COMP 2540 Data Structures and Algorithms

University of Windsor

School of Computer Science

Assignment 2: Due after 1 week (June 15 to 19)

This assignment is to understand the applications of Stack data structures. Your task is to write a program to evaluate an arithmetic expression with the infix notation. For example, if the given arithmetic expression 14 + 3 * 2 – 5, the answer should be 15.

You should perform the following operations in your program.

1. Get the arithmetic expression from the users. (5 points)
2. Validate the expression. (**hint**: check for parentheses matching). (10 points)
3. Convert the infix to postfix expression using Stack ADT. (15 points)
4. Evaluate the above postfix expression (in step 3) using Stack ADT. (15 points)
5. Display the Stack in each ADT operations (pop and push). (5 points)

NOTE:

1. Your program should be able to handle unary negation operator (E.g 5*(-1) = - 5). (**hint**: In your expression, if an operator is appeared first, or comes immediately after another operator, or comes after a left parenthesis, then it is a unary operator. It better to handle unary

operators with different symbol because it is easy to recognize distinguish between binary and unary variants.)

2. You should test the following test cases in your program.
    a. 2 + 4 * 8 / 2 = 18
    b. (100 + 200) / 3 =100
    c. (6 + (- 3) - 1) =2
    d. 5000 + 3000 * 5 * (-1) = -10000
    e. (2 ^ 2) ^ (2 + 1) =64
    f. ((((2)))) ^ 3 = 8
    g. (100 * (2 + 1) = invalid

    To do the test cases, you can,
    a) Create a text file with the above test cases, read from the text file for the input and write the output in the text files. OR
    b) test each case one by one as a user input and create a make file. (submit your make file with your source code.)
3. You can use built-in functions to handle the power operator (e.g., 2^3 =8).
4. The precedence of arithmetic operators is:
    a. Parentheses () [] {}
    b. Exponentiation (power or ^)
    c. Any unary negation operator (-x), (right-associative)
    d. Any multiplication and division (*, /) (left-to-right associative)
    e. Any addition and subtraction (+, -) (left-to-right associative)

Requirements:

I. **UNDOCUMENTED OR IMPROPERLY DOCUMENTED** code will automatically lose 50% marks.

II. Readability: The code is exceptionally well organized and very easy to follow.

III. **PLAGIARIZED** work will not be graded and receive a mark of **ZERO** and reported according to the Senate bylaws.

Submission:

I. Your assignment must be RECEIVED by the due date and time. Late assignment submissions are NOT accepted will receive a mark of ZERO.

II. Before the submission, you must contact your GA/TA through your lab MS team channel for the feedback.

III. Submit your work through Blackboard: Source code

Rubric:

5 - The program works and meets all the above requirement.

4 - The program works, fair readability, most of the code could be reused and fair documentation.

3 - The program works, fair readability, most of the code could be reused and NO documentation.

2- The program produces some results but does not display them correctly, poor readability and NO documentation.

1- The program produces compilation error, poor readability, and NO documentation

0 - Did not complete any task.

PLUS

*** 1 - Contact the GA/TA to answer the questions. (attendance + students' involvement)