CS180 - Linked Queues + Vectors

Note Title 9/27/2010

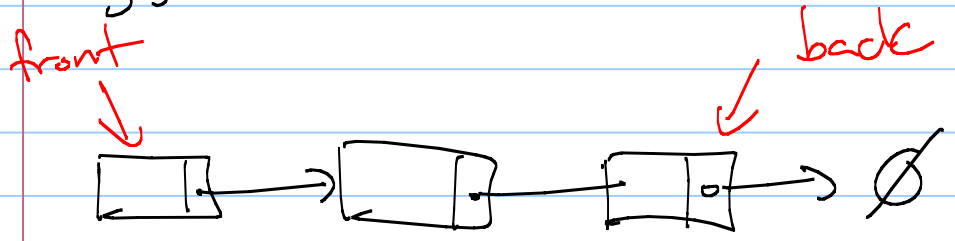
Announcements

- 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)

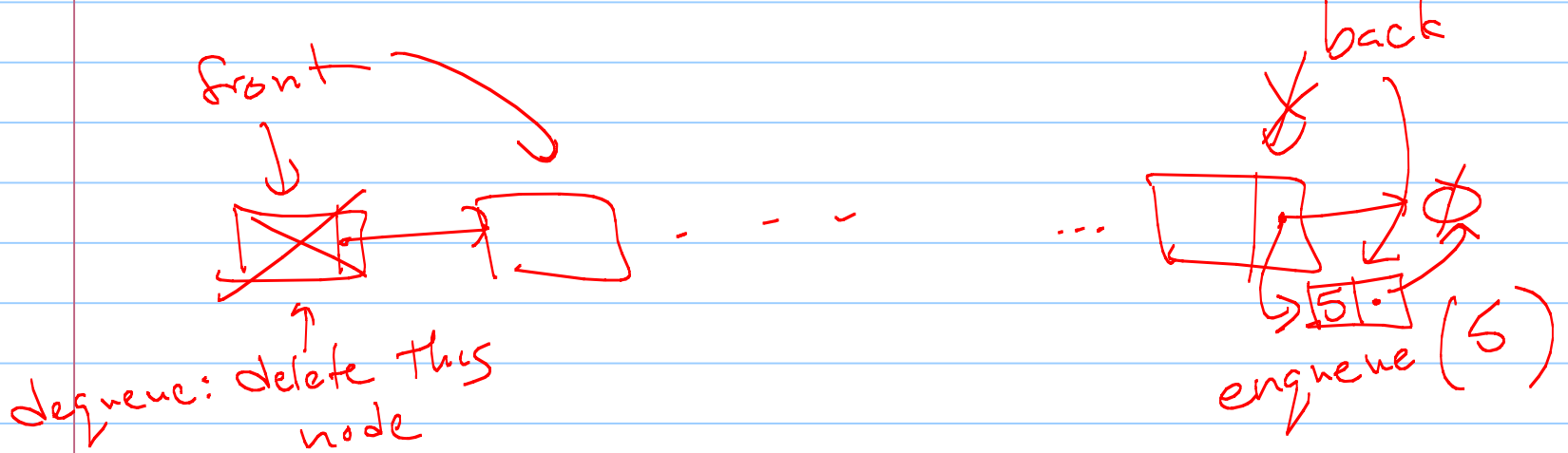
```
struct Node {
    Object element;      // value of this node
    Node* next;          // ptr to next node

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



private data:

```
Node*   front;
Node*   back;
int     size;
```



We coded:

- enqueue
- dequeue
- front
- removeAll
- copyFrom

] Helper fcns

What's left? House keeping

Housekeeping functions

3 things left:

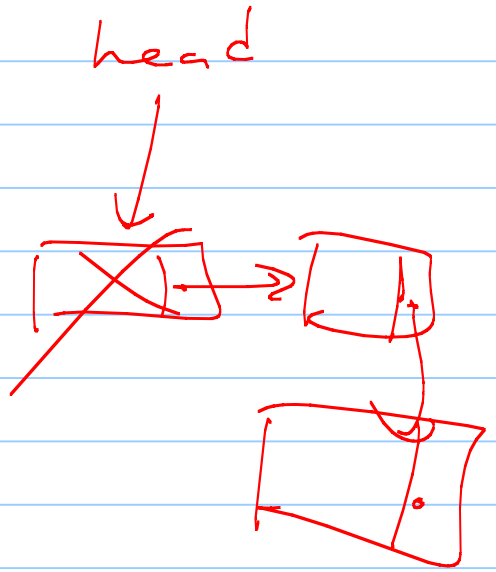
Copy Constructor

Destructor

Operator=

Destructor

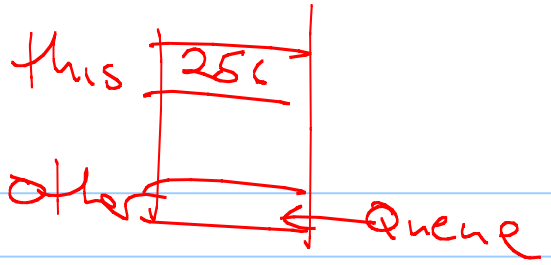
```
~LinkedQueue() {
    removeAll();
}
```



Copy Constructor :

```
LinkedQueue(const LinkedQueue& other){
    copyFrom(other);
}
```

Operator =:




```
LinkedQueue& operator=(const LinkedQueue& other) {
    if (this != &other) {
        removeAll();
        copyFrom(other);
    }
    return *this;
}
```

(end 4.4)

Doubly Ended Queues (Deques)

Deck

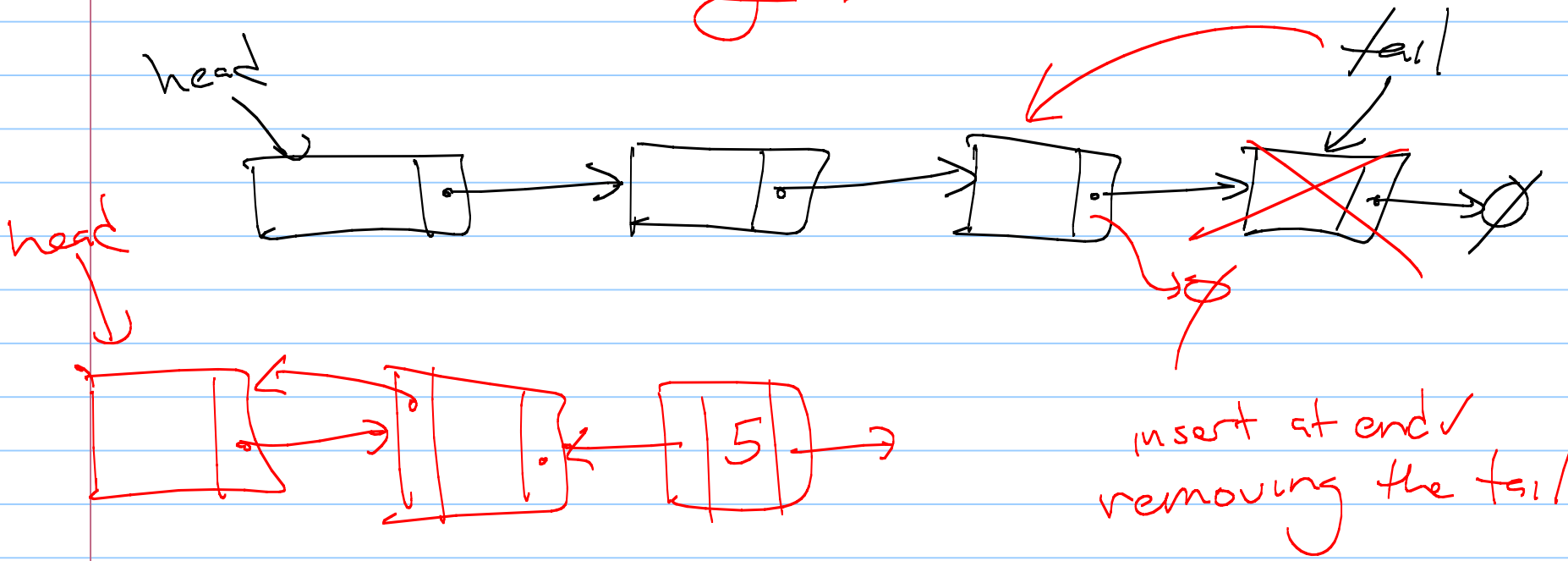
Like a queue, but allows insertion or removal from either end.

Supports:

- insertFirst
- insertLast
- removeFirst
- removeLast

Problem: Will our nodes work here?

Need Daddy Linked List



Solution = new node class

(this struc 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, Ch 4.5

We'll see this node again when writing full list class!

Vectors (Ch 5.1)

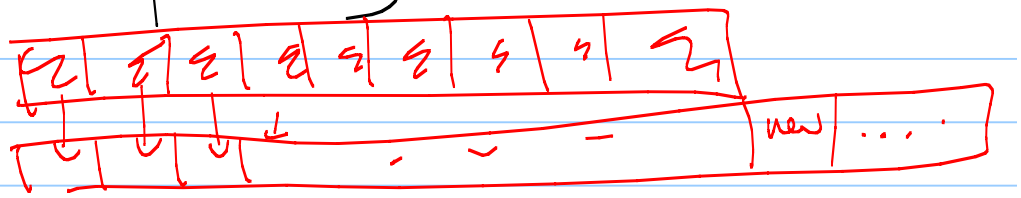
like lists in python

myvector[5] = 6;

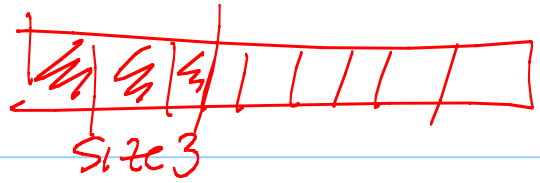
Extendable: if array is too small, double it & copy everything

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:

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

Operator []:

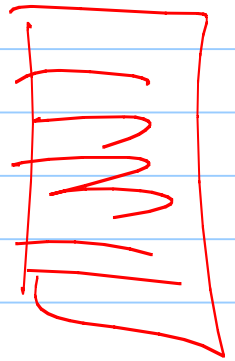
```
cout << myVect[5];
```

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

Destructor:

```
~Vector () {
    delete [] data;
}
```

data | 2700



Insert : Examples

```
myVector.insert(2, 'c');

otherVector.insert(11, "new");

anotherVector.insert(7, -25);
```

Alice, Bob, Dan, Edward, Frank

insert Carol at position 2