Assignment #7
Due Thursday November 30, 5:00 PM

Problems to Work and Hand In

The following questions relate to the "Beer Drinkers' Relational Database Schema" below. Note: this modification of the beer-drinkers' DB should not be confused with the example DB the TA's used in their lectures; there are some differences.

```
Likes(drinker, beer, rating)
Serves(bar, beer, price)
Frequents(drinker, bar)
LocatedAt(bar, addr)
```

1. (10 pts.) Define the following views:

   (a) A view whose triples (x, y, z) are all those drinkers x, beers y, and bars z such that x likes y, z serves y, and x frequents z.

   (b) The same as (1a), but the view also has a triple $(x, y, \texttt{NULL})$ if x likes y, but there is no bar z such that x frequents z and z serves y.

2. (20 pts.) Give domain declarations for drinkers and beers. Then use these in declarations (`CREATE TABLE` statements) for relations `Likes` and `Serves`. You have flexibility about types, as long as the type makes sense (e.g., beers are not integers). As you do, include the following constraints (of SQL2, not the Sybase system we are using for the PDA project) into the declarations:

   - `drinker` and `beer` form the key for `Likes`.
   - `bar` and `beer` form the key for `serves`.
   - No beer may be priced at less than $1.
   - Every bar mentioned in `Serves` must appear in the `LocatedAt` relation.
   - The average rating of any beer served at any bar must be at least 0.5. Note: you must express this constraint as a `CHECK` in the `Serves` relation schema to receive full credit.

3. (10 pts.) Write the following constraints as `CREATE ASSERTION` statements about the beer drinkers' database schema.

   (a) No drinker likes a beer that isn't served at any bar he/she frequents.

   (b) Drinker "Sharon Stoned" only frequents bars that have the minimum price for some beer (not necessarily a beer that Ms. Stoned likes).

Seventh Step of Your Personal Database Application

1. (10 pts.) For your PDA schema, define at least two views, one updatable and one non-updatable. For each one, define the view, write at least two queries that use the view, and (attempt) to update the database by updating the view. Show a single script in which the views are defined and the updates are attempted against those views, including the system's response.

2. (50 pts.) Build a user-friendly interactive application program front end to your personal database using C (C++ does not appear to be supported by the present Sybase system). Your program should consist of a continuous loop in which:

   1. A list of alternative options is offered to the user. One of the alternatives should be *quit*.
   2. The user selects an alternative.
   3. The system prompts the user for appropriate input values.
   4. The system accesses the database to perform the appropriate queries and/or modifications.
   5. Data or an appropriate acknowledgment is returned to the user.

   You should include both queries and modifications in your menu. For example, if your PDA were the running class example of bank customers and accounts, we would expect to find on your menu:

   - A number of useful queries on the database, with both input and output in a format more convenient and pleasing than raw interactive SQL. These queries might include a balance inquiry given an account number, an inquiry for a customer's address and/or phone given a name or social-security number, or a list of the accounts with negative balances.
   - Some updates to the database, such as a credit or debit given an account number, a change of a customer's phone, and so on.

   You should use embedded SQL commands in your program, as described in Handout 18: "Embedded SQL and Stored Procedures in Sybase." Please turn in your code along with a script showing an interaction with your program in which all of its features are exhibited.