CPSC 131- Data Structure : Project 3

Introduction

In this project you will design and implement stack and queues operations for the web browser. You will be provided with a starter template with all the test cases. You need to create a stack and queues from scratch and need to add functions that are specified in the document. Do not change the main function, I have added test cases in it.

Part-A (Stack Operations)

You are hired as a developer in a multinational company and you have a project to make an application that checks whether coding syntax is valid or invalid by checking all the parentheses. A valid syntax represents the proper order of opening and closing parenthesis and brackets (‘()’,’[]’,’{}’) with equal number of them. Addition to that, the project owner told you to find out the index where error (unbalanced parenthesis is present) in the given code occurs. Also, you have to find out how many parentheses are needed to make syntax valid if it's invalid. Use vectors to implement stacks. Create following functions in Stack class:

1) validPara() - It will determine whether all parenthesis are completed or something is missing. It will print valid if all are completed otherwise it will print invalid. It will take one string parameter that is an expression.

   Ex: 1) ‘(({}))’ -here all parenthesis are balanced so, its valid

   2)’(({})’ - here aone parenthesis is missing so, it is invalid

2) indexError() - It will find out the first index where the actual error occurred which is invalid parentheses. It will take one string parameter that is an expression. It will return the first index of unbalanced parenthesis. It means you have to find out first opening parenthesis for which closing parenthesis is missing Return -1 if all parenthesis are completed.

   Ex: 1) ‘(({}))’- return -1 since all are balanced

   2) ‘(({})’ - return 5 since index at 5 for opening parenthesis closing parentheses is missing

3) minPara()- It will determine how many parenthesis are needed to make a string of code as a valid parenthesis string. It will take one string parameter that is an expression. It will print how many minimum parentheses are required to make the string balanced.

   Ex: 1) ‘(({}))’- return -0 since all are balanced

   2) ‘(({})’ - return 1 since index at 5 closing parentheses is missing

4) scorePara()- It will calculate how many valid parentheses are present in the given string. It will take one string parameter that is an expression. It will print how many balanced parentheses are present in the string.

Ex: 1) ‘(({}))’- return -3

2) ‘(({})’ - return 2 since index at 5 closing parentheses is missing

Part-B (Queue Operation)

The company wants you to make one message buffering system for their support team. The application divides the entire message in chunks of 8 characters and adds it into the support team’s buffer. The buffer implements queue data structure. To make secure search the characters in a single packet are reversed. When the message is processed at that time the characters are again reversed to get the right message. Use a vector for queue operations. Write following functions in Queue class:

1) enqueue() - It will divide the entire message into chunks (new string) of 8 characters and insert it into the queue. Before insertion it will reverse the character of the chunk using the reverse function. It will take a string as a parameter.
2) processMsg() - It will remove the chunk from the queue one by one until the queue is not empty. Then it will be reversed and the entire message will be formed. Display the entire message in this function.