Let's look at solutions to the interclass problem.
What did we talk about last class?
Operators in C

- Last time we talked about expressions. They are constructs in C that have a value. With the exception of simple expressions they involve operators.
- **Numerical Operators**
  - Common operators: +, -, *, /
  - Unusual but makes sense: %, ++, --
  - Bitwise: &, |, ^, ~, >>, <<
- **Boolean**: &&, ||, !, ==, <, >, <=, >=, !=
- **Other**: [], f(), .., ->, sizeof, &, *, (cast), ?:, ,
- **Assignment operators**. We can make them by putting an = after any binary operator.
The assignment operators (=, +=, ...) and increment and decrement (++, --) not only return a value, they can alter the value stored in a part of memory.

Placing expressions with side effects inside large, complex expressions is risky.

If a single expression modifies the same memory location twice the results are undefined.

Normally I will never put more than one side effect in a single expression and I will use a consistent style so it is clear what I am doing.

Return value of assignments.
You have seen that C has quite a few different primitive types. Some represent integers/characters while others represent floats (the computer approximation to real numbers).

If you have an expression that involves more than one type, C will do implicit casting to change the types so they agree before doing the operation.

With non-assignments, types will be “promoted” and the whole expression will have the “larger” type.

In assignments the value has to have the type you are assigning into. “Demoting” casts can lose information.
Explicit Type Conversions

- We can also explicitly change the type of expressions with type casts.
- To do an explicit type cast we simply precede the expression with the type we want it to be in parentheses.
- This is helpful when you want to do something like divide two ints and store the result in a double. Without a cast the division will be integer division and fractional parts will be lost.
C programs are made up of statements which come in a number of different flavors.

- Null statement. (;)
- Expression statements. (expression;)
- Return statements. (return expression;)
- Compound statements. ({ [declarations] statements })
This is one of the most important skills you can learn in this class. I like to give tracing test questions.

Given what we know now, when a C program runs we start at the first line in main and execute one statement after another moving down through the code.

Note that assignment is not like equality in math. Doing an assignment simply stores a value in memory. It can be changed by later operations later in the program.
This was our last meeting discussing chapter 3. We now move beyond C basics. What questions do you have about the basic concepts in C?

Remember that the first assignment is due today. It needs to be posted by midnight tonight.

Interclass Problem – Do problem 29 on page 144.