Dijkstra's Shunting Yard

Review

- **Stacks**
  - Push, pop, peek
  - Linked list and vector implementations
  - Reverses order
- **Queues** ("FIFO buffers")
  - enqueue/push, dequeue, peek
  - Linked list and circular buffer implementations
  - Maintains order
- Deque
  - Stack-queue
  - Pronounced "deck"

Evaluate the following **infix** expressions by hand:

1 + 2

10 + 4 / 2

(10 + 4) / 2

6 > 10 + 1

Postfix Evaluation Algorithm

- Input: queue of tokens
- Output: number
- Internal state: stack of numbers

```
while len(Q) > 0:
    t = dequeue(Q)
    if t is a number:
        push(S, t)
    else # t is an operator
        Pop from S the number of args that t consumes
        Push onto S the result of applying t to the args in reverse order
return pop(S)
```

Evaluate the following **postfix** expressions by hand:

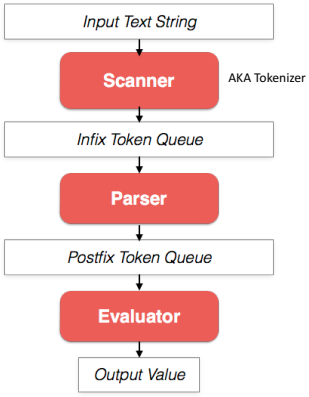
1 2 +

5 3 –

7 3 >

10 2 + 4 6 + *

1 2 3 * +



Convert the following **infix** expressions to **postfix**:
Hint: Numerals should not change order

5 < 100

1 + 4 / 2

7 * 2 * 3

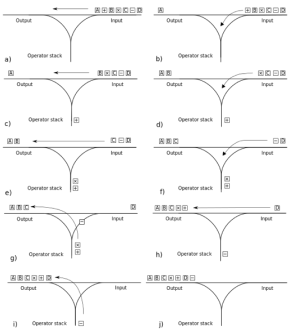
(5 – 1) / 2

10 – 9 + 4 * 2 – 7

Hagen Shunting Yard, Germany

http://en.wikipedia.org/wiki/Hagen

Dijkstra's Shunting Yard



Backwards!

C++ Operator Precedence

The following table lists the precedence and associativity of C++ operators. Operators are listed top to bottom, in descending precedence.

Precedence	Operator	Description	Associativity
1	`::`	Scope resolution	Left-to-right

2	`++ --`
	`type``() `*`type`*`{}`
	`()`
	`[]`
	`. ->`
		Suffix/postfix increment and decrement
		Functional cast
		Function call
		Subscript
		Member access	

3	`++ --`
	`+ -`
	`! ~`
	`(`*`type`*`)`
	`*`
	`&`
	`sizeof`
	`new new[]`
	`delete delete[]`
		Prefix increment and decrement
		Unary plus and minus
		Logical NOT and bitwise NOT
		C-style cast
		Indirection (dereference)
		Address-of
		Size-of[note 1]
		Dynamic memory allocation
		Dynamic memory deallocation
			Right-to-left

4	`.* ->*`	Pointer-to-member	Left-to-right
5	`* / %`	Multiplication, division, and remainder	
6	`+ -`	Addition and subtraction	
7	`<< >>`	Bitwise left shift and right shift	
8	`< <=` `> >=`	For relational operators < and ≤ respectively For relational operators > and ≥ respectively	
9	`== !=`	For relational operators = and ≠ respectively	
10	`&`	Bitwise AND	
11	`^`	Bitwise XOR (exclusive or)	
12	`|`	Bitwise OR (inclusive or)	
13	`&&`	Logical AND	
14	`||`	Logical OR	

15	`?:`
	`throw`
	`=`
	`+= -=`
	`*= /= %=`
	`<<= >>=`
	`&= ^= |=`
		Ternary conditional[note 2]
		throw operator
		Direct assignment (provided by default for C++ classes)
		Compound assignment by sum and difference
		Compound assignment by product, quotient, and remainder
		Compound assignment by bitwise left shift and right shift
		Compound assignment by bitwise AND, XOR, and OR
			Right-to-left

16	`,`	Comma	Left-to-right

Enjoy "Spring" Break