Data Structures

Lab Exercise 5 (Stacks)

Download the starter-code (lab5_startercode.cpp) from Brightspace and complete the following Tasks.

Task-1: **(0.3 Points)**

Create a templated stack class, called **CStack**, should contains at least following Methods:

- **push(e):** Insert element e at the top of the stack
- **pop():** Remove the top element from the stack. This method should generate an error/exception if the stack is empty.
- **top():** Return a reference of the top element on the stack, without removing it. This method should generate an error/exception if the stack is empty.
- **size():** Return the size (number of elements) of the stack.
- **empty():** Return true if the stack is empty, false otherwise.

Task 2: **(0.2 Points)**

Write a function called **isBalancded(expression)** that takes an expression as an argument and checks for balanced parentheses in an expression using Stack. An expression is a string comprising of opening and closing parentheses ( ), you need to check whether symbols are balanced or not.
e.g.:
(2+3)*5 is balanced.
(2+3)*(5 is not balanced.

Task-3: **(0.3 Points)**

Complete the function **infix2postfix(infix)** which converts an *Infix* expression into a *Postfix* expression using stack.

Algorithm to convert Infix to Postfix

Let, ***INFIX*** is an arithmetic expression written in infix notation. This algorithm finds the equivalent postfix expression ***POSTFIX***.

1. Push **(** onto the Stack, and add **)** to the end of ***INFIX.***
2. Scan ***INFIX*** from left to right and repeat Step 3 to 6 for each element of ***INFIX*** until the Stack is empty.
3. If an **operand** is found, add it to ***POSTFIX.***
4. If **(** is found, push it onto the Stack.
5. If an **operator** is found, then:
    a. Repeatedly pop from Stack and add to ***POSTFIX*** each operator from top of the Stack which has the same or higher precedence than operator found.
    b. Push the **operator** onto Stack.
    [End of If]
6. If **)** is found, then:

a. Repeatedly pop from Stack and add to ***POSTFIX*** each operator (from top of the Stack) until **(** is found.
b. Remove (pop) the left Parenthesis from Stack.
[End of If]

7. END.

Task-4: **(0.2 Points)**

Write a function called "**evaluate(string postfix)"** to your program, which evaluate the postfix expression using a Stack and returns the result to the calling program.

Algorithm to evaluate a postfix expression

1. Create a stack to store the operands (values)
2. Scan the ***POSTFIX*** expression from left to right for every element
   a. if an **operand** is found push it to the stack
   b. if an **operator** is found pop 2 elements from the stack, apply the operator on it and push the result back to the stack
3. The element (value) left on the stack is the final answer of the expression.

Note: For simplicity you can assume that every number and operator is only one letter long.

Sample output of your program:

```
iMac:Lab5 (Stack) khalid$ g++ lab5_sol.cpp && ./a.out
Enter an Infix Expression: (2+3)*5
The postfix form is: 23+5*
(2+3)*5=25
Enter an Infix Expression: (3+4)*5)
Expression is not Balanced
Enter an Infix Expression: (2+2)*5-(7-4)
The postfix form is: 22+5*74--
(2+2)*5-(7-4)=17
Enter an Infix Expression: 5*(3+3)*2
The postfix form is: 533+*2*
5*(3+3)*2=60
Enter an Infix Expression: exit
iMac:Lab5 (Stack) khalid$
```

Code of Conduct

All assignments are graded, meaning we expect you to adhere to the academic integrity standards of NYU Abu Dhabi. To avoid any confusion regarding this, we will briefly state what is and isn't allowed when working on an assignment/lab-task.

Any documents and program code that you submit must be fully written by yourself. You can, of course, discuss your work with fellow students, as long as these discussions are restricted to general solution techniques. Put differently, these discussions should not be about concrete code you are writing, nor about specific results you wish to submit. When discussing an assignment with others, this should never lead to you possessing the complete or partial solution of others, regardless of whether the solution is in paper or digital form, and independent of who made the solution, meaning you are also not allowed to possess solutions by someone from a different year or course, by someone from another university, or code from the Internet, etc. This also implies that there is never a valid reason to share your code with fellow students, and that there is no valid reason to publish your code online in any form.

Every student is responsible for the work they submit. If there is any doubt during the grading about whether a student created the assignment themselves (e.g. if the solution matches that of others), we reserve the option to let the student explain why this is the case. In case doubts remain, or we decide to directly escalate the issue, the suspected violations will be reported to the academic administration according to the policies of NYU Abu Dhabi.

(see https://students.nyuad.nyu.edu/campus-life/community-standards/policies/academic-integrity/)