Ethan Coyle

Dr. Saikat – CMPS 4143

Class- Contemporary Programming Java/Python

Program 5 Problem Solving Steps

1.

```
#   A stack elements structure has following functionalities like empty()#
#   , StackSize(), top(), push() and pop(). You need to implement the    #
#   stack with Linkedlist elements structure in python. Implement the    #
#   stack means, all of the stack functionalities including the          #
#   construction of stacks should present on your code.                  #
```

Step 1: **State the required output:**

The items that are necessary for the output of this problem are the result of the linked

List and any other output we would like to display to the user that pertains to adding,removing, displaying the list or displaying the top of the list. All of these will

Have a function method that will be called whenever the user would like to pop,push,display thelist,display the top or doing any adjustments to the list itself inside of the program.

Step 2: State the required input:

The required input for this problem is the input values in which the user would like to push to the stack. Without the elements being pushed to the stack, our stack is empty and will be pointing to none.If the user would like to get a better visual on the concepts of a working stack as a linked list, then they must push values onto the stack which will then be added with another class called node in which the node will be added and the order of the link list is updated accordingly

Step 3 : State the Formulas :

There really aren't any formulas for this problem. However, To properly visualize a link list using stack class, we , must utilize function methods inside of each on of them such as the pop,push,display,printtop etc and inside of the class node, we will be updating the link list whenever the user pushes or pops a value insie of the stack

Step 4: Steps in Order:

1. We must create our class Node which will have a head pointing to none and then we need to utilize how to define the elements and how they will be updated whenever the user removes or adds a value. We must then create a class called stack in which there will be methods

size,empty,pop,push, iterate through in which the user is able to manipulate the elements inside of the stack and display the list or display the head and change the size of the stack accordingly if an item is added or removed and then an iterator in which the user is able to visualize the way the list looks from the head pointer to the null pointer. In side of my program, I pushed multiple values to the stack, displayed the stack and then displayed the top. Then after that, popped a few of the values and then displayed the top value and then the list after the update was completed.

Problem 2:

```
#    Given the expression as string [str] , find the duplicate           #
#    parenthesis from the expression. Your program will output whether   #
#    or not finding the duplicates, that is true or false                #
# Format style for program -                                             #
#       Input type: An expression in string;                             #
#       Output type: A boolean value True or False                       #
```

Step 1: the necessary output for the program.

The necessary output for this problem is the Boolean value after checking if there is duplicate parenthesis or not inside of the test string. If there are duplicate parentheseis, then the flag is changed to true which will be the Boolean output and if there are not duplicate parenthesis surrounding the user input expression, then the flag is False which is the Boolean output that the user is going to see inside of the output for the string they are testing for.

Step 2: State the necessary input for this problem.

The necessary input for this problem is the user to enter in a string expression. Inside of the program, the user will be prompted to enter in a string expression and then after entering the string expression, we must iterate through character by character until the end of the string until the end and they see if any of the left parenthesis are duplicates of the right parenthesis and if they are, our flag Boolean value will be updated accordingly for the proper output test

Step 3: Formulas necessary:

The formulas that we must utilize in this problem are resading the entire string expression and appending the values and then checking character by character to check to see if there are duplicate parenthesis and then update the output according to if we find matches or not

Step 4: Steps in order:

First, the user will be prompted how many test cases that they want to try and then, a prompt will pop up to enter the expression they would like to check. After the user enters in the

test expression, the program will parse through the string expression in search of duplicate parenthesis. The Boolean will be updated according to if there are duplicates or not. Then, for user easement, a print out of the users entered test expression is displayed to what they enter and then right underneath that, the user will see what the Boolean flag displays True or False. Then we wll update the counter to count until the end of how many test expressions that user wants to try and then the program will terminate after all those test expressions have been done and fully satisfied.

Problem 3:

```
#     Write a python code to find the average from a stream. The         #
#     input of this program will receive a stream of numbers and a       #
#     window size to find the moving average of all the numbers in the   #
#     sliding window. Write your code in OOP style and solve the program #
#     with queue and or stack data structure.                            #
```

Step 1: State the necessary output :

The necessary output for this problem will be displaying the window size that the user entered, the stream values that the user entered and the average list of the stream values after being calculated.

Step 2: State the necessary input:

The necessary input from the user will be the prompt that asks the user to enter the window size and then after that, a prompt will prompt the user to enter a list or numbers separated by a comma which will then be stored in a list and parsed through to calculate the average of each one according to the value right before that and then divided by the total number to find the average, which is passed back as a float because the value is an average, we must use float.

Step 3 List the formulas:

The formulas that we will be utilizing inside of this problem will be the sum formula which will be utilized after the user enters the stream values and then after the sum of the values plus the values before which are appendied to the list, we pass that value back and then divide by the number of stream values in the list so far. Aside from this formula, There is really no other formula that we must utilize aside from creating a queue class that iterates through the users input values as a list and then acts accordingly to what we are doing inside of the main program to find the average of the values in the list one by one.

Step 4: Steps in order

First, We must create either a stack or a queue, I chose to go with a queue class because in the first problem, I utilized a stack so I wanted to use a queue. Inside of the Queue class we will perform similarly to the stack with the push and pop but we utilize the enqueue and deque methods which will act as our push and pop and then we declare a method that will return the sum to the other class average in which we will compute the average of each element according to the values that came before it. Inside of the Average calculated Class, we will be prompting the user to enter the size of the window and then after this, we will prompt the user enter the values of the stream which will be stored as a list and then the queue class is called fore every element, it will iterate through all of the numbers the user entered and then go one by one until the end of the read in string and then compute the average. Once the whole list has been iterated through, we will display to the user the final result of the average for each of the numbers they entered inside of a list with a floating point value. After this, if the user would like to enter more inside to test, they have the ability to do so as well if they so choose. If not, then the user will get a visual of what their input is and then the display of the appropriate output of the averages of each stream .