PRACTICE SET – Day 5 **18-11-2024**

DSA

1.BUBBLE SORT

```
import java.util.Scanner;

class bubbleSort {
    public static void bubbleSort(int arr[]) {
        int n = arr.length;
        for (int i = 0; i < n - 1; i++) {
            boolean swapped = false;
            for (int j = 0; j < n - i - 1; j++) {
                if (arr[j] > arr[j + 1]) {
                    int temp = arr[j];
                    arr[j] = arr[j + 1];
                    arr[j + 1] = temp;
                    swapped = true;
                }
            }
            if (!swapped) break;
        }
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the number of elements: ");
        int n = sc.nextInt();
        int[] arr = new int[n];
```

```
        System.out.println("Enter the elements:");
        for (int i = 0; i < n; i++) {
            arr[i] = sc.nextInt();
        }

        bubbleSort(arr);

        System.out.println("Sorted array: " + java.util.Arrays.toString(arr));
    }
}
```

```
Enter the number of elements: 6
Enter the elements:
1 4 7 3 2 9
Sorted array: [1, 2, 3, 4, 7, 9]
```

2.QUICK SORT

```
import java.util.Scanner;

class quickSort {
    static void quickSort(int arr[], int low, int high) {
        if (low < high) {
            int pivotIndex = partition(arr, low, high);
            quickSort(arr, low, pivotIndex - 1);
            quickSort(arr, pivotIndex + 1, high);
        }
    }

    static int partition(int arr[], int low, int high) {
        int pivot = arr[high];
```

```
    int i = low - 1;
    for (int j = low; j < high; j++) {
        if (arr[j] <= pivot) {
            i++;
            int temp = arr[i];
            arr[i] = arr[j];
            arr[j] = temp;
        }
    }
    int temp = arr[i + 1];
    arr[i + 1] = arr[high];
    arr[high] = temp;
    return i + 1;
}

public static void main(String[] args) {
    Scanner sc = new Scanner(System.in);
    System.out.print("Enter the number of elements: ");
    int n = sc.nextInt();
    int[] arr = new int[n];

    System.out.println("Enter the elements:");
    for (int i = 0; i < n; i++) {
        arr[i] = sc.nextInt();
    }

    quickSort(arr, 0, n - 1);

    System.out.println("Sorted array: " + java.util.Arrays.toString(arr));
}
```

}

```
Enter the number of elements: 6
Enter the elements:
1 4 5 1 8 9
Sorted array: [1, 1, 4, 5, 8, 9]
```

3.NON REPEATING CHARACTERS

```
import java.util.*;


class nonRepeat {
    static char nonRepeatingChar(String s) {
        Map<Character, Integer> frequencyMap = new HashMap<>();
        for (char c : s.toCharArray()) {
            frequencyMap.put(c, frequencyMap.getOrDefault(c, 0) + 1);
        }
        for (char c : s.toCharArray()) {
            if (frequencyMap.get(c) == 1) {
                return c;
            }
        }
        return '$';
    }


    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter a string: ");
        String s = sc.nextLine();
        System.out.println("First non-repeating character: " + nonRepeatingChar(s));
    }
}
```

```
Enter a string: racecar
First non-repeating character: e
```

4.EDIT DISTANCE

```
import java.util.*;


class edit {
    static int minEditDistance(String s1, String s2) {
        int m = s1.length();
        int n = s2.length();
        int[][] dp = new int[m + 1][n + 1];


        for (int i = 0; i <= m; i++) {
            for (int j = 0; j <= n; j++) {
                if (i == 0) {
                    dp[i][j] = j;
                } else if (j == 0) {
                    dp[i][j] = i;
                } else if (s1.charAt(i - 1) == s2.charAt(j - 1)) {
                    dp[i][j] = dp[i - 1][j - 1];
                } else {
                    dp[i][j] = 1 + Math.min(dp[i - 1][j - 1], Math.min(dp[i][j - 1], dp[i - 1][j]));
                }
            }
        }
        return dp[m][n];
    }


    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the first string: ");
```

```
        String s1 = sc.nextLine();
        System.out.print("Enter the second string: ");
        String s2 = sc.nextLine();
        System.out.println("Minimum operations required: " + minEditDistance(s1, s2));
    }
}
```

```
Enter the first string: start
Enter the second string: strat
Minimum operations required: 2
```

5.K LARGEST ELEMENT

```
import java.util.*;

class kLargest {
    static List<Integer> findKLargest(int[] arr, int k) {
        PriorityQueue<Integer> minHeap = new PriorityQueue<>();
        for (int num : arr) {
            minHeap.add(num);
            if (minHeap.size() > k) {
                minHeap.poll();
            }
        }
        List<Integer> result = new ArrayList<>(minHeap);
        result.sort(Collections.reverseOrder());
        return result;
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the size of the array: ");
        int n = sc.nextInt();
```

```
        int[] arr = new int[n];

        System.out.println("Enter the elements of the array:");
        for (int i = 0; i < n; i++) {
            arr[i] = sc.nextInt();
        }

        System.out.print("Enter the value of k: ");
        int k = sc.nextInt();

        List<Integer> result = findKLargest(arr, k);
        System.out.println("K largest elements: " + result);
    }
}
```

```
Enter the size of the array: 6
Enter the elements of the array:
15 60 70 90 100 500
Enter the value of k: 3
K largest elements: [500, 100, 90]
```

6.FORM THE LARGEST NUMBER

```
import java.util.*;

class largestNumber {
    static String formLargestNumber(int[] arr) {
        String[] nums = Arrays.stream(arr).mapToObj(String::valueOf).toArray(String[]::new);
        Arrays.sort(nums, (a, b) -> (b + a).compareTo(a + b));
        if (nums[0].equals("0")) return "0";
        StringBuilder result = new StringBuilder();
        for (String num : nums) {
            result.append(num);
        }
```

```
        return result.toString();
    }

    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter the size of the array: ");
        int n = sc.nextInt();
        int[] arr = new int[n];

        System.out.println("Enter the elements of the array:");
        for (int i = 0; i < n; i++) {
            arr[i] = sc.nextInt();
        }

        System.out.println("Largest number formed: " + formLargestNumber(arr));
    }
}
```

```
Enter the size of the array: 7
Enter the elements of the array:
14 20 97 86 54 64 93
Largest number formed: 97938664542014
```