

Project 6 – Help File: Evaluation of a Prefix expression using a Queue

One way to evaluate a prefix expression is to use queues. To solve this problem, you will use 2 queues: a working copy and a temporary copy.

To start, you will read a prefix expression with all its tokens into your working queue.

For each pass through the expression, process the expression in your working queue looking for an operator-operand-operand sequence.

For tokens not in that order sequence, simply enqueue the tokens into your temporary queue in their same order as in your working queue.

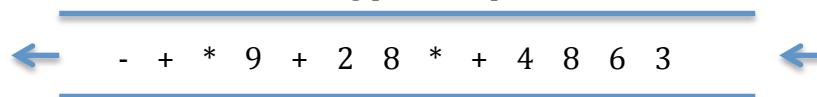
When you have that sequence, calculate and enqueue the intermediate result into your temporary queue.

Copy all remaining tokens in your working queue into your temporary queue.

Then make your temporary queue your working queue for the next pass.

Repeat the process until you complete evaluation of the expression to derive a single value as answer.

For example, to evaluate the following prefix expression:

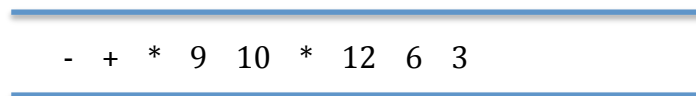


- + * 9 + 2 8 * + 4 8 6 3

We read the expression with all its tokens into a working queue.

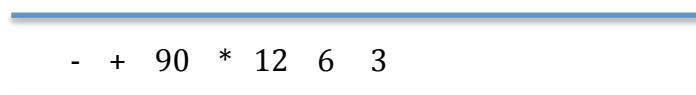
Next we examine each token in the working queue, when we find 3 tokens in the operator-operand-operand sequence, we queue them, calculate, and enqueue the answer into the temporary queue. If tokens are not in that sequence, we enqueue each token in the temporary queue in the order they are dequeued.

In our example, we dequeue “+ 2 8” and enqueue the sum 10 into the temporary queue.



- + * 9 10 * 12 6 3

After the 2nd pass, we have



- + 90 * 12 6 3

After the 3rd pass, we have

- + 90 72 3

After the 4th pass, we have

- 162 3

After the 5th pass, we have

159

Write a program using the Queue class to evaluate a prefix expression. Feel free to add methods to the Queue class as you see fit. You can use the sample prefix expressions in the Lecture Notes for Stacks and add more of your own. You may assume all input expressions are in valid prefix notation. Your program must handle expressions with multiple operations.