Students studying for the AP Computer Science exam, starting in the Fall of 2014, should append this text at the end of Chapter Seven, Lesson One.

Boolean Algebra and De Morgan's Law

Boolean expressions share many of the same features as normal algebraic equations. The Commutative Law, Associative Law, Distributive Law, and Identity Law all hold true for Boolean algebra. If "A", "B", and "C" are all **true**/**false** Boolean values, or boolean expressions that evaluate to **true**/**false**, then the following table describes each law in terms of A, B, and C:

Law	Examples
Associative	((A || B) || C = A || (B || C) (A && B) && C = A && (B && C)
Commutative	A || B = B || A A && B = B && A
Distributive	A && (B || C) = A && B || A && C A || (B && C) = (A || B) && (A || C)
Identity	A || A = A A && A = A

What this means in terms of your code is that you can write your logical expressions as "A || B" or "B || A", for example, and the calculated results will be the same (subject to short-circuiting).

In addition, *De Morgan's Law* is a famous and useful rule that states the following expressions are equivalent:

De Morgan's Law
!(A || B) = !A && !B !(A && B) = !A || !B

This is a little confusing, so what does De Morgan's Law mean in English?

The first line says "the inverse of A OR B is the same as the inverse of A AND the inverse of B". Try plugging in a few true/false values for A and B and you will see the theorem holds true.

Similarly, the second line says "the inverse of A AND B is the same as the inverse of A OR the inverse of B."

Again you can prove this with a few examples. You may find it easier in your code to write expressions in one form or the other, so knowing the De Morgan's equivalents is very handy. Let's walk through a couple of examples to demonstrate use of De Morgan's law.

Simplification Exercise #1

Can we simplify this expression?

Result = A || ! (B && A)

Let's use DeMorgan's Law on the right side:

Result = A || (!B) || (!A)

We can use the associative law to re-arrange things:

Result = A || (!A) || (!B)

We know "A || !A" is always "TRUE"

Result = TRUE || (!B)

Now, TRUE || <anything> is always TRUE, so we can discard the second part completely!

Result = TRUE // no matter what A and B inputs contain

Simplification Exercise #2

Consider this complicated expression with X, Y, and Z:

Result = ! ((! X) || Y) && (! Z))

We can see this expression follows the pattern of DeMorgan's first law, ! (A && B), where A = ((!X) || Y) and B = (!Z). So we can rewrite the expression as:

Result = !((!X) || Y) || !(!Z)

The expression on the end, "!(!Z)" just reduces to "Z"

Result = !((!X) || Y) || Z

We can again apply DeMorgan's law to the left part, !((!X) ||Y)

Result = (X) && (!Y) OR Z

So we have taken an initial complex expression and used DeMorgan's Law to reduce it to something that is a little clearer and easier to evaluate.

