COP3502 C Lab 5

In this lab we will solve the following problem that uses Stack. You should use the uploaded MultiStack.c file and then modify the code to solve the following problems. This code might be lengthy and some cases the logic might not be that straight forward. So, it would be best if you complete most of the part within the lab time with the help of the TA.

Problem1: Parenthesis checker

Write a function *int checkBalance(char exp[]); t*hat can take an expression as input and check whether the parenthesis used in that expression are valid or invalid. It returns 1 if it is valid, and returns 0 if it is not valid.

For example, the following expressions are valid: [A * {B + (C + D)}] and [{()()}]

The following expressions are invalid: [A * {B + (C + D})], [{ ( ]) ( ) }

Hints:

1. **Mainly follow the steps we learned in the lecture. See the stack slide**

2. Download the code MultiStack.c

3. Change the array of the stack structure to hold char array

4. Modify mainly the main function. Also modify the pop function to return char.

Problem 2: Infix to Postfix

We have covered the detailed steps in the class on how to convert infix to post fix. Now, in the lab we will write most part of the code. You have to complete it and upload it by next week's Friday. Build your code based on the solution in problem 1 as you will use the same parenthesis checker in this process.

Implementing the following functions will make the process easier:

```
int priority(char ch); //returns the priority of a given operator
int isOperator(char ch); //check whether it is an operator
char *infixToPostfix(char infix[]); //convert a given infix into postfix
```

int isParentheses(char ch1); //check is it a parenthesis or not.

Sample Input/Output:

Example 1

Enter Expression: (7 - 3) / (2 + 2))
Your input expression: (7 - 3) / (2 + 2))

Checking balance...
INVALID for)!!!

Example 2

Enter Expression: (5+6)*7-8*9
Your input expression: (5+6)*7-8*9

Checking balance...
VALID
The postfix is: 5 6 + 7 * 8 9 * -
evaluation: 5 //optional

Example 3:

Enter Expression: (7 - 3) / (2 + 2)
Your input expression: (7 - 3) / (2 + 2)

Checking balance...
VALID
The postfix is: 7 3 - 2 2 + /
evaluation: 1 //optinal

Example 4:

Enter Expression: 3+(4*5-(6/7^8)*9)*10
Your input expression: 3+(4*5-(6/7^8)*9)*10

Checking balance...
VALID
The postfix is: 3 4 5 * 6 7 8 ^ / 9 * - 10 * +
evaluation: 203

Example 5:

Enter Expression: 1000 + 2000
Your input expression: 1000 + 2000

Checking balance...
VALID
The postfix is: 1000 2000 +
evaluation: 3000

Problem 3: Evaluate postfix (You can work on it later and you DON'T have to submit it)

Update the code you have written above so that you can evaluate the postfix. Note that you will need an int stack during the evaluation process.