Data Structures C, D
FAST-NU, Lahore, Fall 2016

Homework 3

Expression Evaluation using a Stack

Due on Thursday September 15 before class Marked out of 50 points.

In this homework, your task is to build a simple calculator for arithmetic expressions. You must implement your own Stack class for this purpose.

Your program is be expected to:

- Read the infix expressions
- Convert them to postfix
- Evaluate the resulting postfix expressions

Your calculator must support the following minimal set of operations:

- Binary operators: +, -, *, /
- Unary operators: ++ (pre-increment) , -- (pre-decrement)

Default associativity rule to be maintained is left to right

The program must also detect invalid infix expressions and report as such.

The two main algorithms have been reproduced below for your convenience:

Infix to Postfix

initialize stack

while tokens remain in infix expression

x = get next token

if x is operand

    add to postfix expression

else if x is a "("

    push x to stack

else if x is ")"

    while top of stack != "("

unstack operators and add to postfix expression

pop “(“

else if x is an operator

while precedence (top of stock)<= precedence(x)

unstack operators and add to postfix expression

push x to stack

while operator stack not empty

unstack operators and add to postfix expression

Evaluate Postfix

initialize stack

while tokens remain in postfix expression

x = get next token

if x is operand

push to stack

else if x is an operator

if x is binary operator

pop last value in stack – right operand

pop second last value in stack – left operand

apply operation

push result to stack

if x is unary operator

pop last value in stack

apply operation

push result to stack

if expression was correct then last value remaining in the stack is the answer

Following is an example of how your program will be used. Make sure this is exactly how it works you marks will be deducted during evaluations:

The following commands are available:

eval	-- Evaluate the expression and print out its value
quit	-- Quit the system

All binary operators and their operands will be separated by at least one blank space.

All parentheses will also be separated from the rest of the symbols in a similar manner.

Comments following the ``//" in this picture are for explanation, not part of the session.

```
cmd> calculator              // The executable file

calculator> eval 3 + 5
Postfix: 3 5 + 7 -
Answer: 8

calculator> eval 2 * ( 3 - 1 )
Postfix: 2 3 1 - *
Answer: 4

calculator> eval ++17
Postfix: 17++
Answer: 18

calculator> eval 9 + --2
Postfix: 9 2-- +
Answer: 10

calculator> eval 4 - ( 1 + ++3 )
Postfix: 4 1 3++ + -
Answer: -1

calculator> eval -9 + 6
Postfix: 9- 6 +
Answer: -3
```

***

THE END