9534330
6054854654
96543

=== Code Execution Successful ===
```