Assignment Prefix: Lab09
Points: 100
Due Date: Friday, October 28, 2016 @ 11:59pm
This is an individual assignment.

Task:

Write a Java project that:

- Implements the Shunting Yard algorithm to convert an in-fix expression into its corresponding post-fix notation.
- Uses the queue and stack approach to evaluate an expression that is in post-fix notation.
- Uses the queue and stack approach to build the binary expression tree for an expression that is in post-fix notation.

You must use your implementations of trees, stacks and queues from previous assignments.

Your program should:

- Read arithmetic expressions from an input file until the EOF is reached.
    - See file format and example at end of assignment.
- For each expression your program should:
    - Print out the expression that was read from the file.
    - Determine if the expression is valid.
        - Print an invalid expression message for invalid expressions.
        - For each valid expression
            - Print the expression in post-fix notation
            - Print the expression tree using the pre-order traversal
            - Print the expression tree using the in-order traversal
            - Print the expression tree using the post-order traversal

Input file format:

Each token in the input file will be blank separated so the expressions should be easy to parse. Tokens will be one of the following:

- Numeric value possibly includes negative numbers
  - The uniary negative operator will not have a blank space between the operator and its corresponding operand, e.g. -45
  - The binary subtraction operator will have blank space between the operator and its corresponding operands, e.g. 11 - 5

- Operators will be limited to:
  - Addition +
  - Subtraction -
  - Multiplication *
  - Division /

- Parenthesis
  - In order to make expression more readable parenthesis, curly brackets and square brackets may be used.
  - For grouping and nesting purposes the symbols must match correctly.
  - For example:
    - (3 - [{ 4 / 3 } + 7] - 2) is correct grouping
    - ({ [}]) is incorrect nesting

- There will be no "implied" multiplication
  - The expresson 3 * (4 - -5) is valid
  - The expresson 3 (4 - -5) is not valid

- You do **not** need to check for invalid tokens.

Example input file (data.txt):

```
3 * -5
4 - 3 / 5
( 4 - 3 ) / 5
4 + ( 7 / 2 )
4 + 7 8 - 11
( ( 3 + 1 ) * 3 ) / ( ( 9 - 5 ) ) - ( ( 3 * ( 7 - 4 ) ) + 6 ) )
3 + 1 * 3 / 9 - 5 - 3 * 7 - 4 + 6
42
8 * 24 / ( 4 + 3 )
3 + 4 –
```