**CSE 212 – Programming with Data Structures**

**W03 Prove – Response Document**

|  |  |
| --- | --- |
| **Name:** | Philip Marvin |
| **Date:** | 5/8/21 |
| **Teacher:** | Kelly Tucker |

*It is a violation of BYU-Idaho Honor Code to post or share this document with others or to post it online. Storage into a personal and private repository (e.g. private GitHub repository, unshared Google Drive folder) is acceptable.*

**Question 1: From Part 1, describe what the Mystery Stack 1 code does and how the use of a stack helps in the implementation.**

This code’s design is to alter the given text by reversing its order using stack and popping each word in reverse order. The function mystery\_1(text) will output the result after the stack is empty.

**Question 2: From Part 1, what are the three outputs from the Mystery Stack 1 code for the following three different inputs?**

**These are all palindromes, so they will somewhat resemble the same word/passage forwards and backwards**

* **racecar: racecar**
* **stressed: stressed**
* **a nut for a jar of tuna: anut fo raj a rof tun a**

**Question 3: From Part 2, describe what the Mystery Stack 2 code does and how the use of a stack helps in the implementation.**

This code is using the stack to solve operations after a fix. The program stores the operands or numbers as float datatypes, and once an operator is found, the operands are popped to be used by the operators in each part. It’s repeated until the expression is finished, and when there is only one operand left in the stack, the function will return the value from the stack

**Question 4: From Part 2, answer the following regarding what the Mystery Stack 2 code does:**

* **What will the result be if the input parameter is: 5 3 7 + \*: 50.0**
* **What will the result be if the input parameter is: 6 2 + 5 3 - / : 4.0**
* **What input parameter would result in the display of “Invalid Case 1!”**
  + **Let’s say we put 1 4 \* /. The problem is there are insufficient numbers for the operators (we would need two more)**
* **What input parameter would result in the display of “Invalid Case 2!”**
  + **If we use an expression like 3 1 + 0 0 / /, The final operator is /, and dividing by 0 is 0, which evaluates as 0 from the operation stack which is the exception that the elif statement says results in “invalid case 2!”**
* **What input parameter would result in the display of “Invalid Case 3!”**
  + **If we use an expression like 2 2 - ~, there is an invalid character for an operator, which means the expression can’t be solved.**
* **What input parameter would result in the display of “Invalid Case 4!”**
  + **This is the opposite of case 1, so if we had something like 1 2 3 \*, there are more numbers than what the operator is designed to handle in the stack, as it would normally only apply \* to 1 and 2.**