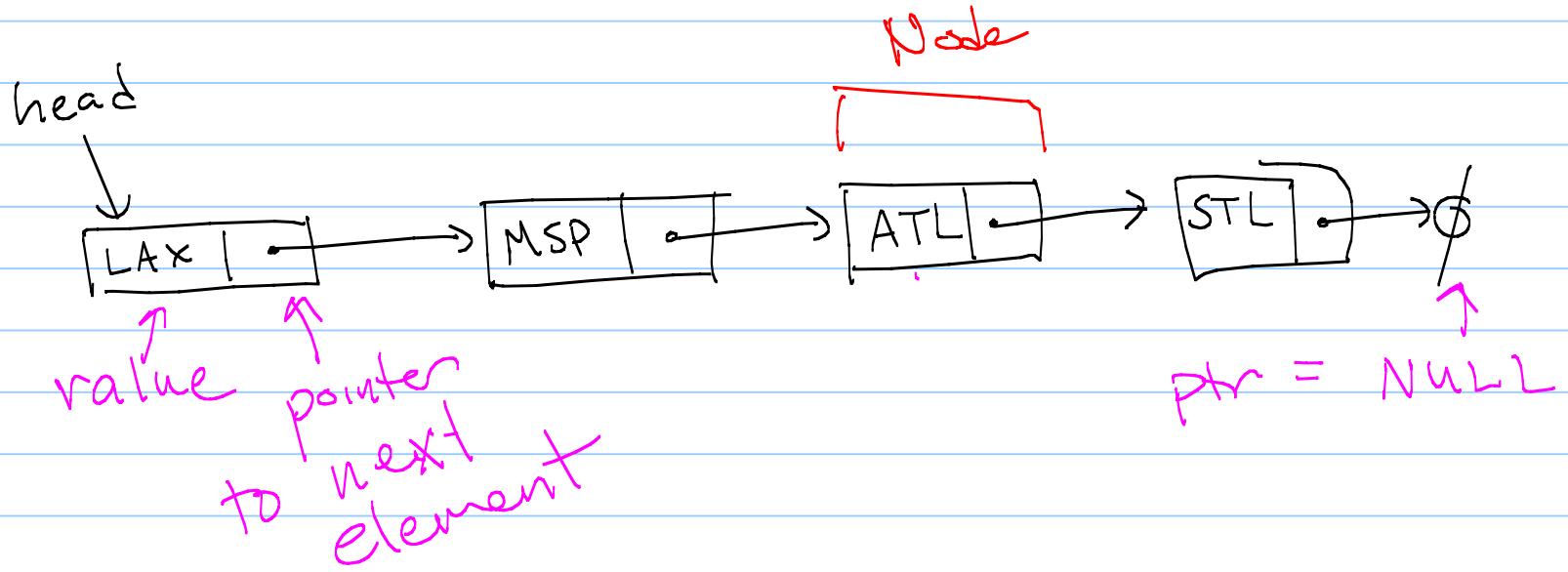
CS180 - Lecture 13

Note Title 9/28/2009

Announcements

- Program 2 up - due next Thurs.
  - in pairs, but not same pair as last time
  - checkpoint is next Monday
- One conflict exam left, so tests will be back (graded) Thursday or Friday
- Look for email with program 1 grade tonight

Last time - linked lists



Our Struct for a Node

```
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) {}

};
```

Linked Stack

A version of a stack which uses an underlying linked list.

Advantage:
- arbitrary size
- doesn't take up memory if empty

Disadvantage:
- takes extra space
- slower
- traversing list can take more time

hoping to avoid →

Code: for LinkedStack

Our node struct will be included as "protected" (instead of public/private).

Why?

Protected is essentially same as private, but is allowed to inherit.

Private data:

```
Node* tp;
int size;
```

Functions (Easy Ones)

Constructor:

```
LinkedStack() : tp(NULL), size(0) {}
```

size

```
int size() const { return size; }
```

empty

```
bool empty() const { return size == 0; }
```

Top function (2 versions)

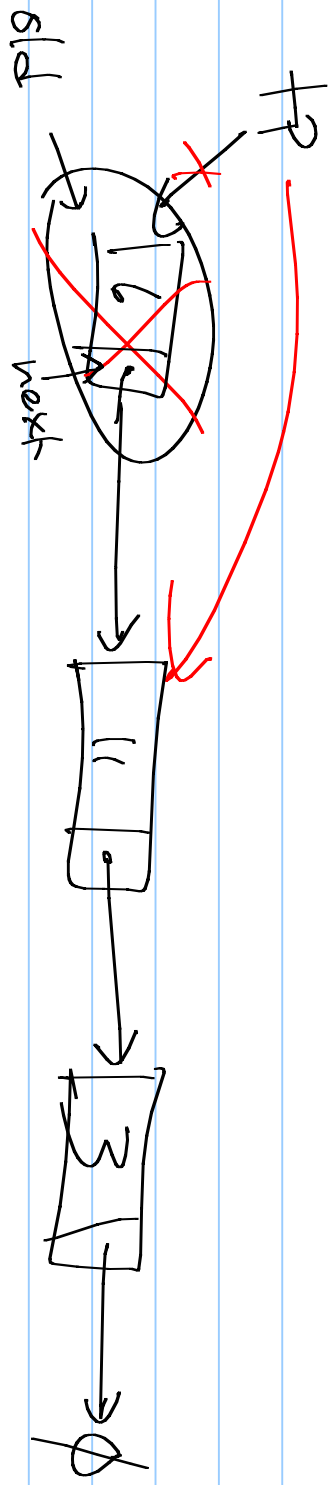
What is our choice here?

```
Object top() const {
    if (empty())
        throw std::runtime_error("Stack empty");
    return tp->element;
}
```

another way: (faster)

```
const Object& top() const {
    if (empty())
        throw std::runtime_error("Stack empty");
    return tp->element;
}
```

pop



Push & Pop

```
void push(const Object& e) {
    tp = new Node(e, tp);
    size++;
}

void pop() {
    if (empty())
        throw std::   (throw error-see top)
    Node* old = tp;
    tp = tp->next;
    delete old;
    size--;
}
```

"House keeping" functions

What else do we need to worry about?

- Destructor
- Copy Constructor
- Assignment operator