**CSE 212 – Programming with Data Structures**

**W03 Prove – Response Document**

|  |  |
| --- | --- |
| **Name:** | Jorge Alberto Chavez Ponce |
| **Date:** | 09/30/2023 |
| **Teacher:** | Ephraim Kunz |

*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 1 code reverses the entered text. It uses a stack data structure to achieve this in reverse.

First it starts an empty list called stack. Second it will loop through each character in the input text and add each character to the stack. This effectively pushes each character onto the stack, one by one.

After the loop, an empty string is created, followed by a while loop that continues if the stack length is greater than 0. Inside the while loop, pop (remove and return) the top element of the stack and return it adds to the result string. This effectively reverses the order of the characters. This causes the order of the letters in the word to be reversed. When the while loop ends, the result string contains the inverted version of the input text.

Using the stack helps in this code for these two things:

* Pushing characters onto the stack and then popping them reverses the removal order. The last character pushed to the stack becomes the first to appear. This reversal effect is essential for reversing the input chain.
* The stack allows an efficient way to reverse the string without directly manipulating the original string, which in many programming languages is an expensive operation because strings are immutable. The stack provides a flexible and efficient approach to handling the rollback process.

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

* **Racecar**

Output: “**racecaR”** (This word is a palindrome)

* **Stressed**

Output: “**dessertS”**

* **a nut for a jar of tuna**

Output: " **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.**

Mystery Stack 2 code is a simple calculator that evaluates arithmetic expressions given in postfix notation.

This is how the code works:

Start by initializing an empty stack. After that a for loop starts. The input text is divided into individual elements based on spacing. The code then iterates through these elements.

If an element is an operator (+, -, \* or /), it checks if there are at least 2 operands on the stack. Otherwise, it prints an error message and returns None. If there are enough operands, it pops the top two values from the stack, performs the corresponding operation, and returns the result to the stack. If an element is a float, the is\_float function converts the element to a float and pushes it onto the stack. If a 0 is found in the second operand and the operation that follows is a division (/), it will throw an error message and return None.

If an element is found and it is an empty string, it is ignored. If an element is not a valid float or operator, prints an error message and returns None. After processing all elements, it checks if exactly one value remains on the stack. Otherwise, indicates an invalid expression and returns None. Otherwise, it returns the single value left on the stack, which is the result of the arithmetic expression.

If the input is valid and the operations are successful, the result is returned, which is the only element left on the stack.

The stack is used to keep track of operands. When an operator is encountered, the code pops the top two operands from the stack, performs the operation, and pushes the result back onto the stack. This way, the code evaluates the expressions in the correct order, following the rules of arithmetic operations. The stack allows the code to handle complex expressions by storing intermediate results until they are needed for further calculations. This implementation ensures that the expressions are evaluated based on the correct operator precedence and associativity.

**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 + \***

The code interprets this postfix expression as: (3+7) \* 5

The stack is:

[5.0]

[5.0,3.0]

[5.0,3.0,7.0] => operator + => op2 = 7 y op1 = 3 res = [10]

[5.0,10.0] => operator \* => op2 = 5 y op1 = 10 res = [50]

[50.0]

So, the result is: **50.0**

* **What will the result be if the input parameter is: 6 2 + 5 3 - /**

The code interprets this postfix expression as: (6+2)/(5-3)

The stack is:

[6.0]

[6.0,2.0] => operator + => op2 = 2 y op1 = 6 res = [8]

[8.0]

[8.0,5.0]

[8.0,5.0,3.0] => operator - => op2 = 3 y op1 = 5 res = [2]

[8.0, 2.0] => operator / => op2 = 2 y op1 = 8 res = [4]

[4.0]

So, the result is: **4.0**

* **What input parameter would result in the display of “Invalid Case 1!”**

If an operator (+, -, \*, /) is encountered without at least 2 operands on the stack, it displays "Invalid Case 1!" and returns “N**one”**

For example, the next parameter: **5 +**

* **What input parameter would result in the display of “Invalid Case 2!”**

If a division operation is encountered with the second operand (op2) being 0, it displays "Invalid Case 2!" and returns “N**one”**

For example, the next parameter: **5 0 /**

* **What input parameter would result in the display of “Invalid Case 3!”**

If a non-numeric and non-operator item is encountered, it displays "Invalid Case 3!" and returns “N**one”**

For example, the next parameter: **5 a b c +**

* **What input parameter would result in the display of “Invalid Case 4!”**

If there is more than one value left on the stack after processing all items, it displays "Invalid Case 4!" and returns “N**one”**

For example, the next parameter: **5 3 + 5**