Key for Exam I
Computer Science 420
Dr. St. John
Lehman College
City University of New York
25 March 2003

1. True or False: `(1 point each)`

   (a) **F** Every relationship in an E/R diagram must have an attribute.

   (b) **F** Subclasses in E/R diagrams are exactly the same as subclasses in Java.

   (c) **T** An entity set can have any number of attributes.

   (d) **F** A relationship connects two or more entity sets.

   (e) **F** To indicate an attribute is a key in an E/R diagram, you circle it.

   (f) **F** Attributes that are keys cannot appear in functional dependencies.

   (g) **T** A weak entity set depends on another entity set for all or part of its key.

   (h) **F** The left side of a multivalued dependency is always a key.

   (i) **T** An SQL query can have at most one relation listed in the `FROM` clause.

   (j) **F** Views cannot be modified.

2. Answer the following in two sentences or less:

   (a) What is the difference between a set and a bag?
   `A set has at most one copy of every element, where a bag (or multiset as it's also known), can have multiple copies.`
   `(5 points total, partial credit (2 points) for relevant, but not completely correct answer.)`

   (b) What is the difference between a functional dependency and multivalued dependency?
   `A functional dependency says that if two tuples of a relation which agree on some particular set of attributes must also agree on some other particular attributes. A multivalued dependency is a weaker condition that says if two tuples of a relation agree on a particular set of attributes then all possible combinations of this particular set (the left hand side of the dependency) and another particular set of attributes (the right hand side).`
   `(5 points total, partial credit (2 points) for relevant, but not completely correct answer).`

3. Write a **complete** Java program "Help, I'm caught inside a database exam!" to the terminal window.

```
public class Question3
{  public static void main(String[] args)
   {  System.out.println("Help, I''m caught inside a database exam!");
   }
}
```

(2 points for having a println() but missing the rest of the program,
-1 points for missing parentheses or minor syntax errors,
-5 points for missing the main() method.)

4. (a) How many different ways are there to represent a relation instance if that instance has 2 attributes and 2 tuples?
The 2 attributes could be in either order and the tuples could be in either order, so, there's $2 \times 2 = 4$ possible ways to represent the instance.
(5 points total. If no explanation, only 1 point partial credit.)

   (b) How many different ways are there to represent a relation instance if that instance has 3 attributes and 2 tuples?
With 3 attributes, any one of them could be listed first, then of the remaining two, either could be next. So, there's $3 \times 2 = 6$ possibilities for the ordering of the attributes. There's 2 ways to order the tuples. So, overall, there's $2 \times 6 = 12$ possible ways to represent the instance.
(5 points total. If no explanation, only 1 point partial credit.)

5. For each of the following, indicate whether the statement is **always true** (for every instance of R), **sometimes true** (that is, you can find some instance of a relation for which the statement holds), or **always false** (there is no instance that will make the statement true).
Given a relation R(A,B,C):

   (a) If $A \rightarrow B$, then $B \rightarrow A$.
   Circle one: ALWAYS TRUE **SOMETIMES TRUE** ALWAYS FALSE
   (3 points total, 1 point partial credit for ALWAYS FALSE)

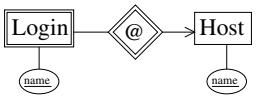
   (b) If $A \rightarrow B$ and $B \rightarrow C$, then $A \rightarrow C$.
   Circle one: **ALWAYS TRUE** SOMETIMES TRUE ALWAYS FALSE
   (3 points total, 1 point partial credit for SOMETIMES TRUE)

   (c) If $A \rightarrow B$ is the only functional dependency, then BC is a key.
   Circle one: ALWAYS TRUE SOMETIMES TRUE **ALWAYS FALSE**
   (3 points total, 1 point partial credit for SOMETIMES TRUE)

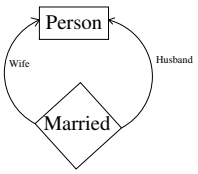
6. For each of the following types of situations, give an example **and** draws its E/R diagram:

(5 points each. No points taken off for missing arrows or keys.
-2 points for wrong example,
-2 points for missing relationship (diamond).)

   (a) a many-one relationship:



(b) an one-one relationship:



7. (a) Draw an E/R diagram for the following situations. Indicate any keys, weak entity sets, or subclasses:
Entity sets *Movies*, *Stars* and *Studios*. A movie has a title, year (in which the movie was made), the length, and the film type (either "color" or "blackAndWhite"). The other two entity sets both have the same attributes: their name and their address. Stars can star in movies. Movies are owned by studios.

`(5 points. See p 26 for diagram.)`

(b) Translate your E/R diagram into a relation schema. Indicate keys for each relation as well as any functional dependencies that hold about each relation.

`(7 points. See Example 3.1 on p 67 for relation schema and Examples 3.16 and 3.17 on p 87-88 for functional dependencies.)`

(c) Write the SQL statement that will create the Stars table above.

`(4 points)`

```
CREATE TABLE Stars(
  name char(30),
  address varchar(50),
};
```

(d) Write the SQL statements that add the following stars to your table that keeps track of stars:

```
Harrison Ford, 123 Maple Street
Carrie Fisher, 456 Broadway
```

`(4 points)`

```
INSERT INTO Stars(name, address)
       VALUES('Harrison Ford', '123 Maple Street');
INSERT INTO Stars(name, address)
       VALUES('Carrie Fisher', '456 Broadway');
```

8. Given the relation schema $R(A, B, C, D)$ with the functional dependencies

$$AB \to C$$
$$C \to D$$
$$D \to A$$

(a) What are the keys of R? `Note that any key must include B, since there's no way to derive B from the functional dependencies that`

yield B. So, we only need to check the closure of subsets that contain B:

$$\begin{array}{lll} A^+ = A & AB^+ = ABCD & ABC^+ = ABCD \\ B^+ = B & BC^+ = ABCD & ABD^+ = ABCD \\ C^+ = ACD & BD^+ = ABCD & BCD^+ = ABCD \\ D^+ = D & & \end{array}$$

This gives superkeys of AB, BC, BD, ABC, ABD, BCD, and ABCD, and keys of AB, BC, and BD.

(4 points total.
-1 point for each missing key.)

(b) Indicate all the Boyce Codd Normal Form violations. Do not forget to consider dependencies that are not in the given set, but follow from them. However, it is not necessary to give violations that have more than one attribute on the right side.

C $\rightarrow$ D, D $\rightarrow$ A, C $\rightarrow$ A,

(3 points total.)

(c) Decompose the relations, as necessary, into a collection of relations that are in **Boyce Codd Normal Form**.

Decomposing around the BCNF violation, C $\rightarrow$ D, gives the smaller relations wtih functional dependencies:

$$\begin{array}{ll} R_1(C, D) & R_2(C, A, B) \\ C \rightarrow D & AB \rightarrow C \\ & C \rightarrow A \end{array}$$

The key for R_1 is C, so R_1 is in BCNF.
The key for R_2 is AB, so C $\rightarrow$ A is a BCNF violation. Decomposing R_2 gives:

$$\begin{array}{ll} R_3(C, A) & R_4(C, B) \\ C \rightarrow A & \end{array}$$

(3 points total.)

9. Given the company database used in laboratory exercises with the relations:

```
companies (co_id serial,
   co_name varchar(64),
   co_postcode varchar(16),
   co_lastchg timestamp );
products (pr_code varchar(6) PRIMARY KEY,
   pr_desc varchar(64));
orders (ord_id serial,
   ord_company int4 REFERENCES companies(co_id),
   ord_product varchar(6) REFERENCES products(pr_code),
   ord_qty int4,
   ord_placed date,
   ord_delivered date,
   ord_paid date);
```

Write SQL queries that:

(a) gives the product codes contained in the database:

```
SELECT pr_code FROM products;
```

(2 points)

(b) gives the product codes and the minimum number order, the average number ordered, and the maximum number ordered:

```
SELECT ord_product, MIN(ord_qty), AVG(ord_qty), MAX(ord_qty)
FROM orders
GROUP BY ord_product;
```

(3 points)

(c) list all orders that not been paid for:

```
SELECT *
FROM orders
WHERE ord_paid = NULL;
```

(2 points)

(d) find each order and its quantity that exceeds the average order quantity for all orders:

```
SELECT *
FROM orders
WHERE ord_qty > SELECT AVG(ord_qty) FROM orders;
```

(3 points)