Assignment 2—Due Monday, November 6

"The Incredible ecs110 Truth Machine!"

A **letter** is any of the symbols `a`, ..., `z`, ... `A`, ..., `Z`. A **digit** is any of the symbols `0`, ..., `9`. A **variable** is a letter followed by zero or more digits. A string is said to be **whitespace** if it is a sequence of spaces, tabs, and newlines.

The following strings are all of the **boolean formulas**:

1. A **variable** is a boolean formula.
2. If s is a boolean formula, then `not` s is a boolean formula.
3. If s and t are boolean formulas, then s `and` t is a boolean formula.
4. If s and t are boolean formulas, then s `or` t is a boolean formula.
5. If s and t are boolean formulas, then s `implies` t is a boolean formula.
6. If s and t are boolean formulas, then s `equals` t is a boolean formula.
7. If s is a boolean formula, then (s) is a boolean formula.
8. If s is a boolean formula and w is whitespace, then ws and sw are boolean formulas.

Here are some examples. The period marks the end of each formula. It is not a part of the formula.

```
B or not B.
P and Q   equals   not (not P or not Q).
A and (A implies B)   implies    B.
(P and Q) or not P.
(x1 or x2 or not x3) and (not x1 or x4 or not x5 )    and
         (x3 or x1 or x4).
```

Each variable of a formula can be **true** or **false**. A setting of each of the variables to **true** or **false** is called an **assignment**. Each assignment makes the formula itself turn out **true** or **false**. In evaluating a formula on an assignment we understand `not` to mean logical negation; it is the highest priority operator. We understand `and` to mean logical conjunction; it is the next highest priority. We understand `or` to mean logical disjunction; it is the next highest priority. We understand `implies` to mean logical implication[1]; it is the next highest priority. We understand

[1] Remember that P `implies` Q is defined as `not` P `or` Q.

`equals` to mean logical equality; it is the lowest priority. Equal-priority operators group evaluate left-to-right. Precedence can be changed with the use of parentheses.

Boolean formula ϕ is a **tautology** (or **tautological**) if *every* assignment makes ϕ come out to be **true**. Above, the first three formulas are tautological; the next two aren't.

You are to write a program to decide if a formula is a tautology. Don't worry about making your program run fast (no one knows how to do that). Your program goes into a loop which does the following: read a textual formula; convert it into the internal representation of a formula (a tree of some sort); print out a description of this tree (you may choose the notation); decide if the formula is tautological; display the answer. If the formula you read IS tautological, just say so; if the formula you read is NOT tautological, print an assignment which proves this. To allow for long formulas to be easily input, terminate each formula by a "`.`" (not a carriage return, which simply counts as whitespace). When your program reaches end-of-file (entered as control-D in UNIX), have your program terminate.

Your program should work on all inputs; if the formula entered is not well-formed, your program should say so, scan until the next "`.`" is consumed, and continue.

You will find it convenient to put your test cases in a file and redirect this as input. A file of test cases will be posted to the newsgroup and our class web page. You must run your program on this file.

As always, turn in a 1–2 page writeup with your solution. The contents of this writeup, as well as other expectations, was described in "Guidelines on Grading of Assignments," posted to the newsgroup and dropped to our course web page.

Have fun! Start early and start often!