Administrative Stuff

Homework 2 was released online today. Turn it in by email and be sure to include the code for your programs as well as the comments/explanations that are required. The code can be in one large text file, or you can submit different files for each program. **The Due date is Feb 6th.**

Basic Boolean Logic (And/Not/Or)

'Variables' can be either **true** or **false**
'Operators' are commands that combine variables into more complex formulas. Depending on the operator used and the values of the variables involved (true or false), the formulas themselves will be either true or false.

In Perl, there are three logical operators:

‘Not’ - which is represented by an exclamation mark, ‘!’
‘And’ - which is represented by two ampersands, ‘&&’
‘Or’ – which is represented by two vertical lines, ‘||’

So if ‘A’ is a variable, let’s consider ‘!A’ (which means ‘Not A’):

‘!A’ is true when ‘A’ is false
‘!A’ is false when ‘A’ is true
Therefore, ‘!A’ is the opposite of ‘A’

Truth Tables

Instead thinking about Boolean operators in terms of written definitions, another way to think about logical operators is through the use of ‘truth tables’.

Truth Table for ‘!’ (Not):

To construct the truth table, first put all of the variables (just ‘A’) and formulas (‘!A’) in the first row. Then, fill out the possible combinations of ‘true’ or ‘false’ for the variables. Finally, use the values for the variables to derive whether the formula will be ‘true’ or ‘false’ in each case.

A	!A
F	T
T	F

- The formula and the variables are displayed in row one.
- The possible combinations of ‘true’ and ‘false’ are in the first column.
- The results of the formula are in the second column.

Just like our written definition, the truth table indicates that when ‘A’ is true, ‘!A’ is false (first row) and when ‘A’ is false, ‘!A’ is true (second row).

The ‘&&’ (And) Operator

The written definition for the formula ‘A&&B’ is that: ‘A&&B’ is true when both ‘A’ is true and ‘B’ is true.” Everything else is false.

Truth Table for ‘&&’ (And):

We construct the truth table in the same manner as the ‘!A’ truth table. List the variables and the possible values of the variables in the first columns. In the remaining columns list the formulas you want to evaluate and the results of those formulas.

A	B	A&&B
F	F	F
F	T	F
T	F	F
T	T	T

In order for ‘A&&B’ to be true, both ‘A’ and ‘B’ must be true. Hence, if either ‘A’ or ‘B’ is false, ‘A&&B’ will also be false. Thus, the first three rows of the truth table evaluate ‘A&&B’ to false, and the last row gives ‘A&&B’ as true, because both ‘A’ and ‘B’ are true.

The ‘||’ (Or) Operator
The written definition for the formula ‘A||B’ is that: ‘A||B’ is true when either ‘A’ is true, or when ‘B’ is true. If one or both of the variables is true, then ‘A||B’ is true.

Truth Table for ‘||' (Or):
For this operation, the first line evaluates ‘A||B’ to false because neither ‘A’ nor ‘B’ is true. The next three lines, however, evaluate ‘A||B’ to true because either ‘A’ or ‘B’ is true for each case.

A	B	A||B
F	F	F
F	T	T
T	F	T
T	T	T

An Aside: The ‘xor’ (Exclusive Or) function – NOT present in Perl!
In some programming languages, there is an ‘exclusive or’ function, ‘xor’. The formula ‘AxorB’ will evaluate to true when only one of ‘A’ or ‘B’ is true. In other words, ‘A’ or ‘B’ must be exclusively true. Thus, ‘AxorB’ will be false when ‘A’ and ‘B’ are both false or both true.

Truth Table for ‘xor’ (Exclusive Or):
As stated above, ‘AxorB’ will only be true when only one of ‘A’ or ‘B’ is true. Thus, only rows two and three will be true. Lines one and two will both be false.

A	B	AxorB
F	F	F
F	T	T
T	F	T
T	T	F

Inventing New Truth Tables
You can envision inventing a new logical operator by filling in the formula evaluations yourself. Let’s call our new logical operator the ‘ted’ operator. By changing false and true in the truth table column corresponding to the evaluation of the formula you can define the meaning of ‘ted’.

Truth table for ‘ted’ (???):

A	B	AtedB
F	F	?
F	T	?
T	F	?
T	T	?

Here, we can change the values in the formula column to whatever we want to make a new theoretical logical operator…

Some Example Formulas

Example 1 – ‘A||!B’
What does ‘A||!B’ mean? One way is to use the written definition of one of our basic operators, and substitute where necessary. In our definition of ‘A||B’ we replace ‘B’ with ‘!B’. This means:

‘A||!B’ is true when only ‘A’ is true or only ‘!B’ is true, or, when both are true. Alternatively, ‘A||!B’ is true when only ‘A’ is true or only ‘B’ is false, or, when ‘A’ is true and ‘B’ is false. Both written definitions have the same meaning.

Truth Table for ‘A||!B’ (A or Not B):
As an exercise, we can use our original truth table for ‘A||B’ to find the evaluation of ‘A||!B’. We do this by expanding the table for ‘A||B’ with a third column marked ‘!B’. Knowing that ‘!B’ is the opposite of ‘B’ we can fill in column ‘!B’ very easily.

A	B	!B	A||!B
F	F	T	?
F	T	F	?
T	F	T	?
T	T	F	?

Now, we can simply use the ‘A’ and ‘!B’ columns to evaluate the ‘||’ (Or) Operator. So when either ‘A’ or ‘!B’ is true, or when both are true, ‘A||!B’ will be true

A	B	!B	A||!B
F	F	T	T
F	T	F	F
T	F	T	T
T	T	F	T

Example 2 – ‘!(A&&B)’
‘!(A&&B)’ is the same as the opposite of ‘A&&B’. Thus, when ‘A&&B’ is false, ‘!(A&&B)’ will be true. So, when is ‘A&&B’ false? We can expand on our A&&B truth table by adding a column for ‘!(A&&B)’.

Truth Table for '!(A&&B)':
Thus, the evaluation of '!(A&&B)' will be the opposite of 'A&&B'.

A	B	A&&B	!(A&&B)
F	F	F	T
F	T	F	T
T	F	F	T
T	T	T	F

Example 3 – '(!A)||(!B)'
This example says that when only 'A' or only 'B' is false, or, when both 'A' and 'B' are false, then '(!A)||(!B)' is true.

Truth Table for '(!A)||(!B)':
In rows 1, 2, and 3 either 'A' or 'B' is false. Thus, '(!A)||(!B)' will evaluate to true in each case.

A	B	(!A)||(!B)
F	F	T
F	T	T
T	F	T
T	T	F

Notice that the truth table for '!(A&&B)' is the same as the truth table for '(!A)||(!B)'. This fact is stated in **'DeMorgan's Law':

!(A&&B) ≡ (!A)||(!B), and,
!(A||B) ≡ !A&&!B
(The '≡' means 'equivalent to')

<u>**Properties of Boolean Operators**</u>

The Identity Property – 'A ≡ A&&A'
This truth table can be written easily. If 'A' is true, then 'A&&A' is true.

A	A&&A
T	T
F	F

The Commutative Property 'A&&B ≡ B&&A'
The truth table for 'A&&B' and 'B&&A' shows that the two are equivalent.

A	B	A&&B	B&&A
F	F	F	F
F	T	F	F

T	F	F	F
T	T	T	T

The Contradiction – 'A&&!A is always false'
Since 'A' can never be the same as '!A' there is no way for 'A&&!A' to be true. It is a contradiction because no matter what you choose for 'A', the formula evaluates to false.

A	!A	A&&!A
F	T	F
T	F	F

The Tautology – 'A||!A is always true'
Since 'A' is necessarily different than '!A', one is always true. Thus, 'A||!A' is always true. It is a tautology because no matter what you choose for 'A', the formula evaluates to true.

A	!A	A||!A
F	T	T
T	F	T

Three Variable Examples

Example 1 – 'A&&(B||C)'
We first start out by writing all of the variables in the first columns and filling in all the possible combinations for true and false.

A	B	C	A&&(B||C)
F	F	F	?
F	F	T	?
F	T	F	?
F	T	T	?
T	F	F	?
T	F	T	?
T	T	F	?
T	T	T	?

Since this is an 'And' statement, both conditions must be true. Since the first condition states 'A' is true, the first four rows all evaluate to false. The second condition of 'A&&(B||C)' states that either 'B' or 'C' must be true. Thus, row five is false. The last three rows all have 'A' true, and either 'B' or 'C' true. Thus, both conditions are satisfied and 'A&&(B||C)' is true.

A	B	C	A&&(B||C)
F	F	F	F
F	F	T	F
F	T	F	F
F	T	T	F
T	F	F	F

T	F	T	T
T	T	F	T
T	T	T	T

Example 2 – '(A&&B)||(A&&C)'
Again start out by writing all of the variables in the first columns and filling in all the possible combinations for true and false.

A	**B**	**C**	**(A&&B)||(A&&C)**
F	F	F	?
F	F	T	?
F	T	F	?
F	T	T	?
T	F	F	?
T	F	T	?
T	T	F	?
T	T	T	?

This formula will have the same evaluation as A&&(B||C). Because this is an 'Or' statement, if either condition is true, then the entire statement is true. Both sides of the 'Or' have the requirement that 'A' must be true. Thus, the first four rows are again false. In addition, for the first condition to be true, 'B' has to be true, and, for the second condition to be true 'C' has to be true. Thus, the formula will evaluate to true when 'A' is true and either 'B' or 'C' is true. Thus, the fifth row is false, and the last three rows are again true for the same reasons outlined above.

A	**B**	**C**	**(A&&B)||(A&&C)**
F	F	F	F
F	F	T	F
F	T	F	F
F	T	T	F
T	F	F	F
T	F	T	T
T	T	F	T
T	T	T	T

The equivalence of examples 1 & 2 is an example of the **Distributive Law:**

A&&(B||C) ≡ (A&&B)||(A&&C)