Announcements

- HW3 will come out this week.
- Lab 6 will be Wednesday the 23rd.
- Review session will be Monday.
- In class I will be there, start by 4:30.
- Midterm I is due Friday.

CS 180 - Lecture 9
Stack: list of objects. Supporting the following operations:

- push(0): insert a at top of stack
- pop(): remove and return top object from stack
- size(): return # of objects in stack
- isEmpty(): return true if stack is empty
- empty(): return true if stack is empty
- top(): return top object of stack without removing it
- list(): return a list of objects on stack without removing it

Input: none
Output: Object
- toP(): return the object on stack without removing it.
(Well use this for lab soon...)

Recommendation is available online.

Functions: push, pop, top, size, isEmpty

Stacks are one of the built-in classes in the Standard Template Library.
```cpp
3. int size() const;
   Object pop();
   Object top() const;
   bool is_empty() const;
   // Implement function
   template <typename Object>
   class Stack {
   public:
      void push(const Object& obj); // Template function

      // Implement...

      // Always worth planning the functions ahead of time.
   };
```

Our interface:
(see return hypers)

Should pop + top be different?

One complication: how should we return objects?
Some Pseudo Code:

```
Some Pseudo Code:
```
(with a few changes).

Based on code from text (p. 163).

Our code: available on webpage.
Size usage? $O(N)$

Object is size $O(1)$

Running Times:

<table>
<thead>
<tr>
<th>Function</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>push</td>
<td>$O(1)$</td>
</tr>
<tr>
<td>pop</td>
<td>$O(1)$</td>
</tr>
</tbody>
</table>

Size is empty? $O(1)$
double, since we need to recover
spend on each item stack
stack overflows.
Other options: double away every item
prefer a maximum size?
What is the mean dislocation?
See p. 166-168 for more detail.

Why is this data structure ideal?

Keep track of local variables.
Called 'the run-time stack'.
C++ actually keeps a private stack.

Sec. 4.2.3 - Function calls + stacks