CS 211 – Programming Practicum Lab Exercise 12 Fall 2015

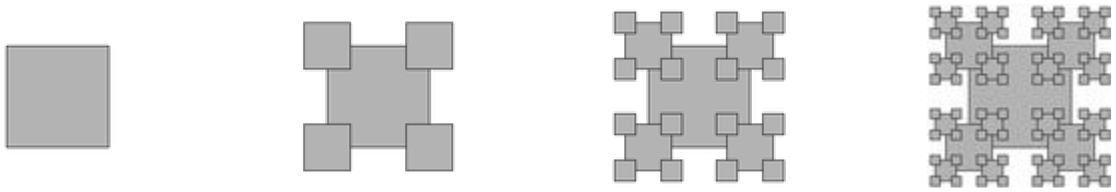
Name: ______________________________ UIN: ______________________

This lab uses links to web pages found at: https://www.cs.uic.edu/bin/view/CS211/LabexercisesF15

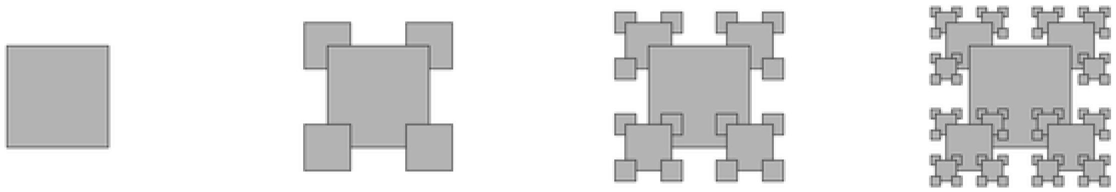
One fun thing to do with recursion is to create fractal-like drawings. Check out the code in Htree.java or Tree.java. Once compiled it will take an integer value as its command line argument do determine how many level of recursion is should do. Both programs rely on the code in StdDraw.java to function.

These programs are taken from http://introcs.cs.princeton.edu/java/23recursion/ by Sedgewick and Wayne.

The code in RecursiveSquares.java will produce one of the following images when the command line argument of 1, 2, 3, or 4 (respectively) is given.



1. What would the code for the draw( ) method from RecursiveSquares.java need to be changed to if the sequence of shapes were to become instead :



Testing is when we determine whether a program executes correctly. I.E. Does it perform as described in the program specification? (Note that debugging is trying to find the causes of failed tests and how to correct the issues. Debugging is different from testing.) In this lab, we will focus on testing. Consider the following Specification for an application:

Program Specification for a Simple Tax Calculation:

Create a web form that will allow the user to calculate the amount of taxes owed to the government. The user should enter the income amount in an input field, then press/click a button and the amount of taxes owed is to be displayed. If the user enters a non-numeric income amount or a negative income amount, display an error message telling the user that a positive numeric value must be entered. The amount of taxes owed is determined by:

- *If the income amount is $5,000 or less, the tax amount is 10% of the income amount.*
- *If the income amount is more than $5,000 and is $50,000 or less, the tax amount is 15% of the income amount.*
- *If the income amount is more than $50,000, the tax amount is 20% of the income amount.*

You will find 3 attempts at completing the above specification at:

- Attempt 1: http://tigger.uic.edu/~troy/tax1.html
- Attempt 2: http://tigger.uic.edu/~troy/tax2.html
- Attempt 3: http://tigger.uic.edu/~troy/tax3.html

For this lab, **create and run** multiple test cases **to fully** test the three attempts at the above specification.

Each test case should specify a specific input value, the expected result, whether the test case is testing an Equivalence Class or a Boundary Case, and the result (pass or fail) for each of the three attempts. A table is provided below for you to fill in. At least 8 test cases will be needed (at least 4 Equivalence Class tests and 4 Boundary Case tests – the first few have been started for you). **These test cases MUST thoroughly test the entire specification, so you may need more than 8 test cases.** (Hint: you will need more than 8 test cases to properly do this but it can be done in 16 or less test cases.)

What are Test Cases and What Kinds of Testing Cases Exist?

Test cases describe the actions/inputs to be given and the expected results. Multiple test cases are needed for each action/input. These test cases should include inputs which result in all possible error messages generated by the program. For this lab, we will only concern ourselves with two different kinds of Test Cases: **Equivalence Class** Cases and **Boundary Cases**.

Test cases should include inputs from all possible ranges of inputs (including error inputs). Each of these possible ranges of input is often called an **Equivalence Class**. For the above specification, one Equivalence Class is the values greater than $5,000 up to and including $50,000. Another Equivalence Class is the non-numeric values. For each Equivalence Class, a test case using a value from the "middle" of the range of inputs is to be created.

Test cases should also include inputs that verify the proper functioning of the program for values at the boundary between two Equivalence Classes. These are often called **Boundary Cases**. Each boundary should generate multiple test cases: a test for the boundary value, and a test for each value on either side of the boundary value. For the above specification, one boundary case is the value of $5,000. So we should have a test case that uses the value of $5,000, another with the value of $4,999 and one more with the value of $5,001.

Test #	**Exact Input Value for Test**	**Exact Expected Result**	**Equivalence or Boundary**	**Result from Attempt 1 (pass/fail)**	**Result from Attempt 2 (pass/fail)**	**Result from Attempt 3 (pass/fail)**
1	$30000	$4500	Equivalence 15% Tax Amount			
2	text	Descriptive Error Message	Equivalence non-numeric input			
3	$4999	$499.90	Boundary Income of $5000 10% vs 15% Tax			
4	$5000	$500.00	Boundary Income of $5000 10% vs 15% Tax			
5	$5001	$750.15	Boundary Income of $5000 10% vs 15% Tax			
6						
7						
8						