COMS 228, Spring 2021

# Project 3: Infix-to-Postfix Conversion

Due: at 11:59 PM
**Thursday**, April 15

## 1. Problem Description

Create a program that reads a file `input.txt` of infix expressions and converts them to the corresponding postfix version, printing the expressions out to a file `output.txt`. Please consult the files `24.pdf` and `24 infix2postfix.pptx` from the Modules page for details on the algorithm.

## 2. Implementation Details

The class containing the `main()` method should be named `Infix2Postfix`. The input file is named `input.txt` and will contain infix expressions on separate lines. The file will be located in the default directory of the project. There is no need to implement file choosing functionality.

The output file should be named `output.txt` and should be created in the same directory as the input file. Each line of the output file should contain the postfix version of the corresponding line in the input file.

The infix-to-postfix conversion functionality should correctly account for **P**arentheses **E**xponent **M**ultiplication **M**odulus **D**ivision **A**ddition **S**ubtraction precedence rules. The input file will not contain braces or brackets BUT may contain many nested levels of parentheses, such as:

12 + ( ( 23 * ( 4 ^ ( 3 – ( 7 / ( 11 ^ 2 ) ) ) ) ) % 25 )

## 3. Input File Format

The input file is named `input.txt` and will contain infix expressions on separate lines. The file will be located in the default directory of the project.

Each line in the file will contain integers, operators, and parentheses. There is no need to guard against other types of tokens. The expression tokens will be separated by a single space. Separate infix expressions will be located on separate lines. There may be many infix expressions in the input file.

Example input file:

```
12 + ( ( 23 * ( 4 ^ ( 3 – ( 7 / ( 11 ^ 2 ) ) ) ) ) % 25 )
7 ^ ^ 3 ^ ( 4 * ( -15 - -2 ) ) / ( ( 29 + 7 ) ^ 2 )
-4 - ) ) + 15
2 * 3 / 12 - -16
```

## 4. Output File Format

The output file should be named `output.txt` and should be created in the same directory as the input file. Each line of the output file should contain the postfix version of the corresponding line in the input file.

Postfix expression operators and operands should be separated by a single space, and postfix expressions should be printed on separate lines in the file. There should not be any spaces after the last token on an output line.

If the infix expression contains errors, the first error in the expression should be printed on its output line INSTEAD of any tokens from the postfix expression. The offending token should be printed after the error, surrounded by parentheses. If the error token is a parenthesis, it should not be printed together with the error message:

```
Error: too many operands (-14)
Error: too many operators (*)
Error: no closing parenthesis detected
Error: no opening parenthesis detected
```

Example output file:

```
-5 -2 8 * +
Error: too many operators (^)
Error: no opening parenthesis detected
Error: too many operands (291)
3 4 2 5 ^ - * 6 +
```

## 5. Plagiarism

Submitting code that is not your original effort or showing your code to other students is not permitted.

## 6. Submission

Write your classes in the `edu.iastate.cs228.hw3` package. Turn in the zip file and not your class files. Please follow the guideline in submission_guide.pdf.

You are not required to submit any JUnit test cases. Nevertheless, you are encouraged to write JUnit tests for your code. Since these tests will not be submitted, feel free to share them with other students.

Include the Javadoc tag @author in every class source file you have made changes to. Your zip file should be named `Firstname_Lastname_HW3.zip`.