CSE 444 Midterm Test

Spring 2007

Name: _____

Total time: 50'

Question 1	/40
Question 2	/30
Question 3	/30
Total	/100

1 SQL [40 points]

Consider a database of social groups that allows people to become members of groups: a person can be a member of several groups and each group maintains a list of pictures that are accessible to all members. In addition to the groups, the database also maintains a list of friends. The schema is:

```
MEMBER(personName, groupName)
PICTURE(groupName, picture)   /* picture = primary key */
FRIEND(personName1, personName2)
```

`PICTURE` stores for each picture the name of the group that owns that picture. The `FRIEND` table is symmetric, i.e. if X is friend with Y then Y is friend with X. Every person is a member of at least one group.

1.1

Write a SQL query that computes for every person the total number of pictures they can access through their group memberships. That is, a person X can access a picture Y if X is a member of some group Z and Z owns the picture Y. You need to turn in a SQL query that returns a result like this:

```
'Fred', 12
'Joe', 7
'Sue', 0
'Rick', 9
 ...
```

```
SELECT personName, COUNT(*)
FROM member LEFT OUTER JOIN picture
ON member.groupName = picture.groupName
GROUP BY (personName)
```

Another solution:

```
SELECT personName, COUNT(DISTINCT picture)
    FROM member LEFT OUTER JOIN picture
    ON member.groupName = picture.groupName
    GROUP BY (personName)
```

Note: If you use normal join instead of `LEFT OUTER JOIN` then you will miss those people who cannot access any picture (e.g. Sue in the example above).

1.2

A "cool person" is one that has at least 40 friends. Write a SQL query that returns all the cool persons in the database. You need to turn in a SQL query that computes a list of names.

```
SELECT personName1
FROM friend
GROUPBY personName1
HAVING (COUNT(*) >= 40)
```

Another solution:

```
SELECT personName1
    FROM friend
    GROUPBY personName1
    HAVING (COUNT(DISTINCT personName2) >= 40)
```

1.3

The marketing department has decided to recommend people to subscribe to additional groups. However, they do not want to issue phony recommendations. They would like to recommend to a person X to subscribe to a group Y if all X's friends are members of the group Y, but X is not a member of Y. Write a SQL query that computes for each person X the set of groups to recommend that that person subscribes. You need to turn in a SQL query that returns a list of (person, group) pairs.

```
SELECT m.personName, g.groupName
FROM member m, member g
WHERE NOT EXISTS
       ( SELECT *
         FROM friend f
         WHERE m.personName=f.personName1
          and NOT EXISTS
              ( SELECT *
                FROM member mf
                WHERE mf.personName = f.personName2
                 and  mf.groupName = g.groupName))
EXCEPT (SELECT * FROM member)
```

1.4

Create a new table `ACCESS(personName, picture)` that lists for each person the list of pictures that they can access. A person X can access a picture Y either if X belongs to a group that owns Y, or of X has a friend Z who belongs to a group that owns Y. Write SQL statements that insert the corresponding tuples in `ACCESS`. You need to turn in one or more `INSERT` queries.

```
CREATE TABLE ACCESS (
personName VARCHAR(50) REFERENCES (member.personName),
picture INT REFERENCES (picture.picture))

INSERT INTO ACCESS
SELECT DISTINCT m.personName, p.picture
FROM member m, member m1, picture p, friend f
WHERE  m.groupName = p.groupName OR
(m1.groupName = p.groupName AND
 m1.personName = f.personName2 AND
 m.personName = f.personName1)
```

Another solution:

```
INSERT INTO ACCESS
   SELECT DISTINCT m.peronsName, p.picture
   FROM member m, picture p
   WHERE m.groupName = p.groupName

INSERT INTO ACCESS
   SELECT DISTINCT f.personName1, p.picture
   FROM friend f, member m, picture p
   WHERE f.personName2 = m.personName
     and m.groupName = p.groupName
```

Note: In the second solution, some tuples may be inserted more than once (why?). To fix it, change the second query to

```
INSERT INTO ACCESS
   SELECT DISTINCT f.personName1, p.picture
   FROM friend f, member m, picture p
   WHERE f.personName2 = m.personName
     and m.groupName = p.groupName
     and not exists (select * from access a where f.personName1=a.personname and ...)
```

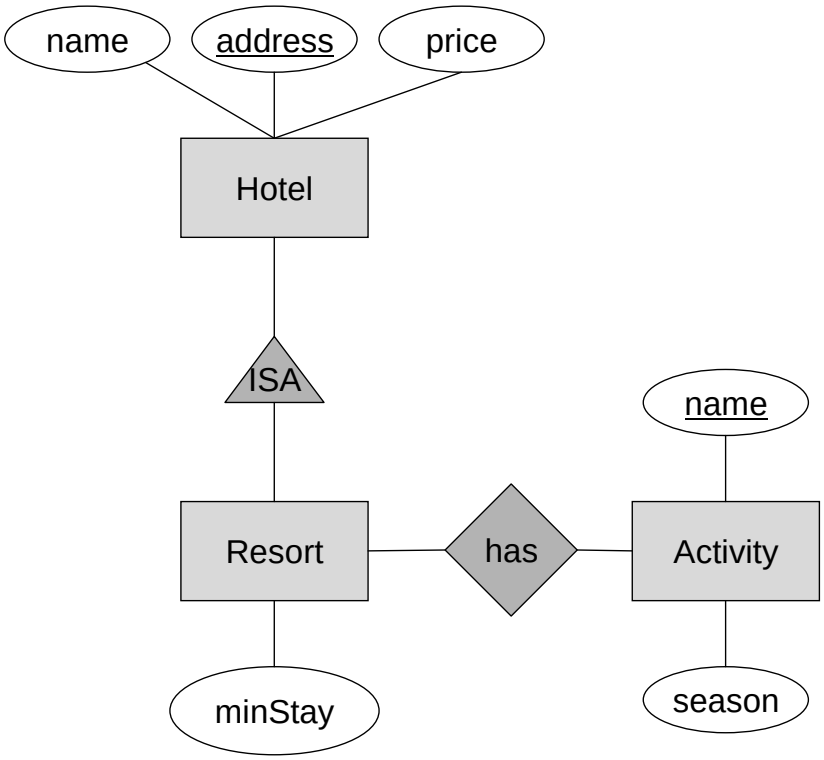
2 Conceptual Design [30 points]

Consider an application that needs to manage data for a travel agency. It needs to store the following entities and relationships:

- **Hotels:** have attributes name, address, price
- **Resorts:** are Hotels, that also have an attribute minimum-stay
- **Activities:** have attributes name, season
- **Has:** is a relationship between Resorts and Activities

2.1

Design an E/R diagram for this application.



Assumption: activities are uniquely identified their names (you could make other assumptions; it's OK as long as you stated them clearly).

2.2

Write the `CREATE TABLE` statement for creating the SQL tables. You may choose very simple atomic datatypes for the attributes. Indicate all keys and foreign keys.

```
CREATE TABLE hotel (
name VARCHAR(50),
address VARCHAR(100) PRIMARY KEY,
price INT
);
CREATE TABLE resort (
address PRIMARY KEY REFERENCES hotel.address,
minStay INT
);
CREATE TABLE activity (
name VARCHAR(50) PRIMARY KEY,
season VARCHAR(50)
);
CREATE TABLE has (
resortAddress REFERENCES resort.address,
activityName REFERENCES activity.name
)
```

3 Functional dependencies and normal forms [30 points]

Consider a table $R(A, B, C, D)$. Recall that a set of attributes X is a superkey if $X^+ = ABCD$; a set X is a key if X is a superkey and no subset of X is a superkey; it is closed if $X^+ = X$.

3.1

Give a set of functional dependencies that satisfies the following conditions: the closed sets are AB, CD, and the keys are AD and BC.

$\{A \to B, B \to A, C \to D, D \to C\}$

Note: The question is a bit ambiguous. You cannot find a set of functional dependencies such that AB and CD are the ONLY closed sets and AD and BC are the ONLY keys. In the solution above, AB and CD are the only closed sets; however, AC and BD are also keys. Therefore, I also accept solutions in which there are other closed sets and keys (i.e., as long as AB and CD are closed sets and AC and BD are keys, you are in a good shape).

3.2

For each of the statements below indicate if they are true of false. You need to answer only "true" or "false" and do not need to justify your answer. X and Y denote sets of attributes.

(a) If AB is a key then ABC cannot be closed. **TRUE** ($ABC^+ = ABCD$ in this case).

(b) If AB is closed then ABC cannot be a key. **FALSE** (Consider the case where there is only one dependency: $ABC \to ABCD$).

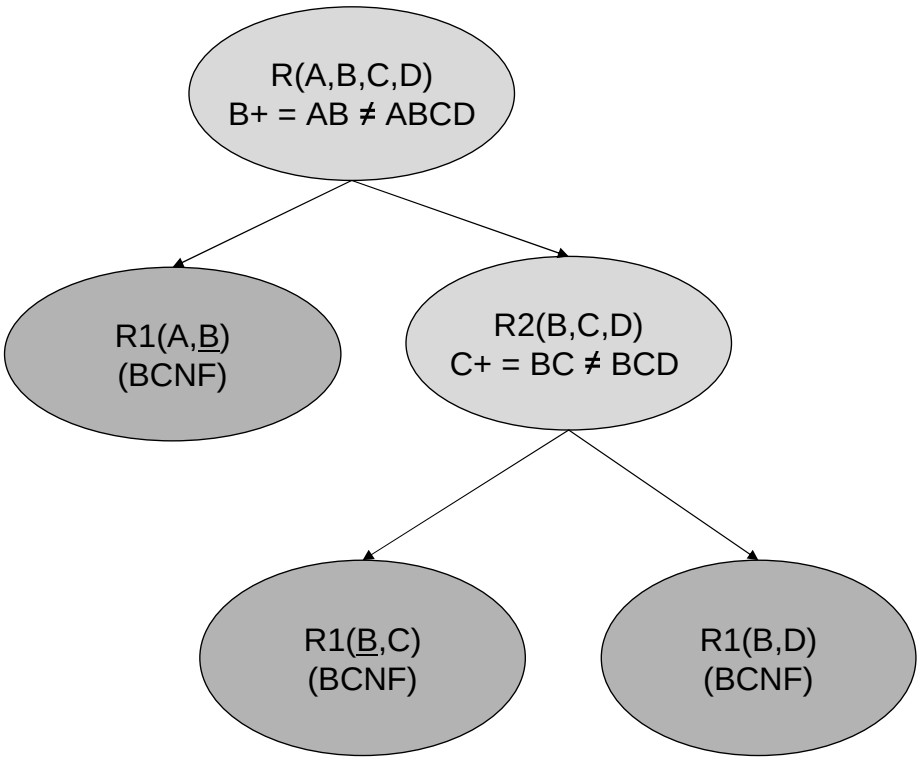
(c) If X, Y are closed then $X \cup Y$ is closed.**FALSE** (Consider the case where ther is only one dependency $AB \to C$ and $X = \{A\}, Y = \{B\}$).

(d) If X, Y are closed then $X \cap Y$ is closed. **TRUE** (Let $Z = X \cap Y$. If Z is not closed then there is an attribute $A \in Z^+ \backslash Z$. Since $A \notin Z$, either $A \notin X$ or $A \notin Y$. Without loss of generality, assume that $A \notin X$. Remember that $A \in Z^+$. Since $Z \subseteq X$, $Z^+ \subseteq X^+$. Therefore, $A \in X^+$. This means $X^+ \neq X$, i.e. X is not closed (contradiction!)).

3.3

Consider the following FDs:

$$\begin{aligned} B &\rightarrow A \\ C &\rightarrow B \end{aligned}$$



End of paper. Good luck!