COMPSCI 105: Principles of Computer Science
Summer 2007

Tutorial Four: Recursion

Helen Gu and Slobodan Vukanovic
{ygu029, svuk002}@ec.auckland.ac.nz

This tutorial is not being assessed. It provides you with an opportunity to become familiar with concepts introduced in lectures.

Question One: A Recursive Sum

What is the output after the `main ()` method is called?

```
public static void main (String[] args) {
   System.out.println (sumFoo (5));
}

public static int sumFoo (int n) {
   System.out.println (n);
   if (n < 0)
      return n;
   int a = sumFoo (n-7);
   int b = sumFoo (n-5);
   return a + b;
}
```

The output is:

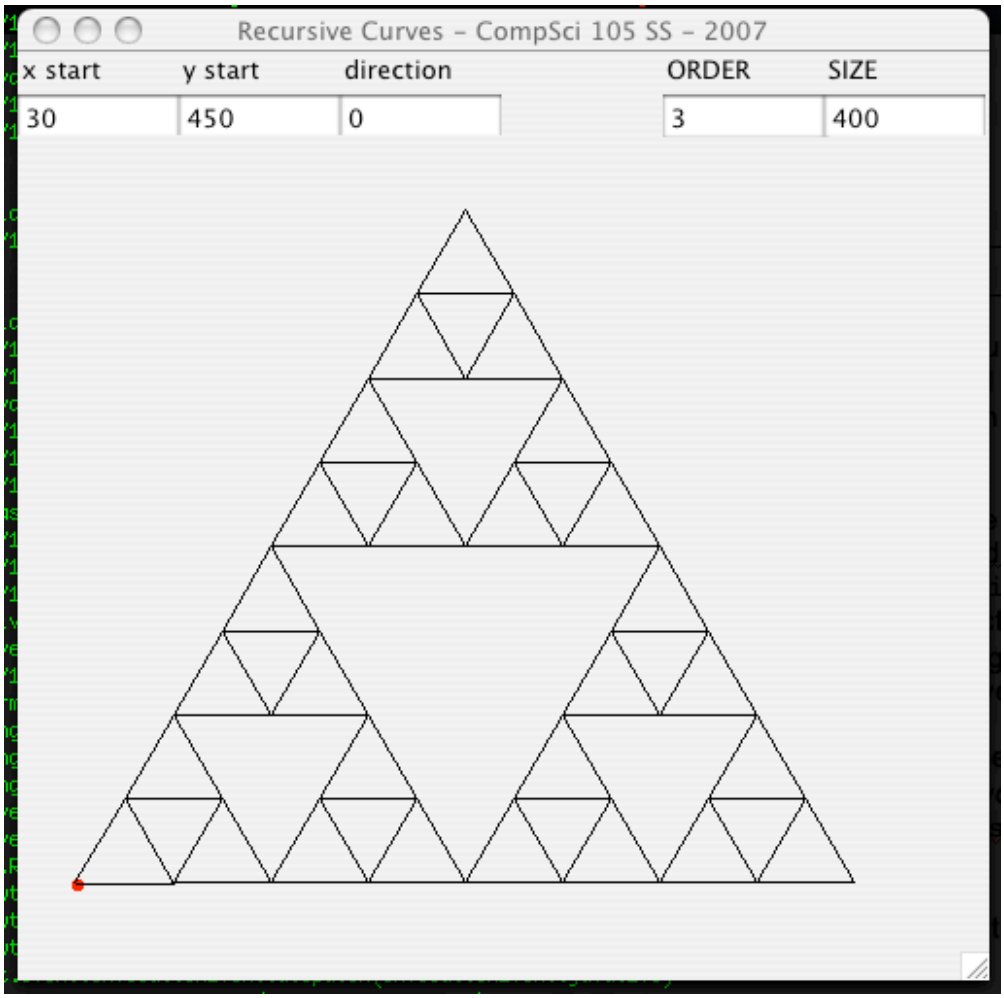
Question Two: Recursive Curves

Download `Curves.zip` from the course website (you will find it under ``Tutorials''). It contains two files written by Paul Denny: a turtle class `Turtle.java`, and `RecursiveCurves.java`. `RecursiveCurves.java` uses the turtle class to draw curves in a window when the application is started.

Your task is to fill in the methods for drawing the C--curve and the Sierpinsky triangle. Everything else is handled by the rest of the code, which you are not required to modify. The methods you need to fill are `void c (int order, double size)` and `void sierp (int order, double size)`. Have a look at Andrew Luxton--Reilly's lectures on uses of recursion to find out the details about these curves. A method for drawing the Koch curve is given. You can use it to see how the application should perform on your curves.

In order to run the curve--drawing application, you need to compile `RecursiveCurves.java`, and pass an argument to the application specifying which curve you want to draw. For example, to draw the Koch curve, you would run `java RecursiveCurves koch`. `c` and `sierp` are the other options.

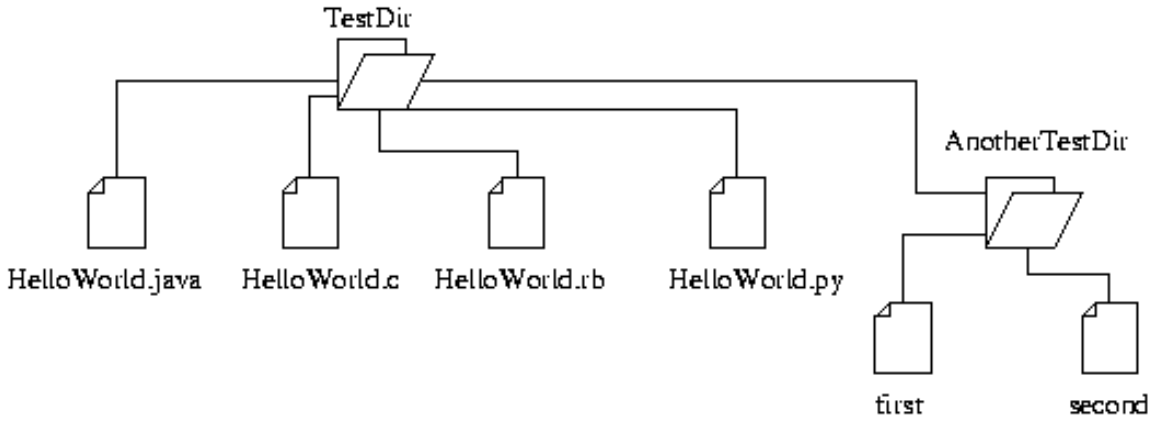
The screenshot shows the curve--drawing application ran with the `sierp` switch.



Question Three: Recursive Directory Information

In the last tutorial, you were asked to write certain information about a directory to a file called `dinf`. Your task now is to modify the answer to that question in order to store information about directories inside directories.

Given a directory, the program must generate a `dinf` file for it (refer to Tutorial Three for details). If any of its entries are directories, the program must recursively descend down those directories and generate a `dinf` file for each of them. Here is an example of a (small) directory structure:



When `java DirInfoRecursive TestDir` is called from the command--line, one `dinf` file should be placed inside `TestDir` containing information about the four `HelloWorld` files and `AnotherTestDir`, and another `dinf` file should be placed inside `AnotherTestDir`, containing information about `first` and `second`.

In order to accomplish this, download the file `DirInfoRecursive.java` from the course website (look under ``Tutorials''), and modify the `processDir ()` method to handle the recursive descent.

Notice that `processDir ()` has been changed slightly from the Tutorial Three version. It now takes a string representing the path of the directory it creates the `dinf` file for.

You can download `TestBig.zip` from the course website, which contains a bigger directory structure to try your code on.