1. Factorial

Given n of 1 or more, return the factorial of n, which is n \* (n-1) \* (n-2) ... 1. Compute the result recursively (without loops).

factorial(1) → 1  
factorial(2) → 2  
factorial(3) → 6

public int factorial(int n) {

if(n==0)

return 1;

else

{

return n\*factorial(n-1);

}

}

2. Bunny Ears

We have a number of bunnies and each bunny has two big floppy ears. We want to compute the total number of ears across all the bunnies recursively (without loops or multiplication).

bunnyEars(0) → 0  
bunnyEars(1) → 2  
bunnyEars(2) → 4

public int bunnyEars(int b) {

if(b==0)

{

return 0;

}

else

{

return 2+bunnyEars(b-1);

}

}

3. Fibonacci

The fibonacci sequence is a famous bit of mathematics, and it happens to have a recursive definition. The first two values in the sequence are 0 and 1 (essentially 2 base cases). Each subsequent value is the sum of the previous two values, so the whole sequence is: 0, 1, 1, 2, 3, 5, 8, 13, 21 and so on. Define a recursive fibonacci(n) method that returns the nth fibonacci number, with n=0 representing the start of the sequence.

fibonacci(0) → 0  
fibonacci(1) → 1  
fibonacci(2) → 1

public int fibonacci(int n) {

if(n==0)

{

return 0;

}

if(n==1)

{

return 1;

}

else

{

return fibonacci(n-1)+fibonacci(n-2);

}

}

4. Bunny Ears 2

We have bunnies standing in a line, numbered 1, 2, ... The odd bunnies (1, 3, ..) have the normal 2 ears. The even bunnies (2, 4, ..) we'll say have 3 ears, because they each have a raised foot. Recursively return the number of "ears" in the bunny line 1, 2, ... n (without loops or multiplication).

bunnyEars2(0) → 0  
bunnyEars2(1) → 2  
bunnyEars2(2) → 5

public int bunnyEars2(int b) {

if(b==0)

{

return 0;

}

if(b%2==0)

{

return 3+bunnyEars2(b-1);

}

else

{

return 2+bunnyEars2(b-1);

}

}

5.Triangle

We have triangle made of blocks. The topmost row has 1 block, the next row down has 2 blocks, the next row has 3 blocks, and so on. Compute recursively (no loops or multiplication) the total number of blocks in such a triangle with the given number of rows.

triangle(0) → 0  
triangle(1) → 1  
triangle(2) → 3

public int triangle(int r) {

if(r==0)

{

return 0;

}

else

{

return r+triangle(r-1);

}

}

6.Sum of Digits

Given a non-negative int n, return the sum of its digits recursively (no loops). Note that mod (%) by 10 yields the rightmost digit (126 % 10 is 6), while divide (/) by 10 removes the rightmost digit (126 / 10 is 12).

sumDigits(126) → 9  
sumDigits(49) → 13  
sumDigits(12) → 3

public int sumDigits(int n) {

if(n==0)

{

return 0;

}

else{

return n%10+sumDigits(n/10);

}

}

7. Count7

Given a non-negative int n, return the count of the occurrences of 7 as a digit, so for example 717 yields 2. (no loops). Note that mod (%) by 10 yields the rightmost digit (126 % 10 is 6), while divide (/) by 10 removes the rightmost digit (126 / 10 is 12).

count7(717) → 2  
count7(7) → 1  
count7(123) → 0

public int count7(int n) {

if(n==0)

{

return 0;

}

if(n%10==7)

{

return 1+count7(n/10);

}

return count7(n/10);

}

8. Given a non-negative int n, compute recursively (no loops) the count of the occurrences of 8 as a digit, except that an 8 with another 8 immediately to its left counts double, so 8818 yields 4. Note that mod (%) by 10 yields the rightmost digit (126 % 10 is 6), while divide (/) by 10 removes the rightmost digit (126 / 10 is 12).

count8(8) → 1  
count8(818) → 2  
count8(8818) → 4

public int count8(int n) {

int s=0,x=0;

if(n==0)

{

return 0;

}

if(n%100==88 )

{

return 2+count8(n/10);

}

if(n%10==8)

{

return 1+count8(n/10);

}

else

{

return count8(n/10);

}

}

9. Power N

Given **base** and **n** that are both 1 or more, compute recursively (no loops) the value of base to the n power, so powerN(3, 2) is 9 (3 squared).

powerN(3, 1) → 3  
powerN(3, 2) → 9  
powerN(3, 3) → 27

public int powerN(int base, int n) {

if(n==1)

{

return base;

}

else

{

return base\*powerN(base,n-1);

}

}

10.Given a string, compute recursively (no loops) the number of lowercase 'x' chars in the string.

countX("xxhixx") → 4  
countX("xhixhix") → 3  
countX("hi") → 0

public int countX(String str) {

int l = str.length();

if(l==0)

{

return 0;

}

if(str.charAt(l-1)=='x')

{

return 1+countX(str.substring(0,l-1));

}

else

{

return countX(str.substring(0,l-1));

}

}

11. Count “Hi”

Given a string, compute recursively (no loops) the number of times lowercase "hi" appears in the string.

countHi("xxhixx") → 1  
countHi("xhixhix") → 2  
countHi("hi") → 1

public int countHi(String str) {

int l = str.length();

if(l==0 ||l==1)

{

return 0;

}

if(str.charAt(l-1)=='i' && str.charAt(l-2)=='h')

{

return 1+countHi(str.substring(0,l-2));

}

else

{

return countHi(str.substring(0,l-1));

}

}

12. Change XY

Given a string, compute recursively (no loops) a new string where all the lowercase 'x' chars have been changed to 'y' chars.

changeXY("codex") → "codey"  
changeXY("xxhixx") → "yyhiyy"  
changeXY("xhixhix") → "yhiyhiy"

public String changeXY(String str) {

int l = str.length();

if(l==0)

{

return "";

}

if(str.charAt(0)=='x')

{

return "y"+changeXY(str.substring(1));

}

else

{

return str.charAt(0)+changeXY(str.substring(1));

}

}

13. Change Pi

Given a string, compute recursively (no loops) a new string where all appearances of "pi" have been replaced by "3.14".

changePi("xpix") → "x3.14x"  
changePi("pipi") → "3.143.14"  
changePi("pip") → "3.14p"

public String changePi(String str) {

int l = str.length();

if(l==0 )

{

return "";

}

if(l==1)

{

return str.charAt(0)+"" ;

}

if(str.charAt(0)=='p' && str.charAt(1)=='i')

{

return "3.14"+ changePi(str.substring(2));

}

else

{

return str.charAt(0) + changePi(str.substring(1));

}

}

14. No “X”

Given a string, compute recursively a new string where all the 'x' chars have been removed.

noX("xaxb") → "ab"  
noX("abc") → "abc"  
noX("xx") → ""

public String noX(String str) {

int l=str.length();

if(l==0)

{

return "";

}

if(str.charAt(0)=='x')

return noX(str.substring(1));

else

return str.charAt(0)+noX(str.substring(1));

}

:::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::Extra::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::

Decimal to Binary:

public static void decToBin(int n) {

        if (n > 0) {

        decToBin(n / 2);

        System.out.printf("%d", n % 2);

        }

        }