CSF
Hwk06
ReCursing

In this lab you will be creating a number of recursive methods. **You may not use iterative looping (while** and **for** statements**).** You can use any other Java constructs we have learned, however, you must use recursion to solve the problems. All inputs and outputs will be assumed to be >= 0.

1. Write method factRec() that takes an integer and returns an integer. It should return the factorial of the given number.
Ex: factRec(0) ==> 1, factRec(1) ==> 1, factRec(3) ==> 6, factRec(5) ==> 120

2. Write method sumRec() that takes an integer and returns an integer. It should return the sum from 1 to the given number.
Ex: sumRec(0) ==> 0, sumRec(1) ==> 1, sumRec(3) ==> 6, sumRec(5) ==> 15

3. Write a method multRec() that takes two integers and returns an integer. It should return the product of the two integers.
Ex: multRec(2,5) ==>10, multRec(5,2) ==> 10, multRec(5,0) ==> 0, multRec(0,5) ==> 0

4. Write a method powRec() that takes two integers and returns an integer. It should return the first number raised to the power of the second number. Note that we will declare 0^0 to be 1.
Ex: powRec(2,5) ==>32, powRec(0,5) ==> 0, powRec(5,0) ==> 1, powRec(0,0) ==> 1

5. Write a method divRec() that takes two integers and returns an integer. It should return the integer division of the two integers. Assume that it is not asked to divide by 0.
Ex: divRec(2,5) ==> 0, divRec(5,2) ==> 2, divRec(0,5) ==> 0, divRec(5,5) ==> 1

6. Write a method modRec() that takes two integers and returns an integer. It should return the modulo function of the two integers. Assume that it is not asked to modulo by 0.
Ex: myMod(2,5) ==> 2, myMod(5,2) ==> 1, myMod(0,5) ==> 0, myMod(5,5) ==> 0

7. Write a method nthFibRec() that takes an integer and returns an integer. It should return the n^{th} number in the Fibonacci sequence. The sequence is 1,1, 2, 3, 5, 8, 13, 21, 34, …
Ex: nthFibRec(2) ==> 1, nthFibRec(4) ==> 3, nthFibRec(6) ==> 8, nthFibRec(9) ==> 34

8. Write a method numDigitsRec() that takes an integer and returns an integer. It should return the number of digits composing the number. Remember: Each time you divide a number by 10, it reduces the number of digits by one, e.g. 172/10 = 17.
Ex: numDigitsRec(0) ==> 1, numDigitsRec(4) ==> 1, numDigitsRec(81) ==> 2
numDigitsRec(307) ==> 3, numDigitsRec(1201) ==> 4

9. Write a method sumDigitsRec() that takes an integer and returns an integer. It should return the sum of all the digits composing the number. Remember: If you take a number mod 10, it will give you the digit in the 1's place, e.g. 172%10 = 2
Ex: sumDigitsRec(4) ==> 4, sumDigitsRec(81) ==> 9
sumDigitsRec(307) ==> 10, sumDigitsRec(1201) ==> 4

10*. Write a method gcdRec() that takes two integers and returns an integer. It should return the greatest common divisor of the two numbers. Use and research Euclid's method.
Ex: gcdRec(13,17) ==> 1, gcdRec(360, 256) ==> 8
gcdRec(850,1836) ==> 34, gcdRec(893,190,) ==> 19

11*. Write a method gcdRec3() that takes three integers and returns an integer. It should return the greatest common divisor of the three numbers. Note: This method itself will not be recursive. Instead, it will use gcdRec, which is recursive.
Ex: gcdRec3(13,17, 24) ==> 1, gcdRec3(360, 256,204) ==> 4
gcdRec3(850,1836, 119) ==> 17, gcdRec3(140,893,190) ==> 1

12*. Write a method chooseRec() that takes two integers and returns an integer. It should return the choose function, choose(a,b) =(a! /(b! *(a-b)!)). Note that in the general case:

$\binom{x}{y} = \binom{x-1}{y} + \binom{x-1}{y-1}$, and you will need to handle three base cases.

Ex: chooseRec(2,5) ==> 0, chooseRec(5,2) ==> 10, chooseRec(5,1) ==> 5
chooseRec(5,5) ==> 1, chooseRec(5,0) ==> 1, chooseRec(0,5) ==> 0