Introduction to Computer Science II
Lab 1
Printing patterns

Our review will properly begin in classes on Wednesday, but we will start our work with *nested loops* (covered in Section 4.7 of the textbook), all while getting familiar (or reaquainted) with the college's servers, the programming environment, etc.

1 Nested loops and 2D patterns

The command-line console provides a nice two-dimensional grid of characters onto which we can print (arguably) pretty patterns. The most natural expression of a such 2D patterns in through loops, somtimes in sequence and sometimes nested within one another. This assignment will require you to figure out how to express increasingly complex geometric patterns using loops in this manner.

To be specific, you will obtain a pre-made program that can already print an empty square of some size:

```
$ java PrintPattern square 6
++++++
+    +
+    +
+    +
+    +
++++++
```

Yes, yes, it looks more like a (non-square) rectangle than a proper square, but it is 6 characters to a size. That is, it's 6 rows by 6 columns, and so we will declare that a *square*.

Your program, when you've completed it, must also print an isosceles *triangle*...

```
$ java PrintPattern triangle 6
+
++
+ +
+  +
+   +
++++++
```

. . . and a *tree*. . .

```
$ java PrintPattern tree 6
     +
    +++
   +++++
  +++++++
 +++++++++
+++++++++++
     |
```

Finally, as an **optional** challenge, you can print Sierpinski's Gasket. How, exactly, that should look depends on how you choose to calculate and print the pattern. It's an extra challenge, so you figure out what it should look like! (And no, I won't be grading it.)

2 Getting started

The following steps will get you started with this lab. For those who have not programmed in this environment before (i.e., those who skipped COSC-111), feel free to ask questions about the details of getting rolling here.

1. **Login to your workstation:** Our lab is full of basic Windows desktop computers. These are run by our Information Technology department, and you must begin by logging into them using your college username and password.

2. **Login to a server:** The computer systems that we will use for our projects are `romulus.amherst.edu` or `remus.amherst.edu`, (henceforth, `remus/romulus`), which are UNIX (Linux) systems. To use these systems, you must login to them from your workstation using *Remote Desktop*, software that allows you to connect graphically to these servers. To do so, follow Remote Desktop Connection instructions that describe not only how to use this software on the Windows machines in Seeley Mudd 014, but also how to install (if needed) and use this software on your own computer, whether Windows, Mac, or Linux.

   Once you have logged into `remus/romulus`, you will see the graphical interface for that server, which looks very much like the desktop that your own computer presents. Right-click anywhere in the empty part of the desktop (not on an icon), and a menu will pop up. In that menu, select *Open in Terminal*. You will then see a *terminal window* within which there is a *shell*—a prompt at which you can type commands to the system. The shell is the place from which you will direct the system to run the program that allows you to edit your source code, to perform the compilation of your source code, and to execute your programs.

3. **Make a directory:** When you first login, you will be working in your *home directory*—the UNIX analog of your *My Documents* folder. Within this directory, you should make a *subdirectory* (a folder) for your work for this lab by using the `mkdir` (*make directory*)

command (shown below). Once you do that, you can `cd` (*change directory*) into that new folder. Using those commands looks like the following, where the dollar sign (`$`) is the *prompt*—what the shell prints before stopping to wait for you to type a command:

```
$ mkdir lab-1
$ cd lab-1
```

4. **Get the starting source code:** Use the following command to obtain a Java source code file that provides some starting structure to your work for this assignment. Be careful to include the tilde (`~`) before my username and the trailing space followed by a period (`.`):

   ```
   $ cp ~sfkaplan/public/COSC-112/lab-1/PrintPattern.java .
   ```

   [If you are doing this work on a different computer (e.g., your laptop), you can also grab this source code by clicking here.]

   To ensure that you have copied the file into your `lab-1` subdirectory, use the `ls` (*list directory*) command to list the files in the current directory, noting that the character following the dash (`-`) is a lowercase letter `L`, and **not** the numeral `1`, making the `-l` part of the command mean, *list directory using the long format*:

   ```
   $ ls -l
   ```

5. **Examine and modify the source code:** Run *Emacs*, a programming text editor, to examine the `PrintPattern.java` file. In the following command, be sure to include the trailing ampersand (`&`), causing the text editor to run in the *background*—that is, to run while allowing you to enter more commands:

   ```
   $ emacs PrintPattern.java &
   ```

   Emacs is a complex text editor that can do a great deal. To learn more about how to use it, you should read this documentation/tutorial on using Emacs. Also note that you are **welcome to use a different editor or an IDE**; so long as your submitted code runs correctly, I don't care what tools you used to write it.

6. **Compile:** Now that you have changed the source code, you must translate it into a form that the computer can execute. Leaving your *Emacs* window open, click over to your terminal window again. In it, use the following command to compile your source code:

   ```
   $ javac PrintPattern.java
   ```

In this case, **no news is good news**. That is, if the computer simply presents the shell prompt to you after you issue this command, then **the compilation succeeded.** The compiler—the `javac` program—will print messages into your terminal window only if it was unable to translate your program.

If you see such an error message, then you must have made some type of mistake in adding your line of code to print one more line of text. Go back to your *Emacs* window and see if you can spot your error. If you can, correct it, save the source code, go back to your shell window, and issue the compilation command (as above) again. If the error persists, or if you could not see what your error was in the first place, then **ask for help**.

7. **Execute:** Once you have successfully compiled your program, it is time to run it and see what happens. Go to your shell window and issue this command:

```
$ java PrintPattern square 8
```

Your program should (very quickly) print the square pattern in your terminal window.

3 Your assignment

Notice that, in the code you have copied from me, `printTriangle()`, `printTree()`, and `printSierpinski()` print a short message that their respective patterns have not yet been implemented. *Fill in the bodies of these methods so that each produces the correct pattern.* Remember that **Sierpinski's Gasket is optional**.

Be sure to test your code so that it works for a range of (reasonable) sizes.

4 How to submit your work

Submit your `PrintPattern.java` file. You may use either of the following two methods to use the CS submission system:

- **Web-based:** Visit the submission system web page. If you did your work on your own computer, open this link in your computer's browser; if you did your work on `remus/romulus`, then open this link in a browser opened within *remote desktop.*
- **Command-line based:** Use the `cssubmit` command at your shell prompt. This command works only while connected to `remus/romulus`.

This assignment is due on Sunday, Jan-29, 11:59 pm.