Program #7: Infix to Postfix (and Evaluation)

For this assignment, I want you to create and submit a `Infix2Postfix.java` file which can be used to convert single digit infix expressions to their postfix equivalents, and then evaluate the postfix expressions. It should receive the infix expressions from `stdin` and send output to `stdout`. An example run of your program might be:
`java Infix2Postfix < SampleInputFile > ResultingOutputFile`
Your program should also make use of the `Stack.java`, `Queue.java`, and `List.java` programs that you created for Programs #3, #4 and #5.

Here are a few things to note:

1. You will be provided with two files i.e. a sample input file (`input`), and its corresponding output file (`output`)that you can compare with your own output.
2. The sample input file provided contains a single infix expression per line.

```
5+7
3*8+6
2^2+9
7^7-6^6
9-5*8/4^2
2^3^2-4^2+4
3*3+9*8-2
6-4-3+3+4*6/2+8*8
4-3*(4*(3*6-(8+9)^2)*(9-3)/3)-7
(3+(9-3)*(5-3)*4)
((((((9-9))))))
((9/2*3/2+1)*1)*1*1*1*1
9*3-(5-7/5)
```

3. Structure your output so that it displays the infix expression on a line, the postfix expression on the next line, the result on the next line, followed by a blank line. Do not put spaces in between operators or operands; for example:

```
5+7
57+
12.0

9*3-(5-7/5)
93*575/--
23.4
```

4. Producing the output above will require a slight modification of your `List.java` file such that it does not print spaces in between list items.
5. Your `Infix2Postfix.java` file will require the `Stack.java`, `Queue.java`, and `List.java` files to exist within the same folder.

6. Evaluations must be precise (i.e., no integer division). Think about what this means in terms of your evaluation stack.
7. You might want to change the `MAX_SIZE` of your list such that it can support an entire line of input from `stdin` and not just 50 items. I suggest changing the `MAX_SIZE` to 1024.

This is a large and relatively complicated program so I suggest a few things

- Start this project as soon as possible. **Procrastination, and NOT program complexity, will be your worst enemy in this project**.
- Break the project into smaller parts or units in both design, and the order in which you implement it. For example, you could:
  - Start with making sure that your code can read a line from `stdin`, and reproduce that line in `stdout`.
  - Follow up by using I/O redirection such that your program can read the input file line by line, and reproduce that line in the output.
  - Write functions that will allow you to successfully evaluate a postfix expression using a stack (the same way we did in class and in your midterm). You can test this part using another input file that you can create for yourself using either your own postfix expressions, or the postfix expressions from the provided `output` file.
  - Implement an algorithm that would convert an infix expression to a postfix expression (We went over the pseudocode for such an algorithm in class, and it should be in your notes).

  Remember this is a suggestion. If you feel like you have a better way of tackling this problem within its restrictions, please feel free to use that way.
- Note that `2^3^2 = 512`.
- Feel free to compare your output to mine using the command prompt. For example, once completed, the three commands below should **NOT** produce any output.

Linux:

```
javac Infix2Postfix.java
java Infix2Postfix < input > myoutput
diff output myOutput
```

Windows:

```
javac Infix2Postfix.java
java Infix2Postfix < input > myoutput
fc output myOutput
```