Programming Language Principles

Solution to Homework #12

1. No, the two lists do not unify. Unifying them would require first that [X, b] unify with [c, Y]. They do unify, but X is bound to c, and b is bound to Y. Next, X would have to unify with d, which it can't, because it is already bound to c. Changing the first list to [ [X, b], c, a] would solve the problem.

2. Cousin(X, Y) :-GrandParent(Z, X), GrandParent(Z, Y), Parent(Z, X), not Parent(Z, Y).

3. Editor(Person, Book) :
   Seen(Person, Book, OtherBook1),
   Seen(Person, Book, OtherBook2),
   DifferentBooks(Book, OtherBook1),
   DifferentBooks(Book, OtherBook2),
   DifferentBooks(OtherBook1, OtherBook2).

   Seen(Person, Book, OtherBook) :-
   AuthorOf(OtherBook, Person),
   SubjectOf(Book, Sub),
   SubjectOf(OtherBook, Sub).

   Seen(Person, Book, OtherBook) :-
   Reviewer(OtherBook, Person),
   AudienceOf(Book, Aud),
   AudienceOf(OtherBook, Aud)

4. The proposed change works fine on a uniprocessor, but on a multiprocessor we need multiple "current running thread"s.

5. A single lock is easy to code and will work quite well so long as the number of processes contending for simultaneous access is not large enough to make the table a bottleneck. Multiple locks will permit a higher degree of concurrency, but they incur higher overhead in the no-contention case, since several of them may have to be acquired and released to perform a single operation. Moreover, it may be tricky to write the multiple-lock version correctly: if for example one process tries to lock buckets A and B, in that order, while another tries to lock B and A, in that order, deadlock may result.