Assignment Questions 10

Question 1

Given an integer `n`, return *`true` if it is a power of three. Otherwise, return `false`.*

An integer `n` is a power of three, if there exists an integer `x` such that `n == 3x`.

Example 1:

Input: n = 27 Output: true

Explanation: 27 = 33

Solution:-

```
class Solution
{
public boolean isPowerOfThree(int n) {
    int c=0;
    boolean x=false;
    if(n!=0){
    while(c==0){
        if(n==1){
            x=true;
            break;
        }
        c=n%3;
        n=n/3;
    }
    }
    return x;
}
}
```

Question 2

You have a list `arr` of all integers in the range `[1, n]` sorted in a strictly increasing order. Apply the following algorithm on `arr`:

- Starting from left to right, remove the first number and every other number afterward until you reach the end of the list.
- Repeat the previous step again, but this time from right to left, remove the rightmost number and every other number from the remaining numbers.
- Keep repeating the steps again, alternating left to right and right to left, until a single number remains.

Given the integer `n`, return *the last number that remains in* `arr`.

Example 1:

Input: n = 1

Output: 1

Solution:-

```
class Solution {
    public int lastRemaining(int n) {
        boolean left = true;
        int head = 1;
        int step = 1;
        int remain = n;
        while(remain > 1){
            if(left || remain % 2 == 1){
                head = head + step;
            }
            step = step * 2;
            remain = remain / 2;
            left = !left;
        }
        return head;
    }
}
```

Question 3

****Given a set represented as a string, write a recursive code to print all subsets of it. The subsets can be printed in any order.

Example 1:

Input : set = “abc”

Output : { “”, “a”, “b”, “c”, “ab”, “ac”, “bc”, “abc”}

Example 2:

Input : set = “abcd”

Output : { “”, “a” ,”ab” ,”abc” ,”abcd”, “abd” ,”ac” ,”acd”, “ad” ,”b”, “bc” ,”bcd” ,”bd” ,”c” ,”cd” ,”d” }

Solution:-

```
class Test{
	static void powerSet(String str, int index, String curr)
{
		int n = str.length();
		if (index == n) {
			System.out.println(curr);
			return;
		}
		powerSet(str, index + 1, curr + str.charAt(index));
		powerSet(str, index + 1, curr);
	}
	public static void main(String[] args)
	{
		String str = "abc";
		int index = 0;
		String curr = "";
		powerSet(str, index, curr);
	}
}
```

Question 4

Given a string calculate length of the string using recursion.

Examples:

Input : str = "abcd"

Output :4 Input :

str = "GEEKSFORGEEKS" Output :13

Solution:-

```
import java.util.*;
public class GFG{
	private static int recLen(String str)
	{
		if (str.equals(""))
			return 0;
		else
			return recLen(str.substring(1)) + 1;
	}
	public static void main(String[] args)
	{



		String str ="GeeksforGeeks";
		System.out.println(recLen(str));
	}
}
```

Question 5

We are given a string S, we need to find count of all contiguous substrings starting and ending with same character.

Examples :

Input : S = "abcab"

Output : 7

There are 15 substrings of "abcab"

a, ab, abc, abca, abcab, b, bc, bca

bcab, c, ca, cab, a, ab, b

Out of the above substrings, there

are 7 substrings : a, abca, b, bcab,

c, a and b.

Input : S = "aba"

Output : 4

The substrings are a, b, a and aba

Solution:-

```
public class GFG {

    static int countSubstringWithEqualEnds(String s)

    {

        int result = 0;

        int n = s.length();

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

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

                if (s.charAt(i) == s.charAt(j))
```

```
                                                result++;


                return result;

        }

                public static void main(String args[])

        {

                String s = "abcab";

                System.out.println(countSubstringWithEqualEnds(s));

        }

}
```

Question 6

The [tower of Hanoi](https://en.wikipedia.org/wiki/Tower_of_Hanoi) is a famous puzzle where we have three rods and **N** disks. The objective of the puzzle is to move the entire stack to another rod. You are given the number of discs **N**. Initially, these discs are in the rod 1. You need to print all the steps of discs movement so that all the discs reach the 3rd rod. Also, you need to find the total moves.**Note:** The discs are arranged such that the **top disc is numbered 1** and the **bottom-most disc is numbered N**. Also, all the discs have **different sizes** and a bigger disc **cannot** be put on the top of a smaller disc. Refer the provided link to get a better clarity about the puzzle.

Example 1:

Input:

N = 2

Output:

move disk 1 from rod 1 to rod 2

move disk 2 from rod 1 to rod 3

move disk 1 from rod 2 to rod 3

3

Explanation:For N=2 , steps will be

as follows in the example and total

3 steps will be taken.

Solution:-

```
import java.io.*;
import java.math.*;
import java.util.*;
class Test {
	static void towerOfHanoi(int n, char from_rod, char to_rod, char aux_rod)
	{
		if (n == 0) {
			return;
		}
		towerOfHanoi(n - 1, from_rod, aux_rod, to_rod);
		System.out.println("Move disk " + n + " from rod "
						+ from_rod + " to rod "
						+ to_rod);
		towerOfHanoi(n - 1, aux_rod, to_rod, from_rod);
	}
	public static void main(String args[])
	{
```

```
        int N = 3;

        towerOfHanoi(N, 'A', 'C', 'B');

    }

}
```

Question 7

Given a string **str**, the task is to print all the permutations of **str**. A **permutation** is an arrangement of all or part of a set of objects, with regard to the order of the arrangement. For instance, the words ‘bat’ and ‘tab’ represents two distinct permutation (or arrangements) of a similar three letter word.

Examples:

Input: str = “cd”

Output: cd dc

Input: str = “abb”

Output: abb abb bab bba bab bba

Solution:-

```
public class Test {

    static void printDistinctPermutn(String str,String ans)

    {

        if (str.length() == 0) {

            System.out.print(ans + " ");

            return;

        }

        boolean alpha[] = new boolean[26];


        for (int i = 0; i < str.length(); i++)
```

```
                char ch = str.charAt(i);
                String ros = str.substring(0, i) +
                                        str.substring(i + 1);
                if (alpha[ch - 'a'] == false)
                        printDistinctPermutn(ros, ans + ch);
                alpha[ch - 'a'] = true;
        }
    }

    // Driver code
    public static void main(String[] args)
    {
        String s = "geek";
        printDistinctPermutn(s, "");
    }
}
```

Question 8

Given a string, count total number of consonants in it. A consonant is an English alphabet character that is not vowel (a, e, i, o and u). Examples of constants are b, c, d, f, and g.

Examples :

Input : abc de

Output : 3

There are three consonants b, c and d.

Input : geeksforgeeks portal

Output : 12

Solution:-

```
import java.util.*;
import java.lang.*;
class Solution
{
static boolean isConsonant(char ch)
{
	ch = Character.toUpperCase(ch);
	return (ch == 'A' || ch == 'E' ||
			ch == 'I' || ch == 'O' ||
			ch == 'U')== false && ch >= 65 && ch <= 90;
}
static int totalConsonants(String str, int n)
{
	if (n == 1)
	{
		if(isConsonant(str.charAt(0)))
			return 1;
		else
			return 0;
	}
```

```
        if(isConsonant(str.charAt(n - 1)))
                return totalConsonants(str, n - 1) + 1;
        else
                return totalConsonants(str, n - 1);
}
public static void main(String args[])
{
        String str = "abc de";
        System.out.println(totalConsonants(str, str.length()));
}
}
```