if Conditionals

9/17/2007
Let's look at some solutions to the interclass problem.
Last class we talked about boolean expressions and the if statement.

The if statement can be used in one of the following forms.
- if(expression) statement
- if(expression) statement else statement

The expression can be any expression but it will be interpreted as a Boolean where 0 is false and everything else is true.

The statements can be compound statements using the {} syntax.
Today we want to run through how we use if statements in C.
Let's start with a basic example then build our way up to more complex ones.
We want to look both at complex Boolean expressions and nesting conditionals.
The if statement is a statement, not an expression. It has no value, it simply changes the way things are executed.

Sometimes it is helpful to have an expression that is conditional. This is provided in C with the operator `?:`.

The syntax is as follows:
- `conditional-expression ? then-expression : else-expression`

If the conditional expression evaluates to true, the whole expression have the value of the then-expression. Otherwise, it gets the value of the else-expression.
- Combining function calls and conditionals also allows us to begin exploring recursion.
- Recursive functions are simply functions that call themselves. The call must be conditional otherwise you have infinite recursion.
- To really see how recursion works we need to understand the call stack. This is a bank of memory on the computer the program uses to store variables and information related to what is happening in the program. Each time a function is called, a new stack frame is “pushed” with the memory that function needs. When the function returns the stack frame is “popped”.

Basic Recursion
Technically the combination of conditionals and function calls gives us all the power to need for full fledged computation. Why do you think C has more control structures for us to discuss?

Interclass Problem – Do problems 35 and 36 on page 294.