Infix to Postfix (Evaluating Arithmetic Expressions)

how do we represent arithmetic expressions?

```
5 + 2 * 9
```

operators take two operands; the operator is in the middle
this is an infix representation of the expression
in → operators come in between operands
how do we handle order of operations?

how about this?

```
5 2 9 * +
```

operators still take two operands, but the operands come first and the operator comes last
this is a postfix representation of the expression
post → operators come after operands
we evaluate postfix expressions by pushing operands on the stack
when we encounter an operator
pop the top two operands
perform the operation on them
push the result

try this one

```
5 + 2 * 3 + 5
```

we could just evaluate `2 * 3`
then add `5`
then add `5`

```
2 3 * 5 + 5 +
```

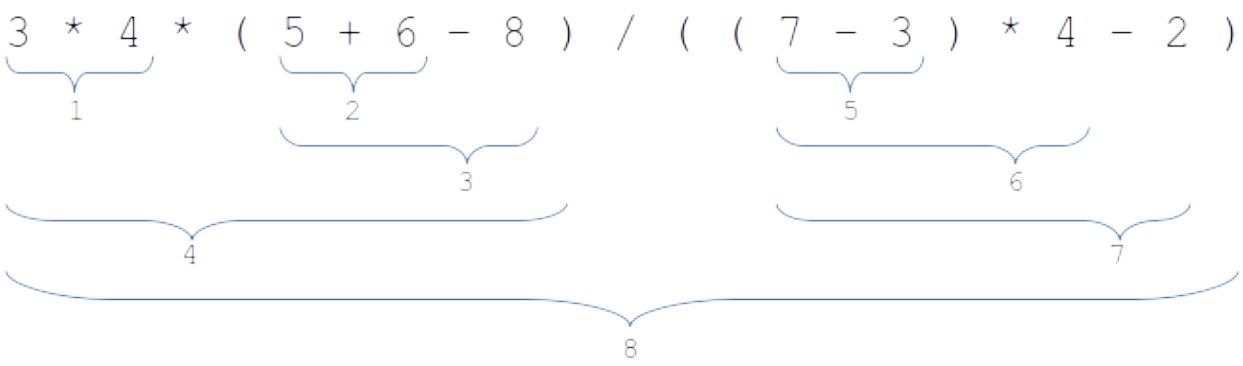
but that would mean that the operands are out of order which is hard to process left-to-right
try `5 2 3 * + 5 +`

e.g.

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

note that the order of the operands **is the same** in both infix and postfix expressions

suggest to group and order with braces; e.g.:



3 4 * 5 6 + 8 - * 7 3 - 4 * 2 - /

how to evaluate postfix expressions?

we just use a stack as described above

push operands on the stack

when we encounter an operator

```
a = Pop();
b = Pop();
Push(eval(a, b));
```

e.g. 5 2 9 * +

input	stack
5	5
2	2 5
9	9 2 5
*	18 5
+	23

e.g. 7 5 + 9 * 6 +

input	stack
7	7
5	5 7
+	12
9	9 12
*	108
6	6 108
+	114

how to convert infix to postfix?

we use an infix queue that represents the arithmetic expression

we use a postfix queue and an operator stack

we define an infix priority which prioritizes operators and parentheses

reflects the relative position of an operator in the arithmetic hierarchy

used to determine how long the operator waits in the stack before being enqueued

Token	(	^	*	/	+	-	default
Value	4	3	2	2	1	1	0

we define a stack priority

determines whether an operator waits in the stack or is enqueued on the postfix queue

Token	^	*	/	+	-	default
Value	2	2	2	1	1	0

```
infixQ ← infix expression
postfixQ ← empty queue
operS ← empty stack
repeat
      token ← infixQ.dequeue()
      if token is an operand
      then
            postfixQ.enqueue(token)
      else if token is a right parenthesis
      then
            op ← operS.pop()
            while op is not a left parenthesis
                  postfixQ.enqueue(op)
                  op ← operS.pop()
            end
      else
            op ← operS.peek()
            while stack_priority(op) >= infix_priority(token)
                  op ← operS.pop()
                  postfixQ.enqueue(op)
                  op ← operS.peek()
            end
            operS.push(token)
      end
until infixQ is empty
while operS is not empty
      op ← oper.pop()
      postfixQ.enqueue(op)
end
```

e.g. `3 * 4 - ( 7 - 8 / 2 )`

infix queue: `3 * 4 - ( 7 - 8 / 2 )`
postfix queue:
operator stack:

token: `3`
infix queue: `* 4 - ( 7 - 8 / 2 )`
postfix queue: `3`
operator stack:

token: *
infix queue: 4 - (7 - 8 / 2)
postfix queue: 3
operator stack: *

token: 4
infix queue: - (7 - 8 / 2)
postfix queue: 3 4
operator stack: *

token: -
infix queue: (7 - 8 / 2)
postfix queue: 3 4 *
operator stack: -

token: (
infix queue: 7 - 8 / 2)
postfix queue: 3 4 *
operator stack: (-

token: 7
infix queue: - 8 / 2)
postfix queue: 3 4 * 7
operator stack: (-

token: -
infix queue: 8 / 2)
postfix queue: 3 4 * 7
operator stack: - (-

token: 8
infix queue: / 2)
postfix queue: 3 4 * 7 8
operator stack: - (-

token: /
infix queue: 2)
postfix queue: 3 4 * 7 8
operator stack: / - (-

token: 2
infix queue:)
postfix queue: 3 4 * 7 8 2
operator stack: / - (-

token:)
infix queue:
postfix queue: 3 4 * 7 8 2 / -
operator stack: -

token:
infix queue:
postfix queue: 3 4 * 7 8 2 / - -
operator stack:

evaluating 3 4 * 7 8 2 / - -
postfix queue: 3 4 * 7 8 2 / - -
stack:

postfix queue: 4 * 7 8 2 / - -
stack: 3

postfix queue: * 7 8 2 / - -
stack: 4 3

postfix queue: 7 8 2 / - -
stack: 12

postfix queue: 8 2 / - -
stack: 7 12

postfix queue: 2 / - -
stack: 8 7 12

postfix queue: / - -
stack: 2 8 7 12

postfix queue: - -
stack: 4 7 12

postfix queue: -
stack: 3 12

postfix queue:
stack: 9

answer: 9