Lab 1 - CS 111C

Purpose: This lab gives you more practice with recursion.

Lab: Write an overloaded method that displays the prime factors of a positive integer in **descending** order. For example,

Argument Value	Display
100	5 5 2 2
17	17
111	37 3
3960	11 5 3 3 2 2 2

Use the following method headers:

```
public static void showFactors(int number)
private static void showFactors(int number, int factor)
```

Java does not have default values for parameters. The public method *showFactors* should call the private method *showFactors*, with a value of 2 for the second argument.

```
public static void showFactors(int number)
{
        showFactors(number, 2);
}
```

The overloaded method effectively gives a default value of 2 for the second argument.

The private method *showFactors* that you write must be recursive. You will not receive any credit for this lab for a solution that does not use recursion.

Do **not** use any loops in the methods that you write. Do **not** use any arrays or private data members in the methods that you write. Write a driver program to test your methods. Submit your entire program, along with a capture of a sample run *that includes the values listed above*, on Insight. Remember to provide documentation for your methods, including purpose, preconditions and postconditions. Be sure to read and follow the guidelines in the Lab Grading handout.

Sample dialog (user input in bold):

Enter a number, and I will display its prime factors in descending order
(enter 0 to exit program): **100**
The factors for 100: 5 5 2 2

```
Enter a number, and I will display its prime factors in descending order
(enter 0 to exit program):  17
The factors for 17:  17

Enter a number, and I will display its prime factors in descending order
(enter 0 to exit program):  111
The factors for 111:  37  3

Enter a number, and I will display its prime factors in descending order
(enter 0 to exit program):  3960
The factors for 3960:  11  5  3  3  2  2  2

Enter a number, and I will display its prime factors in descending order
(enter 0 to exit program):  0
Goodbye!
```

Lab 1 hints

Base case: number is 1

Factor 100:

number	factor
100	2
50	2
25	2
25	3
25	4
25	5
5	5
1	

Is a factor: if remainder is zero (use the % operator)

Two different recursive calls:

```
    if remainder is zero
        try factor again
    else
        add one to factor
        try factor again
```

To get output in reverse order:

Look at writeBackward: which comes first, the recursive call or System.out.print?