**CSE 281: Data structures and Algorithms Lab**

**Lab sheet IV: Application of Stack**

*Instructions*

* *Write the algorithm and java program codes for the questions from* ***1 to 3*** *in the lab record.*
* *Reference to java API is available at :* [*http://192.168.0.48/javadocs/api/index.html*](http://192.168.0.48/javadocs/api/index.html)
* *(Skeleton code for the Exercise 3 is uploaded in the resource)*

**Exercise I:** Implement the balanced parenthesis algorithm discussed in class as a function and add it into previous stack code. Input following test cases and check whether it is balanced or not.

{ ( ) { [ ( ) ] } }

[ ( ] )

) (

{ ( ) ( ) ( ) [ { } ] }

**Exercise II:** Postfix Expression Evaluation

Implement a program in a class named PostFixEvaluator which calculates the value of the post fix expression.

Test case:

Input: 1 2 3 \* + 4 - Output: 3

Input: 2 3 \* 15 5 / + 10 - Output: -1

Input: 10 2 \* 8 4 / + Output: 22

**Exercise III :** Infix to postfix conversion

This problem requires you to write a program to convert an infix expression to a postfix expression. Use the precedence relation discussed in class. ie prec( +,-) =1, prec( \*, / ) =2 , prec( ^) =3

Input: A collection of error-free simple arithmetic expressions. The input has an arbitrary number of blanks between any two symbols. A symbol may be a letter (A – Z), an operator (+, – , \*, or /), a left parenthesis, or a right parenthesis. Each operand is composed of a single letter. The input expressions are in infix notation.

*Example*

A + B – C

A + B \* C

(A + B) / (C – D)

( ( A + B ) \* ( C – D ) + E ) / (F + G)

You can take input in java code as String input = " A + B – C ";

*Output:* Your output will consist of the input expression, followed by its corresponding postfix expression. All output (including the original infix expression) must be clearly formatted.

*Example*

A B + C –

A B C \* +

A B + C D - /

A B + C D - \* E + F G + /