Announcements

- Midterms will be back Wed. or Thurs.
- Program 2 - *checkpoint tomorrow (now due Sunday by midnight)*
- Next program out sometime this week
Last time: Linked Queues (FIFO)

```c
struct Node {
    Object element;    // value of this node
    Node * next;       // ptr to next node

    // constructor
    Node (const Object & e = Object(), Node* n = NULL) {
        element(e), next(n) = 3
    }
}
```

![Diagram of a linked list with front and back pointers]
private data:

Node*
Node*
int
size

front:
back:
dqueue: delete this node
enqueue (6)
We coded:

- enqueue
- dequeue
- front
- remove All
- copy From

What's left? House keeping
Housekeeping functions

3 things left:

Copy Constructor
Destructor
Operator =
Destructor

\[ \sim \text{Linked Queue()} \{
    \text{remove All()},
\}\]
Copy Constructor:

Linked Queue(const LinkedQueue & other)

Copy From (other);
Operator = i

LinkedQueue & operator=(const LinkedQueue & other) 

if (this != & other) 

remove All ()
copyFrom (other)

return *this;

(end 4.4)
Doubly Ended Queues (Deques)

Like a queue, but allows insertion or removal from either end.

Supports:
- insert First
- insert Last
- remove First
- remove Last
Problem: Will our nodes work here?

Need **Doubly Linked List**

- Insert at end
- Removing the tail
Solution - new Node class
(this struct will come in handy)

struct Node {
    Object element;
    Node * next;
    Node * prev;

    Node (const Object & e = Object(),
          Node* p = NULL, Node* n = NULL) :
        element(e), prev(p), next(n) {};

};
Full code

Available in text, on 4.5

We'll see this node again when writing full list class.
Vectors (Ch 5.1)

like lists in python

myvector[6] = 6;

extendable: if array is too small, double it and copy everything

Time: $O(N)$ time for $N$ insertions

(not $O(1)$ time per operation)
Code:

```
template <typename ItemType>
class Vector {

private:

    int size;

    ItemType * data;  // points to an array

    int capacity;

};
```
Constructor:

```java
Vector (int cap = 100) : size (0), capacity (cap),
data (new ItemType[capacity])
```
Operator []:

```cpp
out << myVect[5];
```

```cpp
ItemType operator[](int index) {
    if (index >= 0 && index < size)
        return data[index];
    else
        raise error;
}
```
Destructor:

```cpp
~Vector () {
    delete [] data;
}
```
Insert: Examples

myVector.insert(2, 'c');

otherVector.insert(11, "new");

anotherVector.insert(7, -25);

Alice, Bob, Dan, Edward, Franky

insert Carol at position 2 →