Slide 1: Stacks

Most important things to know about stacks:

- First in, Last out (FILO)
- They are a *Collection ADT*, in that they maintain a collection of items.
- They can be implemented with any type of ordered list object, which right now includes lists, arrays, and vectors (stretchy arrays).
- Have many uses! (can you think of a few)

One of the most popular uses for stacks is parsing! As an example. . .

Example: Stack-based parsing of mathematical expressions.

One of the best ways to parse fully parenthesized (eg each separate operation is parenthesized) mathematical expressions is through using a stack. If you own an HP calculator, this'll be very familiar!

Here's what I mean by parsing expressions. How does the computer do the following?

(1 + (2 + (3 + (4)))) = 10

Pop Quiz: Look carefully at the above equation. What kind of function do you think we could write in order to solve problems like this?

Now, lets use a stack to solve the problem. Again, assume that each operation has been fully parenthesized—I.e., there won't be anything like this: (1 + 2 + 3). Instead, an operation of that type will look like this: ((1 + 2) + 3)

How would we do this??

Answer: Here's what our code should do:

- See open parentheses, push whatever is immediately after it onto the stack.
- Every time we see a closed parentheses, we need to evaluate the top two members of the stack, pop them, and replace them with the result.

Try it with a different problem:

((1 + 2) + (3 + (4 + 5))

Does our stack solution still work?