Recursion Assignment

```
Geometric Sum
Send Feedback
Given k, find the geometric sum i.e.

1 + 1/2 + 1/4 + 1/8 + ... + 1/(2^k)

using recursion.
Input format :

Integer k

Output format :

Geometric sum (upto 5 decimal places)

Constraints :

0 <= k <= 1000

Sample Input 1 :

3

Sample Output 1 :

1.87500

Sample Input 2 :

4

Sample Output 2 :

1.93750

Explanation for Sample Input 1:

1+ 1/(2^1) + 1/(2^2) + 1/(2^3) = 1.87500

import java.lang.Math;
public class solution {

   public static double findGeometricSum(int k){

       if(k==0)
           return 1;

       double smallOutput = 1/Math.pow(2,k);

       return smallOutput+findGeometricSum(k-1);
   }
}
```

```
Check Palindrome (recursive)
Send Feedback
Check whether a given String S is a palindrome using recursion. Return
true or false.
Input Format :

String S

Output Format :

'true' or 'false'

Constraints :

0 <= |S| <= 1000
where |S| represents length of string S.

Sample Input 1 :

racecar

Sample Output 1:

true

Sample Input 2 :

ninja

Sample Output 2:

false

public class solution {

   public static boolean isStringPalindrome(String input) {
       int n = input.length();
       if (n == 0)
           return true;

       return isPalRec(input, 0, n - 1);

   }

   public static boolean isPalRec(String s, int si, int ei){

       if(si==ei || si>ei)
           return true;

       if(s.charAt(si)!=s.charAt(ei))
           return false;
```

```
        return isPalRec(s, si+1, ei-1);
    }
}
```

```
Sum of digits (recursive)
Send Feedback
Write a recursive function that returns the sum of the digits of a given
integer.
Input format :

Integer N

Output format :

Sum of digits of N

Constraints :

0 <= N <= 10^9

Sample Input 1 :

12345

Sample Output 1 :

15

Sample Input 2 :

9

Sample Output 2 :

9

public class solution {

     static int sumOfDigits(int n)
    {
        if (n == 0)
            return 0;
        return (n % 10 + sumOfDigits(n / 10));
    }
}
```

```
Multiplication (Recursive)
Send Feedback
Given two integers M & N, calculate and return their multiplication using
recursion. You can only use subtraction and addition for your calculation.
No other operators are allowed.
Input format :

Line 1 : Integer M
Line 2 : Integer N

Output format :

M x N

Constraints :

0 <= M <= 1000
0 <= N <= 1000

Sample Input 1 :

3
5

Sample Output 1 :

15

Sample Input 2 :

4
0

Sample Output 2 :

0


public class solution {

   public static int multiplyTwoIntegers(int m, int n){
        // Write your code here
        if(n==0)
            return 0;
        return(m+multiplyTwoIntegers(m,n-1));
   }
}
```

```
Count Zeros
Send Feedback
Given an integer N, count and return the number of zeros that are present
in the given integer using recursion.
Input Format :

Integer N

Output Format :

Number of zeros in N

Constraints :
```

```
0 <= N <= 10^9

Sample Input 1 :

0

Sample Output 1 :

1

Sample Input 2 :

00010204

Sample Output 2 :

2

Explanation for Sample Output 2 :

Even though "00010204" has 5 zeros, the output would still be 2 because
when you convert it to an integer, it becomes 10204.

Sample Input 3 :

708000

Sample Output 3 :

4

public class solution {

   public static int countZerosRec(int n){

       if(n==0)
           return 1;
       if(n/10==0)
           return 0;
       if(n%10 == 0)
           return(1+countZerosRec(n/10));
       else
           return(countZerosRec(n/10));
   }
}
```

```
String to Integer
Send Feedback
Write a recursive function to convert a given string into the number it
represents. That is input will be a numeric string that contains only
numbers, you need to convert the string into corresponding integer and
return the answer.
Input format :
```

```
Numeric string S (string, Eg. "1234")

Output format :

Corresponding integer N (int, Eg. 1234)

Constraints :

0 < |S| <= 9
where |S| represents length of string S.

Sample Input 1 :

00001231

Sample Output 1 :

1231

Sample Input 2 :

12567

Sample Output 2 :

12567

public class solution {

   public static int convertStringToInt(String input){

        if(input.length()==1)
            return input.charAt(0)-48;

        int a = input.charAt(0)-48;
        int smalloutput = convertStringToInt(input.substring(1));
        return smalloutput+a*(int)Math.pow(10,input.length()-1);
   }
}
```

```
Pair Star
Send Feedback
Given a string S, compute recursively a new string where identical chars
that are adjacent in the original string are separated from each other by
a "*"."
Input format :

String S

Output format :

Modified string

Constraints :
```

```
0 <= |S| <= 1000
where |S| represents length of string S.

Sample Input 1 :

hello

Sample Output 1:

hel*lo

Sample Input 2 :

aaaa

Sample Output 2 :

a*a*a*a

public class solution {

   // Return the updated string
   public static String addStars(String s) {

       if(s.length()==1)
           return s;

       String ns;
       if(s.charAt(0)==s.charAt(1))
           ns = s.charAt(0)+"*"+addStars(s.substring(1));
       else
           ns = s.charAt(0)+addStars(s.substring(1));

       return ns;
   }

}
```

```
Check AB
Send Feedback
Suppose you have a string, S, made up of only 'a's and 'b's. Write a
recursive function that checks if the string was generated using the
following rules:

a. The string begins with an 'a'
b. Each 'a' is followed by nothing or an 'a' or "bb"
c. Each "bb" is followed by nothing or an 'a'

If all the rules are followed by the given string, return true otherwise
return false.
Input format :

String S
```

```
Output format :

'true' or 'false'

Constraints :

1 <= |S| <= 1000
where |S| represents length of string S.

Sample Input 1 :

abb

Sample Output 1 :

true

Sample Input 2 :

abababa

Sample Output 2 :

false

Explanation for Sample Input 2

In the above example, a is not followed by either "a" or "bb", instead
it's' followed by "b" which results in false to be returned.

public class Solution {

   public static boolean checkAB(String input) {

       if(input.length() == 0){
           return true;
       }

     if(input.charAt(0) == 'a'){
         if(input.substring(1).length() > 1 &&
input.substring(1,3).equals("bb")){
             return checkAB(input.substring(3));
         }else{
              return checkAB(input.substring(1));
         }
     }
       return false;
   }
}
```

```
Staircase
Send Feedback
```

```
A child is running up a staircase with N steps, and can hop either 1 step,
2 steps or 3 steps at a time. Implement a method to count how many
possible ways the child can run up to the stairs. You need to return
number of possible ways W.
Input format :

Integer N

Output Format :

Integer W

Constraints :

1 <= N <= 30

Sample Input 1 :

4

Sample Output 1 :

7

Sample Input 2 :

5

Sample Output 2 :

13

public class Solution {


    public static int staircase(int n){

       if (n < 0)
           return 0;
       else if ( n == 0)
           return 1;
       else
           return staircase(n - 3) + staircase(n - 2) + staircase(n - 1);
   }
}
```