EECS 233 Homework 4

General requirements:

- Due at 11:00 PM on the posted due date.
- Include your name and network ID as a comment at the top of all of your programs.
- Upload all .java files as a .zip file to Blackboard. Do not use other formats such as .rar.
- All work should be your own, as explained in the Academic Integrity policy from the syllabus.

Instructions: This assignment requires you to use the MyLinkedList sample code as the basis for a StackList class. You may copy/paste the code from MyLinkedList.java and revise it, or you may use that class within a StackList class. Calculator1.java was provided in class as an example for Strings. Write the following programs:

1. Create your own StackList class that contains the following methods:
   a. push: Follows the standard "push" operation as described in class.
   b. pop: Returns the value at the top of the stack and removes it from the stack. Error checking for an empty stack is not required, but it may be included if you wish.

2. Ask the user for a parenthesized expression, and determine whether the parentheses in the expression are balanced. Use the push and pop methods from the StackList class in problem #1 above. Note: it can completely ignore all characters that are not parentheses (see below).
   *Output examples (user input in **bold**):*

`Enter expression: (2*(3+4)-5)/6` `Parentheses are balanced.`	`Enter expression: (2*(3+4-5)/6` `Parentheses are NOT balanced.`
`Enter expression: 2*(3+4)-5)/6` `Parentheses are NOT balanced.`	`Enter expression: )()()()(` `Parentheses are NOT balanced.`

3. Ask the user for a fully parenthesized infix expression with single-digit numbers (without spaces), and compute the actual result. Use the push and pop methods from the StackList class in problem #1 above. Assume the expression is correctly parenthesized.
   *Output examples (user input in **bold**):*

`Enter expression: (((1+2)+3)+4)` `Result = 10.0`	`Enter expression: (((2*(3+4))-5)/3)` `Result = 3.0`

4. Ask the user for a postfix expression with single-digit numbers (without spaces), and compute the actual result. Use the push and pop methods from the StackList class in problem #1 above.
   *Output examples (user input in **bold**):*

`Enter expression: 23+4*` `Result = 20.0`	`Enter expression: 6523+8*+3+*` `Result = 288.0`

Tip (error checking): Unless stated otherwise, you do not have to check for valid expressions. You may assume the programmer uses the StackList correctly (no invalid calls to "pop"). Problem #3 does not have to check for balanced parentheses, but you may add such a feature if you wish.

Tip (double digits and integer division): Keep it simple, unless you want the extra challenge. Only single-digit numbers are required. If you wish to accept more than single digits, you will want spaces in your expressions.

Tip (data types): It is recommended that you use the Character data type for operators and parentheses and Double for numbers.

Tip (full parenthesization): Remember that "fully parenthesized" means that you have a pair of parentheses for every operator. An example is (1+(2+3)+4). Note that you will have parentheses around the whole expression.

Rubric:

Item	**Points**
1: Reusing MyLinkedList.java correctly	5
1: pop and pop methods	15
2: Correct use of StackList (declaring, using pop/push)	5
2: User input and printing output	5
2: String processing loop (general loop logic, indexing)	5
2: Logic for handling characters (when to push/pop, etc.)	5
3: Correct use of StackList (declaring, using pop/push)	5
3: User input and printing output	5
3: String processing loop (general loop logic, indexing)	5
3: Logic for handling parentheses	5
3: Logic for handling operators (when to push/pop, etc.)	5
3: Logic for handling operands (when to push/pop, etc.)	5
4: Correct use of StackList (declaring, using pop/push)	5
4: User input and printing output	5
4: String processing loop (general loop logic, indexing)	5
4: Logic for handling operators (when to push/pop, etc.)	5
4: Logic for handling operands (when to push/pop, etc.)	5
General programming (good practices, compiles, etc.)	5
Total	100