CCDSALG Term 3, AY 2023 – 2024

Project 1 – Expression Evaluation (Stack and Queue Application)

Groupings : At most 3 members in a group
Deadline : June 21, 2024 (F) before 8:00 A.M
Percentage : 20% of the Final Grade
Programming Language: C

Let's apply what you learned in stacks and queues. For this project, you will develop your own (simple) calculator. Read, understand, and comply with the project specs described below.

I. INPUT

1. The input is an INFIX expression. For simplicity, assume that the tokens making up the INFIX expression are stored in a string with at most 256 characters, including the NULL byte. To simplify the task further, assume also that there are NO spaces separating the tokens and that the expression is syntactically valid.
2. Handle the following tokens:
    a. Operands are limited to non-negative integers, i.e., 0 and higher. There is no need to consider negative integers, or floating point values. For logical operators only: assume that the operands are limited to 0 and 1 only.
    b. Operators include:
        i. Arithmetic operators: +, -, *, %, ^ (caret is for exponentiation)
        ii. Relational operators: >, <, >=, <=, !=, ==
        iii. Logical operators: !, &&, ||
    c. Parentheses as grouping symbols ( )

Example input #1: `1+2*3`
Example input #2: `(8+2)/3`
Example input #3: `5>=3`
Example input #4: `1&&0`
Example input #5: `!(2>1&&3<2)`
Example input #6: `123/0`

II. OUTPUT

The required output, i.e., printed output are:

a. POSTFIX expression - separate two consecutive tokens with exactly one SPACE character

For example input #1, the output should be: `1 2 3 * +`
For example input #5, the output should be: `2 1 > 3 2 < && !`
For example input #6, the output should be: `123 0 /`

b. Evaluated value of the postfix expression

For example input #1, the evaluated value should be: `7`
For example input #2, the evaluated value should be: `3`
For example input #3, the evaluated value should be: `1`

For example input #4, the evaluated value should be: **0**
For example input #5, the evaluated value should be: **1**
For example input #6, there is no evaluated value. Print instead: **Division by zero error!**

Notes:

- The division operator is applied to integer operands. Following C language notation, the result should also be an integer. For example, 10/3 gives a result of 3 (see Example input #2).
- In case the denominator is zero, do not perform the actual division since it will cause a run-time error. Your program should check this case, and if it occurs, your program should print the constant string "Division by zero error!" (see Example input #6).
- For relational and logical operators: the result should be limited to a 0 and 1, where 0 means False and 1 means True (see Example input #3 to #5).

III. REQUIRED PROGRAM INTERACTION

An example of the required program interaction is shown below. Those in black bold font represent the user input. Those in blue bold font represent the output postfix expression. Those in green bold font represent the output evaluated value. The red bold font represents an error message due to division by zero. Those in brown are not part of the interaction – they serve as an explanation for you to understand what each line means.

The program should run in a loop when executed. The user inputs the infix expression (as a string). The program will then output the postfix expression on the next line and the corresponding evaluated value on another line. To terminate the loop, the user inputs QUIT (all in capital letters). **The program should have a MINIMALIST interaction. This means that you should NOT have any prompts and that you should NOT print anything else except the required output. Non-compliance will result in point deductions.**

```
1+2*3                          // user inputs the infix expression
1 2 3 * +                      // the program prints the postfix expression on the next line
7                              // the program prints the evaluated value on the next line

!(2>1&&3<2)                    // user inputs the infix expression
2 1 > 3 2 < && !               // the program prints the postfix expression on the next line
1                              // the program prints the evaluated value on the next line

123/0                          // you user the infix expression
123 0 /                        // the program prints the postfix expression on the next line
Division by zero error!        // the program prints the error message on the next line

QUIT          // type QUIT in all caps to quit the program
```

IV. TASKS

Task 1. Design and implement your stack and queue data structures

- You will need two stacks, one is for storing operators, another one for storing operands.
- You will need a queue for storing the tokens corresponding to the postfix expression.
- Practice modular programming. That is, compartmentalize the data structures and operations by storing the implementation codes in separate source code files. For example, the codes for stack data structure are stored in **`stack.h`** and **`stack.c`** while the codes for the queue data structure are stored in **`queue.h`**, **`queue.c`**.

- Do not use goto statement.

Task 2. Implement the algorithms for (a) infix to postfix conversion, and (b) evaluation of postfix expression.

- Again, practice good programming. Compartmentalize each algorithm implementation in separate source code files, for example **`infix-to-postfix.c`**, **`evaluate-postfix.c`**. The tokens comprising the postfix expression should be stored in a queue.

Task 3. Integrate the modules.

- Create the main module, which will include the other modules, and call the appropriate functions to achieve the task.

Task 4. Test your solution.

- Create test cases and perform exhaustive testing. You should test each operator and the grouping symbols. You should test that operator hierarchy and operator associativity are observed properly.
- Create at least 50 test expressions. Don't forget to test the division by zero error.
- Hint: Use input and output redirection to minimize keyboard input. Encode your infix expression in a text file. Run your exe file in the command line. For example:

```
CCDSALG> solution < infix.txt > result.txt
```

  where **`solution`** is the exe file, **`infix.txt`** is a text file containing the test cases (infix expressions), and **`result.txt`** is the text file that will contain the corresponding output.

Task 5. Document your solution.

- Give a brief description of how you implemented the data structures. For example did you use arrays? Did you use linked list (dynamic memory allocation)? Did you use a circular queue?
- Give a brief description of how you implemented the algorithms.
- Disclose what is not working (for example, your solution does not handle logical operators).

V. DELIVERABLE

Submit via Canvas a ZIP file named GROUPNAME.ZIP which contains:

- Source files for the stacks, queues, conversion and evaluation algorithms, main module
- Test case file named TESTCASE.TXT which contains at least 50 sample infix expressions that you used for testing your solution. Each infix expression should be encoded in a separate line of text.
- Corresponding RESULT.TXT that contains the result.
- Documentation named GROUPNAME.PDF (see attached Word documentation template).

Do NOT include any EXEcutable file in your submission.

VI. WORKING WITH GROUPMATES

You are to accomplish this project in collaboration with your fellow students. Form a group of at least 2 to at most 3 members. The group may be composed of students from different sections (handled by the same teacher). Make sure that each member of the group has approximately the same amount of contribution to the project. Problems with groupmates must be discussed internally within the group and, if needed, with the teacher.

VII. NON-SUBMISSION POLICY

Non-submission of the project by the due date will result in a score of 0 for this graded assessment.

VIII. HONESTY POLICY AND INTELLECTUAL PROPERTY RIGHTS

Honest policy applies. You are encouraged to read and gather the information you need to implement the project. **However, please take note that you are NOT allowed to borrow and/or copy-and-paste -- in full or in part any existing related program code from the internet or other sources (such as printed materials like books, or source codes by other people that are not online). You should develop your own codes from scratch by yourselves, i.e., in cooperation with your groupmates.**

Cheating or intellectual dishonesty is punishable with a final grade of 0.0.

IX. RUBRIC FOR GRADING

Criteria	RATINGS		
1. Stack	**COMPLETE [20 pts]** Correctly implemented the stack data structure.	**INCOMPLETE [5 to 10 pts]** Implementation is not entirely correct.	**NO MARKS [0 pt]** Not implemented.
2. Queue	**COMPLETE [20 pts]** Correctly Implemented the stack data structure.	**INCOMPLETE [5 to 10 pts]** Implementation is not entirely correct.	**NO MARKS [0 pt]** Not implemented.
3. Infix-to-Postfix	**COMPLETE [25 pts]** Correctly implemented the infix-to-postfix conversion algorithm. Processed all operators, and grouping symbols correctly.	**INCOMPLETE [5 to 20 pts]** Implementation is not entirely correct. Did not process some operators and grouping symbols correctly.	**NO MARKS [0 pt]** Not implemented.
4. Postfix Evaluation	**COMPLETE[20 pts]** Correctly implemented the postfix evaluation algorithm. Processed all operators correctly.	**INCOMPLETE [5 to 15 pts]** Implementation is not entirely correct. Did not process all operators correctly.	**NO MARKS [0 pt]** Not implemented.
5. Documentation	**COMPLETE [10 pts]** Required details were covered.	**INCOMPLETE [1 to 6]** Some required details were not discussed.	**NO MARKS [0 pt]** No discussion or documentation.
6. Compliance with Instructions	**COMPLIANT [5 pts]** All instructions were complied with.	**NON-COMPLIANCE [0 to 4]** Deduction of 1 point for every instruction not properly complied with. Examples: included an EXE file in the ZIP file, filenaming not followed.	**NO MARKS [0 pt]** More than 4 instructions not complied with.
			Maximum Total Points: 100

Question? Please post it in the Canvas Discussion thread. Thank you for your cooperation.

サルバドール・フロランテ