Programming Project 1 (40 points)
Due: See Canvas for due date
Vsp.2022

Assignment Description

Write a C program that can read arithmetic-like expressions from a file and check them (ALL) with regards to parentheses '( )', square braces '[ ]', and curly braces '{ }', which are allowed to occur only in pairs and arranged in the usual way as in mathematical expressions. For example:

- {[[()]{()()}()[()]} → correct
- {(}) → incorrect.
- [{()]} → incorrect
- []{()[()]} → correct
- {[()][{()]}} → incorrect

Implementation Requirements

Your assignment must meet at least the following requirements:

1. Your program must use a stack to validate the expression. Your stack must provide functions to push, pop, and peak. (It's recommended to use an array and implement a LIFO (*last-in-first-out*) behavior.)
2. Your program must accept only input from a text file. The text file will be called "`expressions.txt`"
3. The text file includes one "math expression" per line. (No more than 10 expressions will be in any given test file)
4. The number of lines (expressions) in the file are specified in the very first line of the file
5. Your program must accept only the following characters: {, }, [,], (,) in any given math expression, (no spaces will be between characters)
6. If a different character is encountered besides the specified above, display the message
    a. "Invalid character in expression: *put_here_the_expression*"
7. Your program should contain a separate function (Not in the *main()*) for validating the proper sequence and order of the '{}', '[]', and '()', that is, the correctness of the math expression.

Documentation

8. Your program must include a comment block at the top of every file (see example below)
9. Your program must include a comment block at top of each function (see example below)
    a. The function comments should include a brief description of what the function does and explain any function arguments and return values.

You may use the comment blocks below as a template for the program's and function's comments respectively

```
/*********************************************
* Name: (YOUR NAME) *
* Date: (THE DUE DATE) *
* Assignment: Project 1 - Sequence and Order validation *
*********************************************
* (WRITE A DESCRIPTION OF THE PROGRAM) *
*********************************************/

/*********************************************
* Description: *
* Input: *
* Output: *
* Precondition: <Optional but appreciate it> *
* Post condition: <Optional but appreciate it> *
*********************************************
```

Extra 5 Points << Challenge yourself >>

10. Your program should indicate the location of the error in the expression (e.g., print chars in bold).
    a. In the example above: [{()**]**}→incorrect. (notice the **{** and the **]** in bold)

File Content Example

```
5
{[[()]{()()}()][()]}
{(})
[{()]}
[]{()[()]}
{[()][{()]}}
```

Submission

Your project must be submitted using the Assignment Submission Link in Canvas by the specified due date. You will see this link when you go to the assignment's page. If more than one file, submit a zip file of your project.
If you use CLion, please zip the folder that includes the CLion project.

For this project, Canvas will accept .zip files, and .c files ONLY.

Grading

Programs that do not compile will receive a grade of 0. A grading breakdown for programs that do compile is given below:

Accepts only correct user inputs (text file)	6
Validation of string math expression	20
Proper feedback for the user	7
Correct Program documentation/comments	7
Extra credit: Error location (+5)	
Total	**40**