Exercise 17.8

1. One example that would conflict with the above query is:

   Select eid From Emp Where salary = 10,000

   Suppose there are two employees who both have salary of 10,000. If the update command is carried out first, neither of them will be selected; if the select command is done first, both of them should have been selected. However, if these two were run at the same time, one tuple might be updated first causing one to be selected while the other is not. By locking tuples, it is ensured that either all the tuples are updated first or all the tuples go through the selection query first.

2. One example that would conflict with the above query is:

   Insert (EID, "Santa", AGE, SALARY, DID) Into Emp

3. If there is an index on a particular field of a relation, a transaction can obtain a lock on one or more index page(s). This will effectively lock all the existing records with the field having a certain range of values, it will also prevent insertion of new records with the field value in that particular range. This will prevent the so-called phantom problem. This technique is called **index locking.**

   It is clear to see that index locking could solve the problem mentioned in 2.

Exercise 17.11

1. IS on D; IS on F2; IS on P1200; S on P1200:5.
2. IS on D; IS on F2; IS on P1200, S on 1201 through 1204, IS on P1205; S on P1200:98/99/100, S on P1205:1/2.
3. IS on D; S on F1
4. IS on D; IS on F1; S on P500 through P520.
5. IS on D; S on F1 (performance hit of locking 970 pages is likely to be higher than other blocked transactions).
6. IS and IX on D; SIX on F1.
7. IX on D; IX on F2; X on P1200.
   (Locking the whole page is not neccessary, but it would require some reorganization or compaction.)
8. IX on D; X on F1 and F2.
   (There are many ways to do this, there is a tradeoff between overhead and concurrency.)
9. IX on D; X on F1 and F2.

There is a bug to the solution for 6. It should be IX on D; SIX on F1.