Evaluate Postfix Expressions

$$34/5 - 67 * +89 * -$$

- how to evaluate? left-to-right, "operate" when see operator
- 3, 4, / $\Rightarrow$ 0.75
- 0.75, 5, - $\Rightarrow$ -4.25
- -4.25, 6, 7, * $\Rightarrow$ -4.25, 42 (note: -4.25 stored for latter use)
- -4.25, 42, + $\Rightarrow$ 37.75
- 37.75, 8, 9, * $\Rightarrow$ 37.75, 72 (note: 37.75 stored for latter use)
- 37.75, 72, - $\Rightarrow$ ...

stored where?
stack so closest operands will be considered first!

Stack Solution to Postfix Evaluation

Postfix Evaluation

for each *token* in the input **do**
  if *token* is a number
    push *token* to the stack
  else if *token* is an operator
    sequentially pop operands $a_{t-1}, \cdots, a_0$ from the stack
    push $token(a_0, a_1, a_{t-1})$ to the stack
  end if
end for
return the top of stack

matches closely with the definition of postfix notation

One-Pass Algorithm for Infix to Postfix

infix ⇒ postfix efficiently?

- at /, not sure of what to do (need later operands) so **store**

$$a/b - c + d * e - a * c$$

- at -, know that a / b can be a b / because - is of lower precedence

$$a/b - c + d * e - a * c$$

(a b /)

- at +, know that ? - c can be ? c - because + is of same precedence but {-, +} is left-associative

$$a/b - c + d * e - a * c$$

- at *, not sure of what to do (need later operands) so **store**

$$a/b - c + d * e - a * c$$

ab/c- de* +

stored where? stack so closest operators will be considered first!

Stack Solution to Infix-Postfix Translation

```
for each token in the input do
    if token is a number
        output token
    else if token is an operator
        while top of stack is of higher (or same) precedence do
            pop and output top of stack
        end while
        push token to the stack
    end if
end for
```

- here: infix to postfix with operator stack
  —closest operators will be considered first
- recall: postfix evaluation with operand stack
  —closest operands will be considered first
- mixing the two algorithms (say, use two stacks): simple calculator

Stack Solution to Infix-Postfix Translation

```
for each token in the input do
    if token is a number
        output token
    else if token is an operator
        while top of stack is of higher (or same) precedence do
            pop and output top of stack
        end while
        push token to the stack
    end if
end for
```

pop all

- here: infix to postfix with operator stack
  —closest operators will be considered first
- recall: postfix evaluation with operand stack
  —closest operands will be considered first
- mixing the two algorithms (say, use two stacks): simple calculator

Some More Hints on Infix-Postfix Translation

```
for each token in the input do
    if token is a number
        output token
    else if token is an operator
        while top of stack is of higher (or same) precedence do
            pop and output top of stack
        end while
        push token to the stack
    end if
end for
```

- for left associativity and binary operators
  - right associativity? same precedence needs to wait
  - unary/trinary operator? same
- parentheses? higest priority
  - at '(', cannot pop anything from stack
    —like seeing '*' while having '+' on the stack
  - at ')', can pop until '(' —like parentheses matching

Some More Hints on Infix-Postfix Translation

```
for each token in the input do
    if token is a number
        output token
    else if token is an operator
        while top of stack is of higher (or same) precedence do
            pop and output top of stack
        end while
        push token to the stack
    end if
end for
```

- for left associativity and binary operators
  - right associativity? same precedence needs to wait
  - unary/trinary operator? same
- parentheses? higest priority
  - at '(', cannot pop anything from stack
    —like seeing '*' while having '+' on the stack
  - at ')', can pop until '(' —like parentheses matching

Queues

Queue

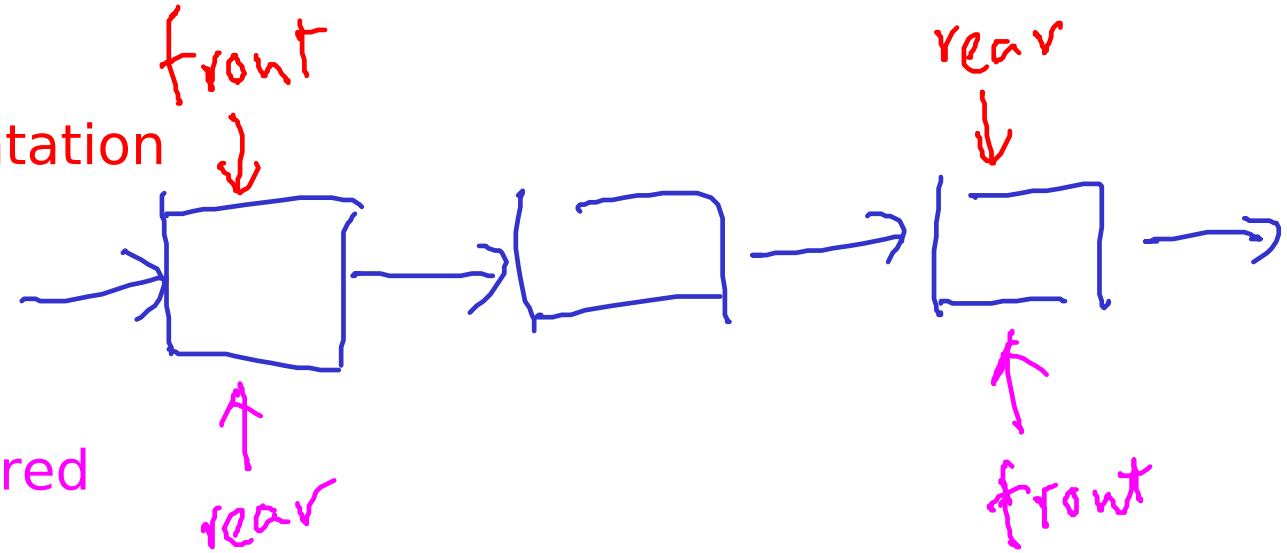
- object: a container that holds some elements
- action: [constant-time] enqueue (to the rear), dequeue (from the front)

- first-in-first-out (FIFO): 買票， 印表機
- also very restricted data structure, but also important for computers

Reading Assignment

be sure to go ask the TAs or me if you are still confused

linked list implementation



not preferred