**CSE 212 – Programming with Data Structures**

**W03 Prove – Response Document**

|  |  |
| --- | --- |
| **Name:** | Eli Hansen |
| **Date:** |  |
| **Teacher:** | Zachariah Alvey |

*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.**

The Mystery Stack code creates a new list object called stack that is a list of characters. The foreach loop iterates over the input text and for each character/letter in the text it pushes/adds the letter to the list of characters named Stack. While the list count is greater than zero the last character/letter is removed from the list and added to the result. The result is the characters in reverse order. This works because of the LIFO (last in first out) concept. We iterate over the text and add the letters/characters to our list of characters using push. We then use stack.Pop which removes the last character and adds it to the result and we keep doing that while there are still characters. This means the text characters are displayed in reverse order.

So if the text input were: the rain in Spain stays mainly in the plane

The end result would be: enalp eht ni ylniam syats niapS ni niar eht

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

* **racecar** racecar
* **stressed** desserts
* **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.**

**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
* **What will the result be if the input parameter is: 6 2 + 5 3 - /**
  + 4
* **What input would result in the display of “Invalid Case 1!”**
  + If there is an operator ‘+’, ‘-‘, ‘\*’, ‘/’ in the input, but the stack does not have at least two numbers available then the “Invalid Case1!” error will be displayed.
* **What input would result in the display of “Invalid Case 2!”**
  + If the operator is ‘/’ and the op2 is 0, then “Invalid Case 2!” would be displayed.
* **What input would result in the display of “Invalid Case 3!”**
  + When the input string is not a valid float or one of the listed operators this would trigger “Invalid Case 3!” to be displayed. Examples of this would be an alphanumeric string like 12xt34 or abc or a number with multiple decimal points 4.3.2
* **What input would result in the display of “Invalid Case 4!”**
  + When the stack count is not 1, so if the stack contained more than one in the count, “Invalid Case 4!” would be displayed.

An example would be: 5 3 + 2 which would result in the following in the stack after processing is done [8, 2], so the count is more than 1.