

CS 172 - Lab 9

Recursive Methods

Submit **Lab9.java** on Canvas.

Programs that do not compile will
receive a grade of zero.

Style: Use header comments and inline comments as in previous labs. ***In addition: in each method, clearly comment each "Base Case", "Error Case", and "Recursive Step".***

Use indentation, spacing, and naming conventions as demonstrated by the instructor.

Summary of Chapter 12 Concepts

- Iteration is the term that refers to using a loop (such as a for or while) to solve a problem. An iterative algorithm uses a loop.
- Recursion is a programming technique in which a method calls itself.
- Any recursive definition must have a non-recursive part, called the base case. The base case permits the recursion to eventually end.
- Mathematical problems and formulas are often expressed recursively.
- For some problems, recursion is an elegant and appropriate way to solve the problem. For other problems, it can be inefficient and less intuitive than iteration.
- In general, recursive methods should be written with the code in this order:
 1. error case(s). If statements to check for conditions where this method does not or cannot work. Some recursive methods won't have an error case (for example, problem 3 below says "for any integer n"). Sometimes, the error case can be combined with the base case.
 2. base case(s). If statement(s) to check for conditions where this method works and this is the stopping point.
 3. recursive step(s). The method calls itself one or more times.

Factorial

The mathematical concept of factorial can be expressed recursively as $n! = n * (n-1)!$ This definition of factorial is valid for any $n \geq 1$.

Here's how you would write a method to calculate factorial:

```
public static int factorial ( int n ) {  
  
    // check for error case, factorial is not valid for zero or negative numbers  
    // however, we must return some integer... so arbitrarily choose -1  
    if ( n <= 0 )  
        return -1;  
  
    // base case - stopping point  
    if ( n == 1 )  
        return 1;  
  
    // now the hard part, recursive step  
    // find n factorial by multiplying n by n-1 factorial  
  
    return n * factorial(n - 1);  
  
}
```

Notice that the last return statement is not inside an if statement. This is important. The Java compiler must find at least one return statement in a method that will definitely be executed (not enclosed in an if, a switch, a loop, etc.).

Part A: Create Lab9.java.

Write the following methods as **recursive** methods in a file called Lab9.java. Methods that are written in a manner that doesn't use recursion at all or doesn't use recursion correctly will receive 0 points.

The header of each method is shown below the description of the method.

1. Write a recursive method called **printLettersForward** that will accept a character parameter. The method should print all of the letters of the alphabet up to (including) the parameter value. For example, if the parameter passed to the method is 'f', the method should print:

abcdef

The sequence of characters printed should be the same case as the parameter. If the parameter value is 'D' (uppercase), then the method should print:

ABCD

If the parameter value is not a letter of the alphabet, do not print anything. Do not return anything.

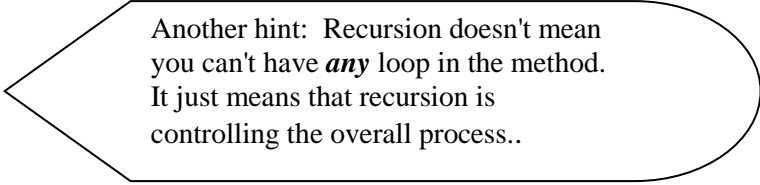
```
public static void printLettersForward(char c) {  
  
}
```

2. Copy the printLettersForward method and rename it printLettersBackwards. Change the method so that it prints the letters in reverse order. . If the parameter value is 'D' (uppercase), then the method should print:

DCBA

3. Write a recursive method called triangle that will display a triangle made of asterisks. The only parameter is an integer that determines the number of asterisks to print on the current line. Do not return anything. For example, if the test call to this method is triangle(5), the output should look like:

```
*  
**  
***  
****  
*****
```



Another hint: Recursion doesn't mean you can't have **any** loop in the method. It just means that recursion is controlling the overall process..

```
public static void triangle(int n) {  
  
}
```

4. Copy the triangle method and rename it upsideDownTriangle. Change the method so that the longest line of asterisks prints first. If the parameter value is 5, the method should print:

```
*****
****
***
**
*
```

5. Write a method called **powerOfTen** that will find and return 10^n for any integer n (n is the parameter). The method should return a double value.

Examples:

powerOfTen(3) should return 1000.0

powerOfTen(-3) should return 0.001

The method should not print anything.

```
public static double powerOfTen(int n) {

}
```

Part B: Create a main method within the same program.

Do not create a separate test program. Put a main method in Lab9.java. Add calls to the main to test each method as many times as needed to insure that all cases are working correctly.