CSF
Hwk05
Methods… Calling Methods….

In this lab you will be creating a number of methods that build from the bottom up to create mathematical functions. **You may not use +,-, *, /, or %.** You can use any other Java constructs we have learned, along with the form *x*++, which increments the variable *x* by 1, and x--, which decrements the variable *x* by 1. Note that we will not deal with negative numbers in this lab. All inputs and outputs will be assumed to be >= 0.

1. Write a method myAdd() that takes two integers and returns an integer. It should return the sum of the two integers.
Ex: myAdd(2,5) ==> 7, myAdd(5,2) ==> 7, myAdd(1,0) ==> 1, myAdd(5,5) ==> 10

2. Write a method mySub() that takes two integers and returns an integer. It should return the difference of the two integers. Note that if the second number is greater than the first, then the result is 0.
Ex: mySub(2,5) ==> 0, mySub(5,2) ==> 3, mySub(1,0) ==> 1, mySub(5,5) ==> 0

3. Write a method myMult() that takes two integers and returns an integer. It should return the multiplication of the two integers.
Ex: myMult(2,5) ==>10, myMult(5,2) ==> 10, myMult(1,0) ==> 0, myMult (5,5) ==> 25

4. Write a method myPow() 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: myPow(2,5) ==>32, myPow(0,5) ==> 0, myPow(5,0) ==> 1, myPow(0,0) ==> 1

5. Write method myFact() that takes an integer and returns an integer. It should return the factorial of the number given. Note that myFact(0) returns 1.
Ex: myFact(1) ==> 1, myFact(4) ==> 24, myFact(5) ==> 120, myFact(0) ==> 1

6. Write method mySqrt() that takes an integer and returns an integer. It should return the "integer" sqrt of the number, that is, the closest integer whose square it not greater than the given integer.
Ex: mySqrt(0) ==> 0, mySqrt(1) ==> 1, mySqrt(2) ==> 1, mySqrt(3) ==> 1, mySqrt(4) ==> 2

7. Write a method myDiv() 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: myDiv(2,5) ==> 0, myDiv(5,2) ==> 2, myDiv(0,5) ==> 0, myDiv(5,5) ==> 1

8. Write a method myMod() 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.
Hint: You don't need any loops for this one! mySub, myMult, and myDiv may be useful!
Ex: myMod(2,5) ==> 2, myMod(5,2) ==> 1, myMod(0,5) ==> 0, myMod(5,5) ==> 0

9. Write a method myChoose() that takes two integers and returns an integer. It should return the choose function, where choose(a,b) = (a! / (b! * (a-b)!)). You don't need any loops.
Ex: myChoose(2,5) ==> 0, myChoose(5,2) ==> 10, myChoose(5,1) ==> 5, myChoose(5,5) ==> 1, myChoose(5,0) ==> 1, myChoose(0,5) ==> 0

10. Write method isPrime() that takes an integer and returns a boolean, indicating whether or not the given number is prime. Use mySqrt() to make the function efficient.
Ex: isPrime(0) ==> false, isPrime(1) ==> false, isPrime(2) ==> true, isPrime(3) ==> true, isPrime(4) ==> false

Remember that we started all of this with just the increment operator x++ and the decrement operator x--. Do we even need the x--? Could we write our own decrement?

```
// myDec() takes an integer, x, and returns x-1.
public static int myDec(int x) {

  int lp=0, result=0;

  result = 0;
  for (lp=1;(lp<x); lp++) {
    result++;
  } // end for
  return result;
} // end myDec()
```