CS381K Artificial Intelligence
Homework 2, due Nov 1st, 2000

Resolution Theorem Prover

Write a resolution theorem prover in Lisp according to the following specifications.

1. Representation: Use a list to contain the set of clauses. Initially, the list will contain the premise clauses and clauses derived from the negated conclusion. As resolvents are generated from these clauses, they are added to the end of the list.

   Each individual clause could be represented as a list of two lists:
   `((negative literals) (positive literals))`
   or by using a `not` operator for negative literals (choose one way to do it).

   Each literal will be of the form: `(<predicate-name> <term> ... <term>)`

   An atomic term is considered to be a variable; a non-atomic term must be a function of the form: `(<function-name> <term> ... <term>)`

   A constant is represented as a function of no arguments. Thus, our notation does not contain any symbols for logical connectives; these and the meanings of named symbols are determined implicitly by the positions of elements of a list structure. The input to your program will be clauses written in this syntax. For example, the clauses representing the statements 'All hounds howl' and 'John has either a cat or a hound' would be written in one of the ways shown below:

```
(  ( ( (HOUND X) )  ( (HOWL X)           ) )
   ( NIL            ( (HAVE (JOHN) (A)) ) )
   ( NIL            ( (CAT (A))
                      (HOUND (A))        ) )  )

(  ( (NOT (HOUND X)) (HOWL X) )
   ( (HAVE (JOHN) (A)) )
   ( (CAT (A)) (HOUND (A)) ) )
```

   For simplicity, you may assume that the sets of symbols used as predicate names, function names, and variable names are disjoint in any given problem.

2. Print out each input clause and each clause which is produced; number the printed clauses and show the numbers of the clauses from which they are derived. (Remember that more than one resolution of a given pair of clauses may be possible. However, we will not worry about resolving on factors of clauses.)

3. Use the unification algorithm described in Rich & Knight (and in the lecture notes). Note that the clauses must be rewritten so that they have no variables in common; a new variable can be created in Lisp by the function calls: `(intern (symbol-name (gensym)))`

4. Use the two-pointer method to select pairs of clauses for resolution (see `/u/cs381k/resob.lsp`); initialize the pointers so that you will be using the set-of-support strategy. This can be done in the following way:

   (a) Initialize: Set the "inner loop" pointer to the front of the list of clauses. Set the "outer loop" pointer to the first clause resulting from the negated conclusion.

   (b) Resolve: If the two clauses denoted by the two pointers can be resolved, produce the resolvent. Add the resolvent to the end of the list of clauses and print it out. (Note: it may be advantageous to check whether the resolvent is already present in the list of clauses; if so, it need not be added.) If the resolvent is empty ($\Box$), stop; the theorem is proved. Note that there may be multiple ways to resolve a single pair of clauses.

   (c) Step: Move the "inner loop" pointer forward one clause. If the "inner loop" pointer has not reached the "outer loop" pointer, go to the Resolve step. Otherwise, reset the "inner loop" pointer to the front of the list of clauses and move the "outer loop" pointer forward one clause. If the "outer loop" pointer goes beyond the last clause in the list, stop; the theorem cannot be proved.

5. Use the 'howling hounds' problem as a test problem.

6. In addition, test your unification algorithm by trying to resolve the following pairs of clauses (stated just as they appear below; the variable G0002 is one possible variable name after renaming).

Inputs:	Resolvent:
$P(x) \lor R(x)$ $\neg P(x) \lor Q(x)$	$Q(x) \lor R(x)$
$P(f(g(h(x)))) \lor R(x)$ $\neg P(f(g(y))) \lor Q(y)$	$R(x) \lor Q(h(x))$
$P(f(g(h(x), x))) \lor R(x)$ $\neg P(f(g(y, a))) \lor Q(y)$	$R(a) \lor Q(h(a))$ (a is a constant)
$P(x, f(x)) \lor R(x)$ $\neg P(f(x), x) \lor Q(x)$	none (does not unify)
$P(f(x), x) \lor R(x)$ $\neg P(y, f(x)) \lor Q(x) \lor S(y)$	$R(f(\texttt{G0002})) \lor Q(\texttt{G0002}) \lor S(f(f(\texttt{G0002})))$

7. Select one of examples 3–10 from the resolution excercise handout and solve it with your prover.

8. (Extra) Implement some of the preprocessing techniques and more intelligent search strategies. The "two-pointer method" is a breadth-first method and will generate many duplicate clauses.