CSF
Hwk04
For ever…

1. Write method sumTo() that takes an integer and returns an integer. It should return the sum of all the integers between 1 and the number given.
Ex: sumTo(1) ==> 1, sumTo(4) ==> 10, sumTo(10) ==> 55, sumTo(0) ==> 0

2. Write a method sumOfSums() that takes an integer and returns an integer. It should return the sum of all the "sumTo's" of the integers between 1 and the number given. Use your sumTo() method to help.
Ex: sumofSums(1) ==> 1, sumTo(4) ==> 20, sumTo(7) ==> 84, sumTo(10) ==> 220,
sumTo(0) ==> 0

3. Write a method sumOfEvenBetween() that takes two integers and returns an integer. It should return the sum of all the integers between the first number and the second number that are even.
Note: Either number can be the largest.
Ex: sumOfEvenBetween(2,5) ==> 6, sumOfEvenBetween(5,2) ==> 6
sumOfEvenBetween(1,10) ==> 30, sumOfEvenBetween(5,5) ==> 0

4. Write a method sumPropDivisors() that takes an integer (x) and returns an integer. It should return the sum of all the positive integers less than x that divide evenly into x.
Ex: sumPropDivisors(0) ==> 0, sumPropDivisors(1) ==> 0, sumPropDivisors(6) ==> 6,
sumPropDivisors(10) ==> 8, sumPropDivisorsv(13) ==> 1

5. Write a method isPerfect() that takes an integer (greater than 0) and returns a boolean. It should return true if the number equals the sum of its proper divisors. Such a number is called a "perfect number". Use sumPropDivisors() as a helper method.
Ex: isPerfect(1) ==>false, isPerfect(4) ==> false, isPerfect(6) ==> true
isPerfect(10) ==> false, isPerfect(28) ==> true, isPerfect(496) ==> true

6. Write a method isPrime() that takes an integer and returns a boolean. It should return true if the number is prime, and false otherwise. Note that one way to determine if a number is prime is to see if the sum of its proper divisors is 1. Use sumPropDivisors() as a helper method. Remember, 1 is not a prime and therefore it is again an exception.
Ex: isPrime(1) ==>false, isPrime(2) ==> true, isPrime(5) ==> true, isPrime(9) ==> false
isPrime(93) ==> false, isPrime(101) ==> true

7* Write a method nthPerfect() that given integer *n*, will return the *n*th perfect number. Use it to find the 4th perfect number. The first three are 6, 28, and 496. There is no test method for this one.
Ex: nthPerfect(1) ==> 6, nthPerfect(2) ==> 28, nthPerfect(3) ==> 496, nthPerfect(4) ==> ???

8* Write a method areAmicable() that determines if two numbers are amicable. It takes two integers and returns a boolean. Two numbers *x* and *y* are considered amicable if they are not the same number and the sum of the proper divisors of *x* equals *y* and vice versa. There is no test method for this one.

9* Write a method nthAmicable() that takes an integer *n* and determines the *n*th amicable pair. Note: Since you can't return two values, simply have the method return the smaller of the two values. If it returned value x, then the other member must be sumPropDivisors(x). You may also decide to put print statements in your method to display the answer. Your method should not have any "magic" numbers in it. Can you make your method reasonably efficient? There is no test method for this one.

* Extra problems given in class:

myMult(x,y): Takes two non-negative (>=0) integers and returns *x***y*. You cannot use "*" !

myDiv(x,y): Takes two non-negative (>=0) integers and returns *x*/*y*. You cannot use "/ " !

myPi(n): Takes an integer n and returns a double that is Pi computed with *n* terms using the formula:

$$\pi = 4 * \sum_{k=1}^{\infty} \frac{(-1)^{k+1}}{2k-1}$$

$$= 4 * \left(\frac{1}{1} - \frac{1}{3} + \frac{1}{5} - \frac{1}{7} + ...\right)$$

Ex: myPi(1) ==> 4, myPi(2) ==> 2.666666…, myPi(3) ==> 3.4666666…, myPi(4) ==> 2.8952…