Lab 8 Methods and Functions

In object oriented languages such as Java, classes provide "methods" that enable you to manipulate data in the class. In Java there are only methods. That said, there really are two kinds of methods: those that belong to a class object (such as a method that returns the radius of a ball object, or, one that computes the mpg for a given car type), and those that are simply static … you give it something (without using an object) and it gives you something back. An example of the latter would be the Math.sqrt() method.

This is all unnecessarily very confusing. In the old days there were functions and procedures (as in the languages C and Pascal).

Think of it this way: sometimes you want to have a bunch of statements that do something. You may want to use them again and again. It is easier to package these up in a static method and just call the method. Again, think of the Math.sqrt method (or you can also call it a function). You called it several times in your earlier programs. It went off, did something and gave you a result. The actual code for this was quite long, but you didn't have to repetitively type the code in. THAT'S the beauty of static methods. The same is true for those that belong to classes. This lab assignment can be done using static methods. Here is an example of a static method that when passed an array of numbers, returns the smallest number found. It assumes for simplicity that the array is full and no numbers are smaller than zero.

```
static int smallest(int  ar[]) {

  int  i = 0;
  int currentsmallest = 0;

 for(i=0;i<ar.length; i++) {
   if (ar[i] < currentsmallest) currentsmallest = ar[i];
}

return currentsmallest;
}
```

Note: Java is really particular that the last thing you do in a method that returns something is the return statement. It gets horribly confused if you try to return something deep inside other statements. It's also good programming practice.

Now, the code above could just as easily be placed in-line if you only use it once. But, if you need to find the smallest integer quite often, then put it in a method and call the method.

Your job:

1. **Find the smallest Number.** Write a program that inputs three integers and passes them to a function (static method) that returns the smallest number.

2. **Prime Time.** Write a function that checks to see if a number is prime and returns either true or false (i.e. make your function a type bool and actually return a true or false value).
3. **Prime Factorization.** Write a function that will print out the prime factorization of any integer. You might find the work you did on the GCD useful and what you did on #2.
4. **Temp Changer.** Write two functions: one that converts from Fahrenheit to Celsius and one that converts from Celsius to Fahrenheit. Once you've got them working (and they should use doubles), try running in a loop and swap the calls back and forth. Start with one number, feed it to F then the output to C and back and forth within the loop a million or so times. Do you end up with the same number? Why?
5. **More rectangles!** Write a function that takes three arguments: width, height and a character and then draws a rectangle using the specified character. For example, makeRectangle(4,3,%) should draw:

   %%%%
   %%%%
   %%%%

   Put this inside a program that asks the user to input these three things, draws the shape and repeats until they enter -1 as a width.

6. **Reverser.** Write a function that takes an integer value of any length and returns the number with the digits reversed.
7. **Perfect Numbers.** An integer is said to be a perfect number if the sum of its divisors, including 1 (but not the number itself), is equal to the number. For example, 6 is a perfect number because 1 + 2 + 3 = 6. Write a function that checks to see if a number is a perfect number. Use this function in a program that determines and prints all the perfect numbers between 1 and 1000. Print the divisors of each number to confirm the number is indeed perfect!