CS 140 Logic Programming
Fall 2006
Machine Assignment #1
Due September 18

First read my paper that appeared in *Comm.* ACM, December 85.
You may choose any language for implementing this assignment
(e.g., Scheme, Java.)

Write a program that implements the unification of trees. Obviously, the program will have recursive calls.

Trees will be input using lists (a la Lisp or Scheme)

For example, the expression:

list (X, list (a, Y, Z)) *(1)*

can be viewed as a tree. And so does:

list (p, list (a, U, b) *(2)*

Please follow the following recommendations:

- Lists to be unified may be input in the program itself (as constants), to avoid having to develop procedures that read lists. If you are able to read lists automatically so much the better.

- If you use Java, lists can be implemented as binary structures or records.

- The function *unify* should have 3 parameters: the 2 lists to be unified and a variable *B* that is assigned to **true** if unification succeeds; otherwise *B* is **false**.

- The result of the unification is expressed as a list of pairs *(Variable, Value)*. For example, when unifying (1) and (2) your program should yield the list:

  list (list (X,p), list (Y,U), list (Z,b))

  the call to obtain the above is simply:

unify (*list (X, list (a, Y, Z)) , list (p, list (a, U, b), B)*
and *B* will be assigned the value **true**

- Your program should handle lists of arbitrary depths or lengths.

- For those of you using Java or any other imperative language, you will have to write a recursive program that prints lists as shown in the example.

- Test your program with several lists, including one in which you unify *X* with *f(X)*. Report what will happen in that case.

Write an extension of *unify* using the *occur* test so that infinite trees are filtered and the result of the unification is simply **false** when dealing with those trees.