Indian Institute of Information Technology, Allahabad

Advanced Data Structures and Algorithms (Jul-Dec 2018)

Lab Assignment

TAs: Fahiem Altaf, Chanchal Maurya, Ruchi Dewangan, Vasudha, Shiva Charan, Shrishti, Neeraj Kumar and Abhinavraj Singh

Date of Assignment: 20/08/18 **Due Date: 23/08/18** **Instructor: Dr. S. Maity**

Note: Code the following exercise(s) in C++. Use templates to declare stack ADT wherever required.

Write a program to evaluate a given infix expression.

- An infix expression may contain five types of operations: addition (+), subtraction (-), division (/), multiplication (*) and exponentiation (^).
- The expression may or may not contain parenthesis.
- Order of precedence: exponentiation > multiplication/division > addition/subtraction. Assume left to right associativity for all operators except in exponentiation in which we assume right to left associativity. Parenthesis can be used to override these precedence rules.
- Your program should accept only *valid* infix expressions. A valid infix expression has equal number of left and right parenthesis such that every right parenthesis is preceded by a matching left parenthesis.
- The input infix expression may contain *single* or *multiple* digit operands having *positive* or *negative* sign.
- *Reduce*, if possible, the given infix expression to an equivalent infix expression by removing all unnecessary parenthesis. Display the *reduced infix expression*.
- *Convert* the *original* infix expression to postfix expression. At each step of conversion, display the contents of *stack* and *intermediate postfix expression*. Display the *final postfix expression*.
- Finally, *evaluate* the postfix expression derived in previous step. At each step of evaluation, display the contents of *stack*, *operands* being acted upon and the *operation* being performed.

***