Name: ThankGod Ofurum

Professor: Francois Neville

Class: CS 2322

Title: Data Structures: Recursion

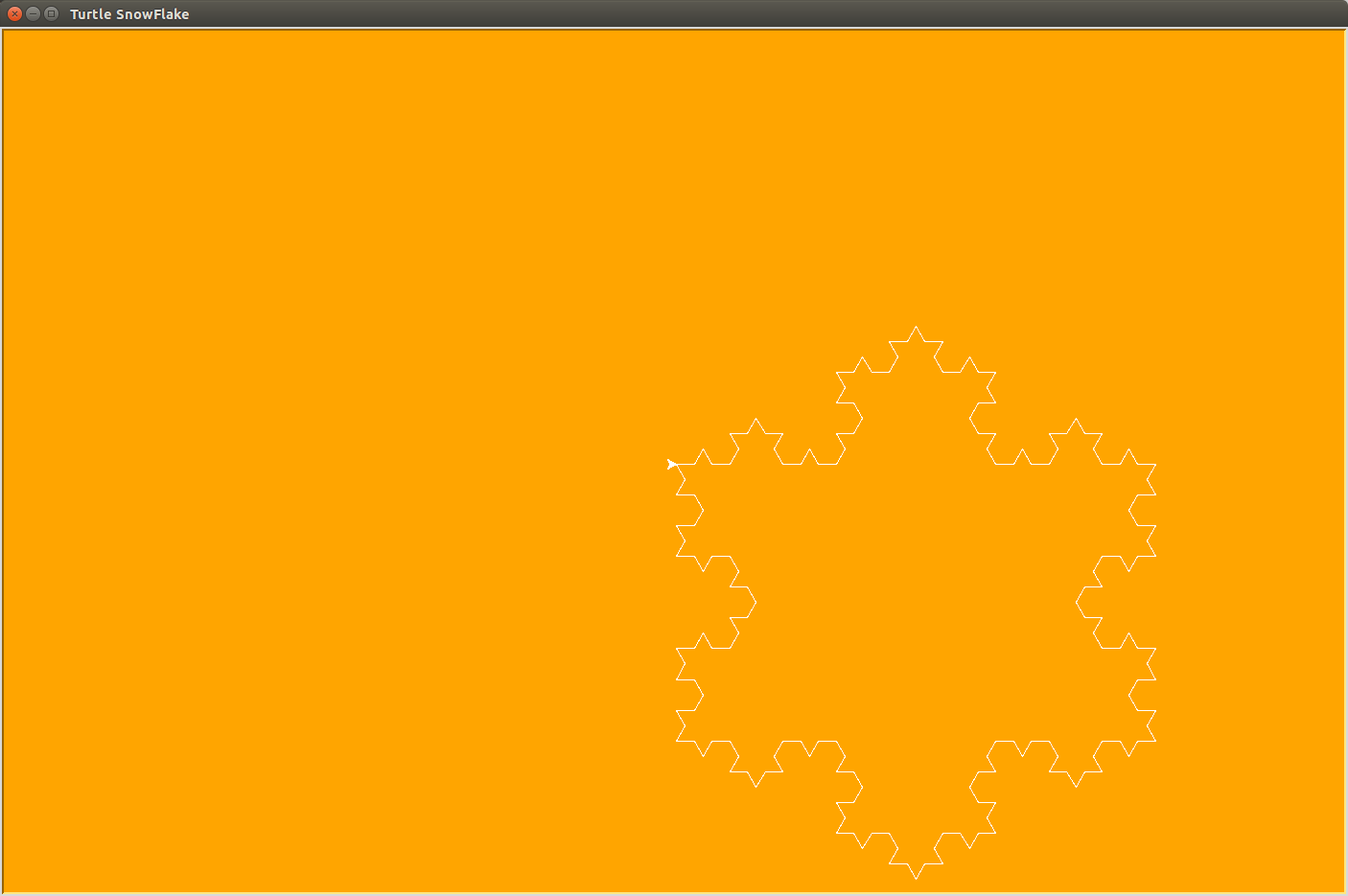
Questions

* **Lab 5: Recursion (Due 3/23)** 
  + **pg 183-184 #3, 5, 8  
    (#5 special instr.) Show series of average ItFib and RecFib timings in the same Excel chart. Explain your results.  
    In fact, you should write a paragraph (or a page) of explanation for each of the three projects above, including answers for any questions posed in the book's exercises.**

Answers

The iterative Fibonacci program has a Big-O running time that’s linear (O (n)), while the recursive Fibonacci program has a Big-O running time that’s exponential (O (2^n)). This is due to the fact that the recursive Fibonacci program calls itself repeatedly at a smaller scale for each value in the Fibonacci sequence in order to complete its task, and this get to be time consuming even when the desired number is small-a good example is recFib (6) has to call itself 15 times in order to arrive at its answer. While, the iterative Fibonacci simply stores values in variables while iterating through the Fibonacci sequence, and this save time as the variables are assigned new values for each iteration through the sequence. This, however, can also be time consuming especially when large numbers are used as the program is required to iterate through each number in the sequence before arriving at the desired number, but it’s faster than recursion. I decided to time the average amount of time needed to compute the recursive sequences of 10, 20 and 30, and both programs returned the results of 10 and 20 relatively quickly. However, when the value of 30 was needed, the recursive Fibonacci program was time consuming as it had to call itself multiple times, while the iterative Fibonacci was constant as it only had to iterate through 30 values.

The Koch snowflake is a fractal, which is basically a drawing which also has a self-similar structure. The Koch snowflake basically calls itself repeatedly at a smaller scale based on the given order and the resulting line structure of the previous order is multiplied by 4: an order of 0 draws a straight line, an order of 1 has 4 lines, and an order of 2 has 16 lines. The structure of the lines of the Koch snowflake is influenced by the angle at which turtle turns after each recursive function call, and in order to form the shape of an actual snowflake a triangle has to be drawn. In order to accomplish this I ran my Koch snowflake function in a for loop which iterates 3 time, and I used an order of 3 and a size of 200. After each execution of the function in the iteration the turtle is turned right by 120 in order to form a triangle with the three iterations.



The Fractal tree is also a fractal and consist of self-similar structure. The tree basically a repetition of one branch call but at a smaller scale each time, 2/3rd the size of the previous branch, and each branch of the first recursive case is turned right by 45 in order to form the shape of a tree branch. The first branch call serves as the trunk of the tree and the other branches form the branches and twigs of the tree. Important to note is the inclusion of the backward command and the left turn of 90 which allows the drawing of the sidewards shaped branches and different segments of branches in the tree. In my turtle program I decided to change the width of the branch based on which call of the branch was being drawn; the further the call, the smaller the width of the branch, thus reflecting a real tree image. I also choose to add green dots at the tip furthest branches of the tree as these form the leaves of the tree. I also moved the turtle backwards by 150 before even running the function in order to align the fractal tree with the screen.

