Lab 8 – Recursion

Computer Science 2334

Due by: Friday, 21 April 2017, 11:59 pm

This lab is an individual exercise. Students must complete this assignment on their own.

Objectives:

1. To understand the concept of recursion and how it can be used to solve complex problems in a straightforward manner.
2. To understand the relationships between recursion and iteration and, correspondingly, the relationships between recursive and iterative code.
3. To demonstrate this knowledge by completing a series of exercises.

Instructions:

This lab exercise requires a laptop with an Internet connection. Once you have completed the exercises in this document, you will submit it for grading.

Make sure you read this lab description and look at all of the source code posted on the class website for this lab exercise before you begin working.

Assignment:

Many problems in computer science require relatively complex code that is difficult to write and even more difficult to debug. However, we can solve many of these problems in elegant and simple fashion by using the concept of recursion. Direct recursion occurs when a method calls itself to accomplish some task, instead of calling another method to do that work or doing the work iteratively. Methods can use recursion indirectly by calling each other in a recursive loop but for now we will just concern ourselves with direct recursive methods. In this lab, we will look at two recursive programs. One computes a mathematical function. The other explores a relationship graph. We will analyze these programs as part of today's lab.

The alternative to recursion is iteration. With iteration, repetition is accomplished with loops such as `while` and `for` loops, rather than through recursive method calls. While recursive solutions are often straightforward for some problems, the overhead of numerous recursive method calls often makes iterative solutions to those problems more efficient in terms or time or space (or both). The other portion of today's lab will be converting the two recursive programs mentioned above to iterative equivalents.

1. Download the `Lab8-eclipse.zip` project archive from the class website. Import the project into your Eclipse workspace using the slides from Lab 2.

2. Review the source code for **RecursiveExample1**. What is the name of the recursive method found there?

3. Try out the program using several different values for the integer pairs entered. Fill in the following table with the values you used and the values returned.

Integer 1	**Integer 2**	**Returned Value**

4. Describe the function that is being calculated by this recursive method.

5. Make a new class called **IterativeExample1**. Copy all of the code from **RecursiveExample1** into your new class. Change the recursive method from **RecursiveExample1** into an iterative method in **IterativeExample1**. You may add any additional data structures, variables, methods, classes, standard Java libraries, etc., as needed to make this method work. However, you may not change the signature of this method and you must ensure that the work of this method is done iteratively, not recursively. In addition, you may not change the `main` method. Ensure that there are no warnings generated for your code. **Do not suppress warnings.** Fix your code so that warnings are not necessary. Submit your code for **IterativeExample1** to Mimir for grading.

6. Review the source code for **RecursiveExample2**. What is the name of the recursive method found there?

7. Try out the program using several different values for the string, integer pairs entered. Fill in the following table with the values you used and the text printed. (Use names found in the assignment and small values for the integers.)

String	**Integer**	**Text Printed**

8. Describe what is being printed by this recursive method.

9. Make a new class called **IterativeExample2**. Copy all of the code from **RecursiveExample2** into your new class. Change the recursive method from **RecursiveExample2** into an iterative method in **IterativeExample2**. You may add any additional data structures, variables, methods, classes, standard Java libraries, etc., as needed to make this method work. However, you may not change the signature of this method and you must ensure that the work of this method is done iteratively, not recursively. In addition, you may not change the `main` method or the **Friend** class. Ensure that there are no warnings generated for your code. **Do not suppress warnings.** Fix your code so that warnings are not necessary. Submit your code for **IterativeExample1** to Mimir for grading.

10. Submit a completed electronic copy of this handout to D2L for grading.

All parts of this assignment are due by **Friday, 21 April 2017, at 11:59 pm**.