JAVA CODING CHALLENGE: 29-05-2020

1. Write a Program in Java to check whether a number is an Armstrong Number or not.

SAMPLE OUTPUT:

```
Input an integer:  153
Is Armstrong number? True
```

Solution:

```
import java.util.*;
public class solution {
          public static boolean is_Amstrong(int n)
         {
                  int remainder, sum = 0, temp = 0;
                  temp = n;
                  while (n > 0)
                   {
                           remainder = n % 10;
                           sum = sum + (remainder * remainder * remainder);
                           n = n / 10;
                  }return sum == temp;
         }
         public static void main(String[] args)
         {
                  Scanner scanner = new Scanner(System.in);
                  System.out.print("Input an integer: ");
                  String input = scanner.nextLine();
                  int number = Integer.parseInt(input);
                  System.out.println("Is Armstrong number? "+is_Amstrong(number));
         }
}
```

****

2. Write a Java program to find the second most frequent character in a given string.

Sample Output:

```
The given string is: successes
The second most frequent char in the string is: c
```

Solution:

```
import java.util.*;

public class Main

{

    static final int NOOFCHARS = 256;

    static char get2ndMostFreq(String str1)

    {

        int[] ctr = new int[NOOFCHARS];

        int i;

        for (i = 0; i < str1.length(); i++)

        (ctr[str1.charAt(i)]) ++;

         int ctr_first = 0, ctr_second = 0;

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

        {

            if (ctr[i] > ctr[ctr_first]) {

            ctr_second = ctr_first;

            ctr_first = i;

        }

        else if (ctr[i] > ctr[ctr_second] && ctr[i] != ctr[ctr_first])

         ctr_second = i;

    }

    return (char) ctr_second;
```

```
        }
        public static void main(String args[])
        {
                String str1 = "successes";
                System.out.println("The given string is: " + str1);
                char res = get2ndMostFreq(str1);
                 if (res != '\0')
                System.out.println("The second most frequent char in the string is: " + res);
                else
                System.out.println("No second most frequent character found in the string.");
        }
}
```

****

3. Write a Java program to find the length of the longest consecutive elements sequence from a given unsorted array of integers.

SAMPLE OUTPUT:

Sample array: [49, 1, 3, 200, 2, 4, 70, 5]
The longest consecutive elements sequence is [1, 2, 3, 4, 5], therefore the program will return its length 5

Solution:

```
import java.util.HashSet;
public class Example
{
        public static void main(String[] args)
        {
                 int nums[] = {49, 1, 3, 200, 2, 4, 70, 5};
                System.out.println("Original array length: "+nums.length);
                System.out.print("Array elements are: ");
                 for (int i = 0; i < nums.length; i++)
                {
```

```
            System.out.print(nums[i]+" ");
        }
        System.out.println("\nThe new length of the array is: "+longest_sequence(nums));
    }
    public static int longest_sequence(int[] nums)
    {
        final HashSet<Integer> h_set = new HashSet<Integer>();
        for (int i : nums) h_set.add(i);
        int longest_sequence_len = 0;
        for (int i : nums)
        {
            int length = 1;
            for (int j = i - 1; h_set.contains(j); --j)
            {
                h_set.remove(j);
                ++length;
             }
            for (int j = i + 1; h_set.contains(j); ++j)
            {
                h_set.remove(j);
                ++length;
             }
            longest_sequence_len = Math.max(longest_sequence_len, length);
         }
        return longest_sequence_len;
    }
}
```

****

4. Write a Java program to segregate all 0s on left side and all 1s on right side of a given array of 0s and 1s.

SAMPLE OUTPUT:

Sample array: [1,0,1,1,0,0,1,1]
So, the Output must be: [0,0,0,1,1,1,1,1]

Solution:

```
import java.util.Arrays;

public class Main

{

        public static void main(String[] args)

        {

            int arr[] = new int[]{ 0, 0, 1, 1, 0, 1, 1, 1,0 };

            int result[];

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

            System.out.println(Arrays.toString(arr));

            int n = arr.length;

            result = separate_0_1(arr, n);

            System.out.println("So, the Output must be:");

            System.out.println(Arrays.toString(result));

        } static int [] separate_0_1(int arr[], int n)

        {

            int count = 0;

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

            {

                if (arr[i] == 0)

                count++;

            }

            for (int i = 0; i < count; i++)

            arr[i] = 0;

             for (int i = count; i < n; i++)ss

            arr[i] = 1;

            return arr;

         }

  }
```

****