Recursion

EASY 01. Write a program to print natural numbers from 1 to n by using recursion.

```
package com.ds;
import java.util.Scanner;
class Demo
{
static void print(int n){
if(n>=1)
{
//System.out.print(n+" "); //===> n, n-1, n-2,... 1
print(n-1);
System.out.print(n+" "); // ==> 1, 2, 3, 4, .... n
}
}
public static void main(String[] args)
{
Scanner obj = new Scanner(System.in);
System.out.println("Enter a Number");
int n = obj.nextInt();
Demo.print(n);
}
}
```

Result:-

```
Enter a Number
5
1 2 3 4 5
```

EASY 02. Write a program to calculate sum of 'n' natural numbers by using recursion.

```
package com.ds;
import java.util.Scanner;
class Demo
{
static int sum(int n){
if(n==1)
{
return 1;
}
else
{
return n+sum(n-1);
```

```
}
}
public static void main(String[] args)
{
Scanner obj = new Scanner(System.in);
System.out.println("Enter a Number");
int n = obj.nextInt();
System.out.println(Demo.sum(n));
}
}
```

Result:-

```
Enter a Number
10
55
```

MEDIUM 03. Write a program to calculate a^b (a to the power b) by using recursion.

```
package com.ds;
import java.util.Scanner;
class Demo
{
static int power(int a,int b)
{
if(b>=1)
{
return a*power(a,b-1);
}
else
{
return 1;
}
}
public static void main(String[] args)
{
Scanner obj = new Scanner(System.in);
System.out.println("Enter Cofficient");
int a = obj.nextInt();
System.out.println("Enter Exponent");
int b = obj.nextInt();
System.out.println("Result="+Demo.power(a,b));
}
}
```

Result:-

```
Enter Cofficient
5
Enter Exponent
3
Result=125
```

EASY 4. Write a program to find the factorial of the given number by using recursion.

```
package com.ds;
import java.util.Scanner;
class Demo
{
static int fact(int n)
{
if(n==0)
{
return 1;
}
else
{
return n*fact(n-1);
}
}
public static void main(String[] args)
{
Scanner obj = new Scanner(System.in);
System.out.println("Enter Number");
int n = obj.nextInt();
System.out.println("Sum="+Demo.fact(n));
}
}
Result:-
Enter Number
4
Sum=24
```

MEDIUM 07. Write a program to check whether the given number is a prime number or not by using recursion.

```
package com.ds;
import java.util.Scanner;
class Demo
{
static boolean isprime(int n,int i)
```

```
{
if(i==1)
{
return true;
}
else if(n%i==0)
{
return false;
}
else
{
return isprime(n,--i);
}
}
public static void main(String[] args)
{
Scanner obj = new Scanner(System.in);
System.out.println("Enter Number:");
int n = obj.nextInt();
System.out.println(Demo.isprime(n,n/2));//true or false
}
}
```

Result:-

```
Enter Number:
11
True
```

HARD 08. Write a program to find a sum of digits present in the given number by using recursion.

```
package com.ds;
import java.util.Scanner;
class Demo
{
static int sumofdigits(int n)
{
if(n==0)
return 0;
else
return (n%10)+sumofdigits(n/10);
}
public static void main(String[] args)
{
Scanner obj = new Scanner(System.in);
```

```
System.out.println("Enter n value:");
int n = obj.nextInt();
System.out.println(Demo.sumofdigits(n));
}
}
Result:-
Enter Number:
8888
32
```

HARD 09. Write a program to calculate the reverse of the given number by
using recursion.
Formula to find reverse :-((n%10)*pow(10,len-1))+rev(n/10,--len)
n=98123, len=5 ----> 3*pow(10,4) + rev(9812,4) ----> 3*10000=30000
n=9812, len=4 -----> 2*pow(10,3) + rev(981,3) -----> 2*1000 = 2000
n=981, len=3 ------> 1*pow(10,2) + rev(98,2) ------> 1*100 = 100
n=98, len=2 -------> 8*pow(10,1) + rev(9,1) -------> 8*10 = 80
n=9, len=1 --------> 9*pow(10,0) + rev(0,0) -------> 9*1 = 9
n=0 ---------------> terminate --------------------> 32189
rev(98123) = 32189

```
package com.ds;
import java.util.Scanner;
class Demo
{
static int reverse(int n,int len)
{
if(n==0)
{
return 0;
}
else
{
return ((n%10)*(int)Math.pow(10,len-1)) + reverse(n/10,--len);
}
}
public static void main(String[] args)
{
Scanner obj = new Scanner(System.in);
System.out.println("Enter Value");
String s = obj.nextLine();
System.out.println("Reverse Result:-
"+Demo.reverse(Integer.parseInt(s),s.length()));//reverse of 'n'
```

```
}
}
Result:-
Enter Value
12345
Reverse Result:-54321
```

MEDIUM 10. Write a program to count the number of digits present in the given number by using recursion.

```
package com.ds;
import java.util.Scanner;
class Demo
{
static int c=0;
static int count(int n)
{
if(n!=0)
{
c++;
count(n/10);
}
return (c!=0)?c:1;
}
public static void main(String[] args)
{
Scanner obj = new Scanner(System.in);
System.out.print("Enter Number:::");
int n = obj.nextInt();
System.out.println("Number Of Digit::"+Demo.count(n));
}
}
Result:-
Enter Number:::12346
Number Of Digit::5
```

11. Write a program to convert decimal number into binary by using recursion.

```
package com.ds;
import java.util.Scanner;
class Demo
{
static int convert(int n)
{
```

```
if(n==0)
{
return 0;
}
else
{
return (n%2+10*convert(n/2));
}
}
public static void main(String[] args)
{
Scanner obj = new Scanner(System.in);
System.out.println("Enter Decimal Value::");
int n = obj.nextInt();
System.out.println("Binary Value is::"+Demo.convert(n));
}
}
```

Result:-

```
Enter Decimal Value::10
Binary Value is::1010
```

HARD 12. Implement a program to find the nth Fibonacci number by using recursion.

```
package com.ds;
import java.util.Scanner;
class Demo
{
static int fib(int n)
{
if(n==0 || n==1)
{
return n;
}
else
{
return fib(n-1)+fib(n-2);
}
}
public static void main(String[] args)
{
Scanner obj = new Scanner(System.in);
System.out.print("Enter Number:: ");
int n = obj.nextInt();
```

```
for(int i=0;i<n;i++){
System.out.print(Demo.fib(i)+", ");
}
}
}
```

Result:-

```
Enter Number:: 10
0, 1, 1, 2, 3, 5, 8, 13, 21, 34,
```

MEDIUM 13 Write a program to find the reverse of the given string using
recursion.

```
package com.ds;
import java.util.Scanner;
class Demo
{
static String strrev(String s)
{
if(s==null || s.length()<=1)//BC
{
return s;
}
return strrev(s.substring(1))+s.charAt(0);
}
public static void main(String[] args)
{
Scanner obj = new Scanner(System.in);
System.out.print("Enter any String:");
String s = obj.nextLine();
System.out.println("Reverse Result::"+Demo.strrev(s));
}
}
```

Result:-

```
Enter any String:abcdefghi
Reverse Result::ihgfedcba
```

HARD 13. Write a program to remove the given character from a string by
using recursion.

```
package com.ds;
import java.util.Scanner;
class Demo
{
```

```
static String newS(String s,int index)
{
if(index<1)
{
return s.substring(0,index+1);//s.charAt(index)+"";
}
return newS(s,index-1)+"*"+s.charAt(index);
}
public static void main(String[] args)
{
Scanner obj = new Scanner(System.in);
System.out.print("Enter any string:");
String s = obj.nextLine();
System.out.println(Demo.newS(s,s.length()-1));//abc ---> a*b*c
}
}
```

```
Result:-
Enter any string:
axbxcxxdefxghxx
abcdefgh
```

HARD 14) Write a program to return a new String, where all the adjacent characters are separated by a "*" by using recursion.

```
"hello" ----> "h*e*l*l*o"
"abc" ------> "a*b*c"
"ab" -------> "a*b"
package com.ds;
import java.util.Scanner;
class Demo
{
static String newS(String s,int index)
{
if(index<1)
return s.substring(0,index+1);//s.charAt(index)+"";
return newS(s,index-1)+"*"+s.charAt(index);
}
public static void main(String[] args)
{
Scanner obj = new Scanner(System.in);
System.out.println("Enter any string:");
String s = obj.nextLine();
System.out.println(Demo.newS(s,s.length()-1));//abc ---> a*b*c
}
```

```
}
Result:-
Enter any string:
afghjmot
a*f*g*h*j*m*o*t
```

15) Implement a program to return a new string where identical adjacent chars are separate by *.

Ex:

abc ----> abc

hello --> hel*lo

xxyy ---> x*xy*y

```
package com.ds;
import java.util.Scanner;
class Demo
{
static String newS(String s,int index)
{
if(index<1)
{
return s.substring(0,index+1);
}
if(s.charAt(index-1)==s.charAt(index))
{
return newS(s,index-1)+"*"+s.charAt(index);
}
else
{
return newS(s,index-1)+s.charAt(index);
}
}
public static void main(String[] args)
{
Scanner obj = new Scanner(System.in);
System.out.print("Enter any string::");
String s = obj.nextLine();
System.out.println(Demo.newS(s,s.length()-1));//abc ---> a*b*c
}
}
Enter any string::hello
hel*lo
```

MEDIUM 16) Write a program to count the number of times, the given char
occurred by using recursion.

```
package com.ds;
import java.util.Scanner;
class Demo
{
static int count(String s,char ch,int index) //x
{
if(index<0)
{
return 0;
}
if(s.charAt(index)==ch)
{
return 1+count(s,ch,index-1);
}
else
{
return count(s,ch,index-1);
}
}
public static void main(String[] args)
{
Scanner obj = new Scanner(System.in);
System.out.println("Enter any string:");
String s = obj.nextLine();
System.out.println(Demo.count(s,'a',s.length()-1));
}
}
```

Result:-

```
Enter any string:
abcabcabc
3
```

17) Write to replace the given old character with a new character in the original string by using recursion.

```
'x' --------> 'y'
"codex" ----> "codey"
"xxhixx" ---> "yyhiyy"
"xbix" -----> "ybiy"
package com.ds;
import java.util.Scanner;
class Demo
```

```
{
static String replace(String s,int index)
{
//Base condition
if(index<0)
return "";
if(s.charAt(index)=='x')
return replace(s,index-1)+"y";
else
return replace(s,index-1)+s.charAt(index);
}
public static void main(String[] args)
{
Scanner obj = new Scanner(System.in);
System.out.println("Enter any string:");
String s = obj.nextLine();
System.out.println(Demo.replace(s,s.length()-1));
}
}
Result:-
Enter any string:
abcdefghxxaxtccx
abcdefghyyaytccy
```

MEDIUM Q:-18 Find max from array by using recursion.

```
 public class RecursionDemo {
    public static void main(String[] args) {
       int []arr={15,22,11,23,44,1,5,2};
        System.out.println(new RecursionDemo().findMax(arr,0));
    }
 public int findMax(int []arr, int idx)
 {
     if(idx==arr.length-1)
     {
         return arr[idx];
     }
     int tmax=findMax(arr,idx+1);
     if(tmax>arr[idx])
     {
         return tmax;
     }
     else
     {
```

```
            return arr[idx];
        }
    }
}
MEDIUM Q:-19 Find first index value of given target.
public class RecursionDemo {
    public static void main(String[] args) {
       int []arr={15,21,11,23,44,11,22,2};
        System.out.println(new
RecursionDemo().firstOccurance(arr,0,2));
    }
 public int firstOccurance(int []arr,int idx,int target)
 {
     if(idx==arr.length)
     {
         return -1;
     }
     if(target==arr[idx])
     {
         return idx;
     }
     else
     {
         return firstOccurance(arr,idx+1,target);
     }
 }
}
MEDIUM Q:-20 Find the last occurrence.
public class RecursionDemo {
    public static void main(String[] args) {
        int[]arr={1,2,3,3,4,4,5,6,3,2};
        System.out.println(new
RecursionDemo().lastOccurance(arr,1,0));
    }
    public int lastOccurance(int []arr, int target,int idx)
    {
        if(arr.length==idx)
        {
            return -1;
        }
        int loc=lastOccurance(arr,target,idx+1);
        if(loc==-1)
        {
```

```
            if(arr[idx]==target)
            {
                return idx;
            }
            else
            {
                return -1;
            }
        }
        else {
            return loc;
        }
    }
}
```

HARD Q:-21 Write a program to print Zig-Zag.

```
import java.util.Scanner;
public class RecursionDemo {
    public static void main(String[] args) {
        Scanner sc=new Scanner(System.in);
        System.out.println("Enter Number");
        int n=sc.nextInt();
        printZigZag(n);
    }
    public static void printZigZag(int n)
    {
        if(n==0)
        {
            return ;
        }
        System.out.println("PRE:: "+n);
        printZigZag(n-1);
        System.out.println("IN:: "+n);
        printZigZag(n-1);
        System.out.println("POST:: "+n);
    }
}
```

Result:-

```
Enter Number
2
PRE:: 2
PRE:: 1
IN:: 1
```

```
POST:: 1
IN:: 2
PRE:: 1
IN:: 1
POST:: 1
POST:: 2
```

HARD Q:-22. Towers of Hanoi:- Tower of Hanoi is a game of rods and discs that requires a certain number of discs of different sizes to be transferred
from one rod to another.
Rule of Towers of Hanoi game:-
1. We play this game with the help of only three rods source, destination, and helper.
2. Only one disk can be moved at a time.
3. No disk may be placed on top of a smaller disk.
program to implements Towers of Hanoi game.

```
package com.ds;
import java.util.Scanner;
class Demo
{
static void towersOfHanoi(int n,String src,String helper,String
dest)
{
if(n==1){
System.out.println("Move The Disk "+n+" from "+src+" to "+dest);
return;
}
towersOfHanoi(n-1,src,dest,helper);
System.out.println("Move The Disk "+n+" from "+src+" to "+dest);
towersOfHanoi(n-1,helper,src,dest);
}
public static void main(String[] args)
{
Scanner obj = new Scanner(System.in);
System.out.println("Enter number of disks:");
int n=obj.nextInt();
Demo.towersOfHanoi(n,"S","H","D");
}
}
```

Result:-

```
Enter number of disks:
3
```

```
Move The Disk 1 from S to D
Move The Disk 2 from S to H
Move The Disk 1 from D to H
Move The Disk 3 from S to D
Move The Disk 1 from H to S
Move The Disk 2 from H to D
Move The Disk 1 from S to D
```