MCS-274 Final Exam

Serial #:

This exam is closed-book and mostly closed-notes. You may, however, use a single 8 1/2 by 11 sheet of paper with *hand-written* notes for reference. (Both sides of the sheet are OK.)

Please write your name only on this page. Be sure to look at all problems before deciding which one to do first. Some problems are easier than others, so plan your time accordingly. You have 120 minutes to work.

Write the answer to each problem on the page on which that problem appears. You may also attach additional paper, which should be labeled with your test number and the problem number.

You must sign the honor pledge below and abide by it.

Printed name: _____

On my honor, I pledge that I have not given, received, nor tolerated others' use of unauthorized aid in completing this work.

Signature for above honor pledge: _____

Problem	Page	Possible	Score
1	2	12	
2	3	6	
3	4	12	
4	5	12	
5	6	12	
6	7	12	
7	8	12	
8	10	12	
9	11	10	
Total		100	

1. [**12 Points**] The following tables are used to describe a simplified course's students and their grades on assignments:

```
create table students (
  name varchar(30) not null,
  id int primary key
  );

create table grades (
  student int references students(id),
  assignment int,
  grade char(1) not null,
  primary key(student, assignment)
  );
```

Write each of the following SQL queries. None of your queries should produce duplicate rows.

(a) List the following information for each grade: the assignment number, the student ID number, the student name, and the grade.

(b) List the following information for each student: the student ID number, the student name, and the number of grades the student has on record. This last piece of information should be in a column labeled GRADES and should be present even if it is zero.

(c) List the ID number and name of each student who has a grade of A for at least one assignment.

(d) List the ID number and name of each student who has no grade of A.

(e) List the ID number and the number of A grades for each student who has at least 3 grades of A. The list should be arranged so that the ID numbers are increasing.

2. [**6 Points**] The first two parts of this problem concern the following relation:

A	B	C
a_1	b_1	c_1
a_1	b_2	c_1
a_2	b_1	c_2

(a) Give one example of a nontrivial FD that holds in the relation.

(b) Give one example of an FD that does not hold in the relation.

(c) Mark each of the following as *true* or *false*. You are to consider them in general, not specifically for the relation shown above.

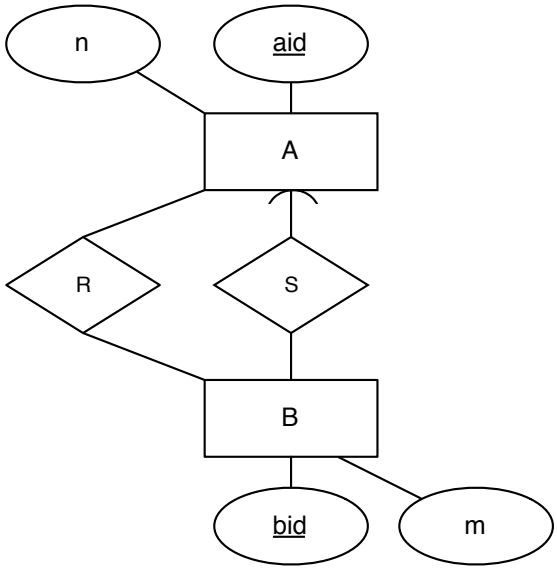
i. Any BCNF decomposition is also a 3NF decomposition.

ii. Any 3NF decomposition is also a BCNF decomposition.

iii. A lossless, FD-preserving BCNF decomposition must exist.

iv. A lossless, FD-preserving 3NF decomposition must exist.

3. [**12 Points**] Consider the following E/R diagram:



(a) Translate this design into an appropriate collection of CREATE TABLE statements, including all relevant constraints. You may assume that all attributes are of type INT.

(b) Make the necessary modifications to the diagram to indicate that B is a weak entity set.

4. [**12 Points**] Next to each of the CREATE VIEW statements, indicate whether the view is updatable, and if not, why not.

```
create table A (
  a1 int not null,
  a2 int not null
  );

create table B (
  b1 int,
  b2 int
  );

create view V1 as
select b1
from B
where b1 < 0;

create view V2 as
select b1, b2
from B
where b1 < 0;

create view V3 as
select a1, a2, b1, b2
from A, B
where b1 < 0
and a2 = b2;
```

5. [**12 Points**] The following problems concern programming in PL/SQL:

   (a) Give an example of what an IN parameter would be useful for in a PL/SQL procedure and what an OUT parameter would be useful for.

   (b) Suppose some PL/SQL code has declared a numeric variable, EARLIEST, and that this variable should contain the minimum value found in the LAUNCHED column of the SHIPS table. Write a reasonably simple PL/SQL statement to achieve this.

   (c) How would the EARLIEST variable be declared to have the same type as the SHIPS table's LAUNCHED column, without needing to know what that type is?

   (d) What would a suitable declaration be for SHIPS_CURSOR to enable the following loop?

```
for s in ships_cursor loop
  -- The body of the loop goes here.
  -- This body is executed for each row of the SHIPS table,
  -- from earliest LAUNCHED year to latest LAUNCHED year, and can make use
  -- of each row's NAME, CLASS, and LAUNCHED.
end loop;
```

   (e) Within the body of the preceding loop, what notation would be used for the value of the CLASS column in the current row?

6. [**12 Points**] Considering the following list of movies:

title	*year*
Star Wars	1977
The Fugitive	1993
Lincoln	2012

(a) Write an XML file corresponding to this data that uses empty elements with attributes. You can omit the initial XML declaration line, which is

```
<?xml version="1.0" ?>
```

(b) Write a second XML file corresponding to the same data but this time using only nonempty elements and no attributes. You can again omit the initial XML declaration.

7. [**12 Points**] Circle the four places in this XML document where it violates the schema on the next page. Write a brief explanation next to each of the four violations.

```
<?xml version = '1.0' encoding="UTF-8"?>
<hotels
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="bogotels.xsd">
  <hotelinfo NAME="ACME Luxury Acapulco">
    <STARS>4</STARS>
    <BEACH>Y</BEACH>
    <DESCRIPTION>Acapulco beach-front luxury with 4-star amenities</DESCRIPTION>
  </hotelinfo>
  <hotelinfo NAME="ACME Luxury Tofte">
    <STARS>5</STARS>
    <BEACH>Rocky</BEACH>
  </hotelinfo>
  <hotelinfo>
    <DESCRIPTION>What do you expect at this price?</DESCRIPTION>
    <STARS>0</STARS>
    <BEACH>N</BEACH>
  </hotelinfo>
</hotels>
```

```
<?xml version='1.0' encoding='UTF-8'?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
 <xs:simpleType name="starsType">
  <xs:restriction base ="xs:integer">
   <xs:minInclusive value="1"/>
   <xs:maxInclusive value="5"/>
  </xs:restriction>
 </xs:simpleType>
 <xs:simpleType name="ynType">
  <xs:restriction base ="xs:string">
   <xs:enumeration value="Y"/>
   <xs:enumeration value="N"/>
  </xs:restriction>
 </xs:simpleType>
 <xs:element name="hotels">
  <xs:complexType>
    <xs:sequence>
     <xs:element name="hotelinfo" maxOccurs="unbounded">
       <xs:complexType>
        <xs:all>
         <xs:element name="DESCRIPTION" type="xs:string"/>
         <xs:element name="STARS" type="starsType"/>
         <xs:element name="BEACH" type="ynType"/>
        </xs:all>
        <xs:attribute name="NAME" type="xs:string" use="required"/>
      </xs:complexType>
    </xs:element>
   </xs:sequence>
  </xs:complexType>
 </xs:element>
</xs:schema>
```

8. [**12 Points**] Consider the XML document shown here with some omissions:

```
<Ships>
        <Class name = "Kongo" type = "bc" country = "Japan"
                        numGuns = "8" bore = "14" displacement = "32000">
                ...
                <Ship name = "Kirishima" launched = "1915">
                        <Battle outcome = "sunk">Guadalcanal</Battle>
                </Ship>
                <Ship name = "Haruna" launched = "1915" />
        </Class>
        ...
</Ships>
```

Write XQuery expressions for each of the following:

(a) Find the `Ship` elements for those ships that were launched prior to 1919 and produce the sequence of these elements surrounded by the tags `<WWI>` and `</WWI>`.

(b) Find the names (as strings, not name attributes) for the classes that contain ships launched in both 1913 and 1915. List each only once.

(c) Find the `Class` elements that have more than three ships.

(d) Find the `Class` elements where every ship in that class was launched after 1918 and before 1939.

9. [**10 Points**] An XML document begins as follows:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE X [
          <!ELEMENT I EMPTY>
          <!ELEMENT II EMPTY>
          <!ELEMENT III EMPTY>
          <!ELEMENT A EMPTY>
          <!ELEMENT B (I | II | III)>
          <!ATTLIST X
                    c (short | longer) #REQUIRED
                    d CDATA #IMPLIED>
          <!ELEMENT X (A+, B*)>
          ]>
```

(a) Write the shortest possible conclusion to this XML document so that it will be valid with regard to the included DTD. That is, you should use as few characters as possible while keeping the XML valid (not just well formed).

(b) Write a different possible conclusion to this XML document that again makes it valid with regard to the included DTD. Be sure to use at least one new tag that you didn't use in your first answer, rather than just more instances of the same tags.