Quan Yuan
Dec 4, 2022
Project 9 report

This project is about stack and queue. There 2 tasks in this project. This report is structured by tasks.

Task 1- check valid brackets:

This task requires us to take an input string from the client, and check if the brackets in the string is valid. By valid, it means that the brackets are paid and with correct order. Any extra single bracket or reverse of the order will cause it to be invalid.

The implement is realized via stack. We read character by character from the string.
- If the character is an opening bracket, then push it into the stack for later check.
- If the character is a closing bracket, then we pop the last opening bracket we pushed to see if it matches with the current closing bracket. If there is no previous opening bracket, the string is invalid.
- If the character is neither an opening bracket nor a closing bracket, we could just ignore it and continue to next character.

After we read in all characters, we need a final check to see whether the current stack is empty. If it is not empty, it means there are some extra opening brackets. Therefore, the string is invalid.

If passed all previous checks, the string is valid, and hence the function could return true.

By using this flow, an empty string will skip all the checks except the last one. And it will pass the last check to make itself a valid string. No further codes are needed to eliminate the empty string situation, unless it requires the empty string to be an invalid string.

Task 2 - mimic a ticket line service:

this task requires us to mimic a ticket line where people are being served one by one, and more people are joining the back of the line. The overall flow is to iterate for 100 times. In each time

random decide whether 0,1,or 2 people join the line, and serve only 1 people. At the end, decide how many people are still in the line.

The implement is realized via queue, with 2 helper functions.

The queue is to store every customer's number, starting from 0. So we need to keep track on how many customers are there. And then enqueue the current customer amount for each new customer as his customer number.

So the helper function to realize the new customer joining the line will take 2 parameters. One is the queue of customers, and other is the total amount of customers entered into the line. Ignoring whether or not they have been served.

Then we generate a random number from 0 to 2, to pass in the new customers. This function will return the latest amount of customers entered into the line.

Every time the top of the customer is served, we need to print a message that we have served customer XX. And at the end, we need to decide how many customers are still in the queue. Because the size of the queue is not accessible, we need to keep track of the total amount of customer we served. Therefore the parameters of this helper function are a line of customers, and the total amount of customers we served.

Obviously, if the line is empty, we have no one to serve. Hence we just return the total amount of customers we served without any changes. If the line is not empty, we need to dequeue from the top of the queue, and print the message that we have served customer XX. Then we increment the total amount of customers we served by 1. And return the total amount of customers we served.

Finally, we calculate the difference of total amount of customers entered the line, and total amount of customers we served, to get the customers still in the line.

Reflection:

One interesting thing is that, the size of the queue and stack is very useful when performing the tests. Therefore, I designed the size() function only for tests using.

Acknowledgment:
None.