Week 2: Programming Logic

As humans most of our problem solving involves decisions. "If this then that." This is called a *conditional* because there is no guarantee of "this". For example if someone says, "If I have \$ 5, then I will order a beer." Well, that person may or may not have \$ 5.

Everything in programming basically boils down to loops and conditionals. Even loops are just conditionals.

Loops: Suppose we want to solve $\int_0^1 f(x)dx$ on the computer. The computer doesn't understand continuity so we must discretize:

$$\int_0^1 f(x)dx \approx \sum_{n=1}^{N} g(x,n). \tag{1}$$

We know the sum gets closer to the integral the bigger N is, so it could be a million or more. We can't possibly do this one at a time, so we use loops. (We will mainly use "for" loops because "while" loops may blow up.)

To find the sum lets write the algorithm in "pseudocode". This is code mainly written in natural language that can then be used to create code in the language we are using.

Algorithm 1 Pseudocode

```
for $n$ from 1 to $N$ do
    Sum = Sum $+g(x,n)$
end for
```

Notice that the way I wrote the pseudocode here is different from the way I wrote it on the board. We have a lot of flexibility when writing pseudocode.

Conditionals: Conditionals carry out logical operations. We already saw the syntax, but lets think about the logic. Consider a statement x and a statement y. We don't know if they are true or false, but it can only be one or the other. To analyze logical operations we can use "truth tables". These are tables where we explore all possibilities for the statements. For one statement there are two possibilities: either true or false. For two statements there are four combinations. We will write true as the boolean "1" (which is an on transistor), and a false as "0" (which is an off transistor).

The truth statement for the **NOT** operator will be:

x	**NOT** x
0 (false)	1 (true)
1 (true)	0 (false)

With the NOT operator, if x is one boolean value, **NOT** x is the opposite boolean value.

The truth statement for the **AND** operator will be:

x	y	x **AND** y
0	0	0
0	1	0
1	0	0
1	1	1

For the **AND** operator we have four combinations of boolean values. The statement x **AND** y is only true if both statements are true.

The truth statement for the **OR** operator will be:

x	y	x **OR** y
0	0	0
0	1	1
1	0	1
1	1	1

For the **OR** operator we again have four combinations of boolean values. The statement x **OR** y is true if either statement is true.

An example of pseudocode where we would use such operations is as follows: Here we are pretending that we can solve a

Algorithm 2 Pseudocode

```
if $x$ AND $y$ then
    Solve a Millennium problem
end if
```

Millennium problem if only some things came true. Unfortunately, to solve a Millennium problem many different statements have to be true at once, and us mere mortals have a hard time getting one or two to be true – this is represented by the **AND** operator.