**Self-Evaluation for Lab 2 – Recursion**

|  |  |
| --- | --- |
| Your name: | David Oftedahl |
| Date: | 04/17/2022 |

Instructions  
After completing this evaluation you will submit this form along with screen shots of each of your applications running on your machine. Your source code will be submitted by providing the url for your github repository. You may make corrections to your work and submit an updated evaluation in your final version of the lab.

**General comments and notes:**

Overall I feel confident about what I’ve completed, however I’d like to find way to fix the triangle display I put together for the first problem, as well as build out a more intensive version of the tower of Hanoi problem that actually includes a screen representation of the disks being moved.

**One thing that you learned from completing the lab:**

I learned overall how to manage recursion. Initially I looked at recursion as being the more difficult control structure to utilize, however once I started to understand the basics and realizing I can have multiple recursion within the same call things became a bit easier to manage.

|  |  |
| --- | --- |
| ***Programming style for all programs*** | |
| Is proper indentation used? | Yes |
| Are comments used appropriately? | No – minimal comments being used to call out what each function is doing |
| Do variable names use camelCase or snake\_case? | camelCase |
| Do function/method names use camelCase or snake\_case? Do functions/methods pass parameters and return values appropriately? | camelCase |

|  |  |
| --- | --- |
| ***Professional development tools and techniques*** | |
| Created a private git repository for the lab? | Yes |
| Committed at least once for each problem? | Yes |
| Pushed to and pulled from the remote repository at regular intervals? | Yes – improving from the last lab, I submitted a few times (usually after completing each problem) and even utilized GIT to move from my main computer to my wife’s laptop |
| Url for github repo: <https://github.com/LCC-CIT-Programming-CS162P/recursion_lab-oftedahld.git> | |

|  |  |
| --- | --- |
| ***Triangular Numbers - Problem 1*** | |
| Completed triangular numbers using recursion?   * Defined a recursive function? Base case returns 1 when n = 1? Non- base case returns n + a recursive call, decreasing the parameter value by 1? * Created an application that displays the first 10 triangular numbers? * Screen shots of the program running correctly are included? | Utilized recursive function with base case returning 1 when 1.  Displays the numbers as well as visual representation for each number.  Screenshots included in repo. |

|  |  |
| --- | --- |
| ***Fibonacci Numbers – Problem 2*** | |
| Completed Fibonacci numbers using recursion?   * Defined a recursive function? Base case includes 0 and 1? Non- base case includes 2 recursive calls? * Created an application that displays the first 10 Fibonacci numbers? * Screen shots of the program running correctly are included? | Utilized recursive function with base case when 1 or 0 and 2 recursive calls for non-base case.  Application will display as many numbers as required per the user-input.  Screenshots included in repo. |

|  |  |
| --- | --- |
| ***Palindromes – Problem 3*** | |
| Completed palindromes using recursion?   * Defined a recursive function? Base case includes string length of 0 and 1? Non- base case includes recursive call with a “shorter” string? * Created an application that tells whether keyboard input is a palidrome? Effectively ignores whitespace, capitalization and punctuation? * Screen shots of the program running and producing correct results are included? | Base case implied and configured for anything < 2. Non-base case includes recursive call with a shortened version of the string.  Application affectively calculates palindrome and ignores all capitalization and non-alpha numberics.  Screenshots included in repo. |

|  |  |
| --- | --- |
| ***Tower of Hanoi – Problem 4*** | |
| Completed Tower of Hanoi using recursion?   * Defined a recursive function? Base case includes 1 disk? Non- base case includes 2 recursive calls? * Created an application that asks the user for the number of discs and displays the moves necessary to solve the puzzle? * Screen shots of the program running and producing correct results are included? | Base case includes 1 disk. Non-base case includes 2 recursive calls and a disk move.  User can provide number of disks to be solved against.  Screenshots included in repo. |