1. True or False:

(a) T. Every set is a bag.
(b) T. SQL regards relations as bags of tuples, not sets of tuples.
(c) T. Views can be queried as if they were tables.
(d) T. Keys are optional in ODL.
(e) F. There are no subclasses in ODL.
(f) F. In the relational algebra, there is only one way to write any given expression.
(g) F. Once defined, tables cannot be altered in SQL.
(h) T. Every natural join can be written as a theta join.
(i) T. The semistructured data model does not require relation schema for the data stored.
(j) F. In SQL, the expression, (NULL AND TRUE) OR FALSE evaluates to FALSE.

2. Suppose we have a relation \( R(A, B, C) \). Circle the statement that best completes the following:

(a) If \( A \rightarrow B \), and \( A \rightarrow C \),
   i. \( A \) is a key.
   ii. \( B \) is a key.
   iii. \( B \rightarrow C \).
   iv. all of the above.
   v. none of the above.

(b) Assume only that \( B \rightarrow C \). What are all the keys for \( R \)?
   i. \( A \) is a key.
   ii. \( B \) is a key.
   iii. \( AB \) is a key.
   iv. all of the above.
   v. none of the above.

(c) If \( A \rightarrow B \) and \( B \rightarrow C \), then
   i. \( A \) is a superkey.
ii. \(AB\) is a superkey.

iii. \(A \rightarrow C\).

iv. all of the above.

v. none of the above.

(d) If \(A \rightarrow B\) and \(B \rightarrow A\), then

i. \(A\) is a superkey.

ii. \(A \rightarrow B\) is a BCNF violation.

iii. \(A \rightarrow C\).

iv. all of the above.

v. none of the above.

(This is from Prof. Ullman’s Fall 1999 exam.)

3. Answer the questions below based on the following schema:

\[
\text{companies(co\_id, co\_name, co\_postcode, co\_lastchg);}
\]
\[
\text{products(pr\_code, pr\_desc);}
\]
\[
\text{orders(ord\_id, ord\_company, ord\_product, ord\_qty, ord\_placed,}
\]
\[
\text{ord\_delivered, ord\_paid);}
\]
\[
\text{diary(dy\_id, dy\_company, dy\_timestamp, dy\_type, dy\_notes);}
\]

(a) Write a query that returns the company names contained in the database:

\[
\text{SELECT co\_name FROM companies;}
\]

(b) Write a query that returns the product codes with the largest amount ordered for each product.

\[
\text{SELECT ord\_product, MAX(ord\_qty)}
\]
\[
\text{FROM orders}
\]
\[
\text{GROUP BY ord\_product;}
\]

(c) Create a view that contains the name of each company and the total number of orders placed for that company:

\[
\text{CREATE VIEW orderTotals AS}
\]
\[
\text{SELECT co\_name, SUM(ord\_qty)}
\]
\[
\text{FROM companies, orders}
\]
\[
\text{WHERE ord\_company = co\_id}
\]
\[
\text{GROUP BY ord\_company;}
\]

(This is from the first SQL lab and appeared on an exam from last semester.)

4. Consider a relation \(R(A, B, C, D)\) with the following function dependencies:

\[A \rightarrow B, D \rightarrow B, \text{ and } C \rightarrow A,\]

For each of the following decompositions of \(R\), state whether all relations in the decomposition are in Boyce Codd Normal Form.
(a) \( R_1(A, B), R_2(D, B), R_3(C, A) \)  
All in BCNF? Circle one: **YES**  **NO**

(b) \( R_1(A, B), R_2(B, C), R_3(C, D) \)  
All in BCNF? Circle one: **YES**  **NO**

(c) \( R_1(A, C, D), R_2(A, B) \)  
All in BCNF? Circle one: **YES**  **NO**

(d) \( R_1(B, C, D), R_2(A, C) \)  
All in BCNF? Circle one: **YES**  **NO**

For each decomposition above, you only need check if there’s BCNF violations. So, it’s not necessary to do all the closures, but you do need to check if there’s any more dependencies that can be derived from the ones above. The only extra (non-trivial) dependency that follows from above is \( C \to B \).

Note that if a relation has only two attributes, then it’s in BCNF. If you remember this fact, then you can write down the answer to the first two parts quickly. For the third part, \( C \to A \) is a BCNF violation for \( R_1 \). For the fourth part, \( D \to B \) and \( C \to B \) are BCNF for \( R_1 \).

(This is from Prof. Widom’s Spring 2000 exam.)

5. Consider the following relational schema:

\[
\begin{align*}
\text{Name}(ID, \text{name}) & \quad \text{// ID is a key} \\
\text{GPA}(ID, \text{gpa}) & \quad \text{// ID is a key}
\end{align*}
\]

Rewrite each of the following relational algebra expressions in SQL:

(a) \( \sigma_{\text{gpa} > 3.0}(\text{GPA}) \)

\[
\begin{align*}
& \text{SELECT } * \\
& \text{FROM GPA} \\
& \text{WHERE gpa > 3.0;}
\end{align*}
\]

(b) \( \rho_{\text{Student}(ssn, \text{name})}(\text{Name}) \)

\[
\begin{align*}
& \text{CREATE VIEW Student AS} \\
& \text{SELECT ID as ssn, name} \\
& \text{FROM Name;}
\end{align*}
\]

(You could leave off the view and still get full credit.)

(c) \( \text{Name} \bowtie \text{GPA} \)

\[
\begin{align*}
& \text{SELECT name, Name.ID, gpa} \\
& \text{FROM Name, GPA} \\
& \text{WHERE Name.ID = GPA.ID;}
\end{align*}
\]

(d) \( \pi_{\text{gpa}}( \rho_{0}(\text{ID1, gpa}) \bowtie \text{GPA} ) \)
SELECT gpa
FROM GPA, GPA g1
WHERE GPA.gpa = g1.gpa;

6. (a) A card can be represented as a struct with fields rank (2, 3, ..., 10, Jack, Queen, King, and Ace) and suit (Clubs, Diamonds, Hearts, and Spades). Give a suitable definition in ODL of a structured type Card. This definition should be independent of any class declarations but available to them all.

(b) A hand is a set of cards. The number of cards may vary. Give a declaration in ODL of a class Hand whose objects are hands. That is, this class declaration has an attribute theHand, whose type is a hand.

(This is a starred exercise from the book (see p 165), and the answer to it is written on the book’s webpage.)

7. Assume you have a database called exam2db which contains a table called grades. Write a Java program that prints out the first row in the grades table to the screen:

```java
import java.sql.*;
public class TestDB
{
    public static void main(String[] args)
    {
        //Implicitly load the driver:
        try {
            Class.forName("postgresql.Driver");
        }
        catch(ClassNotFoundException e)
        {
            System.out.println(e);
            System.exit(1);
        }
        try {
            //Open the database:
            Connection db = DriverManager.getConnection("jdbc:postgresql:exam2DB",
                                                            "testPerson", "pass");

            //Read the first line:
            Statement st = db.createStatement();
            ResultSet rs = st.executeQuery("select * from grades");
            //Print the first line:
            System.out.println(rs.getString());
        }
        catch(SQLException e)
        {
        }
    }
    catch(SQLException e)
    {
    }
}
```
(This comes directly from Lab 6.)

8. Suppose $R(A, B)$ and $S(A, C)$ are relations.

(a) Suppose relations $R$ and $S$ have 2 tuples and 1 tuple, respectively.
In the relational algebra, what is the maximum number tuples $R \times S$ could have?
The size of the cross product is the size of $R$ times the size of $S$ or 2.
In the relational algebra, what is the maximum number tuples $R \bowtie S$ could have?
Tuples are only included in the natural join if they have a ‘‘match’’ in each of the relations. So, the maximum number of tuples would be
the larger of the size of $R$ and the size of $S$ or 2.

(b) Suppose relations $R$ and $S$ have 3 tuples and 2 tuple, respectively.
In the relational algebra, what is the maximum number tuples $R \times S$ could have?
The size of the cross product is the size of $R$ times the size of $S$ or $3 \times 2 = 6$.
In the relational algebra, what is the maximum number tuples $R \bowtie S$ could have?
Tuples are only included in the natural join if they have a ‘‘match’’ in each of the relations. So, the maximum number of tuples would be
the larger of the size of $R$ and the size of $S$ or 3.

(c) Suppose relations $R$ and $S$ have $m$ tuples and $n$ tuple, respectively.
In the relational algebra, what is the maximum number tuples $R \times S$ could have?
The size of the cross product is the size of $R$ times the size of $S$ or $m \times n = mn$.
In the relational algebra, what is the maximum number tuples $R \bowtie S$ could have?
Tuples are only included in the natural join if they have a ‘‘match’’ in each of the relations. So, the maximum number of tuples would be
the larger of the size of $R$ and the size of $S$ or $\max(m, n)$.

(A version of this question appeared on the exams from last semester.)

9. Assume you are given the XML DTD below:

```xml
<!DOCTYPE Bars [  
  <!ELEMENT BARS (BAR*)>  
  <!ELEMENT BAR (NAME, BEER+)>  
  <!ELEMENT NAME (#PCDATA)>  
  <!ELEMENT BEER (NAME, PRICE)>  
  <!ELEMENT PRICE (#PCDATA)> ]>
```
Circle the answer for the following questions:

(a) Using a DTD means you’re in which mode of XML:
   i. Well-formed XML
   ii. Valid XML

(b) To use a DTD, the first line of your file should be:
   i. Answer:
      <? XML VERSION = "1.0" STANDALONE = "no"?>
   ii. <? XML VERSION = "1.0" STANDALONE = "yes"?>

(c) A root tag surrounds the entire balance of the document. For the DTD above, the root tag is:
   i. <BAR>
   ii. Answer:
      <BARS>
   iii. <NAME>
   iv. <PRICE>
   v. <ROOT>

(d) A file using the DTD above may contain data for
   i. zero or more BARS
   ii. one or more BARS
   iii. zero or one BARS

(e) Which of the following does not conform to the DTD above:
   i. <BARS></BARS>
   ii. <BARS><BAR><NAME>Joe’s Bar</NAME>
      <BEER><NAME>Bud</NAME>
      <PRICE>2.50</PRICE></BEER>
      <BEER><NAME>Miller</NAME>
      <PRICE>3.50</PRICE></BEER> </BAR></BARS>
   iii. Answer:
      <BARS><BAR><NAME>Joe’s Bar</NAME> </BAR>
      <BAR><NAME>Jose’s Bar</NAME>
      <BEER><NAME>Bud Lite</NAME>
      <PRICE>3.50</PRICE></BEER>
      <BEER><NAME>Pete’s Ale</NAME>
      <PRICE>5.00</PRICE></BEER> </BAR></BARS>

10. Convert the following ODL description of a schema to a relational database schema. Remember that Course objects have an “object identity,” and you may invent an attribute representing this OID, e.g. CourseID.
interface Course {
    attribute integer number;
    attribute string room;
    relationship Dept deptOf inverse Dept::coursesOf;
};

interface LabCourse : Course {
    attribute integer computerAlloc;
};

interface Dept (key name) {
    attribute string name;
    attribute string chair;
    relationship Set<Course> coursesOf
        inverse Dept::deptOf;
};

(This comes from Exam 1 from last semester and as a running example for exercises in the textbook. The answer is provided in the key to that exam.)