# Data Structures and Algorithms

Session 5. February 2, 2009

Instructor: Bert Huang

http://www.cs.columbia.edu/~bert/courses/3137

# Announcements

- Homework 1 is due now
- Homework 2 will be posted after class

# Review

- Stacks
- Applications
  - Recursion
  - Syntax Checking
  - Postfix Evaluation
- Implementation

# Today

- Briefly look over Homework 2
- (Header Nodes for Linked Lists)
- Stack Wrap up
- Queues

# Header Nodes

- Convenient way to keep track of empty lists
- header nodes a.k.a. **sentinel** nodes
- Without sentinels, removing first and last nodes need special handling
- With sentinels, all adds and removes are the same operation

# Stack Implementations

- Linked List:
    - Push(x) <-> add(x,0)
    - Pop(x)  <->  remove(0)
- Array:
    - Push(x) <-> Array[k++] = x
    - Pop(x)  <->  return Array[--k]

# Queues

- Stacks are **L**ast **I**n **F**irst **O**ut
- Queues are **F**irst **I**n **F**irst **O**ut, first-come first-served
- Operations: **enqueue** and **dequeue**
- Analogy: standing in line, garden hose, etc

# Queue Implementation

- Linked List
  - add(x,0) to enqueue, remove(N-1) to dequeue
- Array List won’t work well!
  - add(x,0) is expensive
  - Solution: use a circular array

# Circular Array Queue

- Don’t bother shifting after removing from array list
- Keep track of start and end of queue
- When run out of space, wrap around
    - modular arithmetic
- When array is full, increase size using list tactic

# Stacks and Queues in Java

- Queue interface uses different operation names
  - offer == enqueue
  - poll == dequeue
- LinkedList implements Queue
- Stack is a built in class. Uses push() and pop()
- Deque interface is “double-ended queue”

# Assignments

- Homework 2
- Weiss Sections 4.1 and 4.2