PROGRAMMING IN LOGIC

It is often instructive to have students program some of the logical
truth tables they encounter in courses on logic or discrete structures.
Most programming languages, such as Java have few logical operators
(usually And, Or, Not) but do not have Nor, Nand, Implication, etc.
Programming of even a simple truth table is not trivial in such languages.
The following project, on Dilemmas, is a programming challenge.
More on such Logic, especially in Java, can be found at:
`http://www.csun.edu/~jmotil/comp110/Logic/LogicInProgramming.pdf`

Other possibly interesting material involving unusual number bases is at
`http://www.csun.edu/~jmotil/comp110/Logic/logicBases.pdf`

Enjoy

John Motil, Prof Emeritus
Cal State University, Northridge
jmotil@csun.edu

<snip here for student assignment that follows>

PROGRAMMING LOGIC: DILEMMAS
Dilemmas describe conflicts, which are logically interesting.
The following example illustrates the concept of a dilemma.

If you accept certain advice then you feel dominated.
If you reject the advice then you feel guilty.
But you must either accept the advice or reject it.
So therefore you feel dominated or guilty.

The general form of dilemmas is as follows (not entirely in Java),
where c and d are usually inescapable, unavoidable consequences.

```
a -> c   // If a then c
b -> d   // If b then d
a or b   // a or b
------   // therefore
c or d   // c or d
```

The corresponding formula should be a tautology (always true).

```
((a -> c) & (b -> d) & (a ^ b)) -> (c | d)
```

There are two forms of the Or; the inclusive (|) and exclusive (^).
This leads to four forms of the dilemma, not all are proper.
Find out which of the four are indeed proper. Guess first!

To create by hand all 4 truth tables of 16 rows each is exhausting
(but you may wish to do at least one of these interpretations).
To create a program to do this is challenging. Try it.

There are many ways to do this; all involve multi nested loops.
One way is use built-in logic and values true and false where x-->y is !x|y.
Another is to use values 0, 1 and define static functions such as xor(x,y).
Yet another is to use character values 'T' and 'F' and define functions.
And another is to use string constants "TRUE" and "FALSE".
Finally another is to create a logic class with functions such as x.implies(y).
You might try two of these ways, and compare them.

Hint: You could just begin with a simpler formula, such as DeMorgan's Law
which involves only two logical variables, and a double nest of loops,
creating a table of only 4 rows. Then going to 4 variables and 16 rows is simpler.