The following two parts of the question are based on the XML "Vehicles" document below.

```
<Vehicles>
   <Car manf="Hyundai">
      <Model>Azera</Model>
      <HorsePower>240</HorsePower>
   </Car>
   <Car manf="Toyota">
      <Model>Camry</Model>
      <HorsePower>240</HorsePower>
   </Car>
   <Truck manf="Toyota">
      <Model>Tundra</Model>
      <HorsePower>240</HorsePower>
   </Truck>
   <Car manf="Hyundai">
      <Model>Elantra</Model>
      <HorsePower>120</HorsePower>
   </Car>
   <Car manf="Toyota">
      <Model>Prius</Model>
      <HorsePower>120</HorsePower>
   </Car>
</Vehicles>
```

Part A. Which of the following XPath expressions, when applied to the "Vehicles" document, does NOT produce a sequence of exactly three items?

a) /Vehicles/Car[@manf="Toyota"]/HorsePower
b) /Vehicles/*[HorsePower>200]/Model
c) //*[@manf="Toyota"]/@manf
d) None of the above (i.e., they all produce exactly 3 items)

Part B. In a DTD that the "Vehicles" document satisfies, which of the following element declarations would you definitely NOT find? Explain Why?

a) <!ELEMENT Vehicles (Car*, Truck+, Car*)>
b) <!ELEMENT Vehicles (Car+, Truck*, Car)>
c) <!ELEMENT Vehicles ((Car|Truck)*)>
d) <!ELEMENT Vehicles (Car*, Truck*, Car*, Truck*, Car*)>

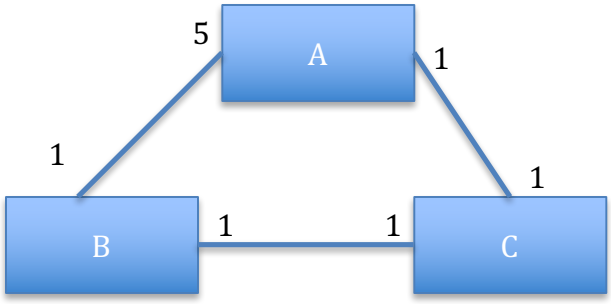
Which of the following statements about E/R models is/are correct?

I. Many-to-many relationships cannot be represented in E/R-diagrams
II. Relationship sets can have attributes of their own.
III. All many-to-one relationships are represented by a relationship between a weak and a non-weak entity set.

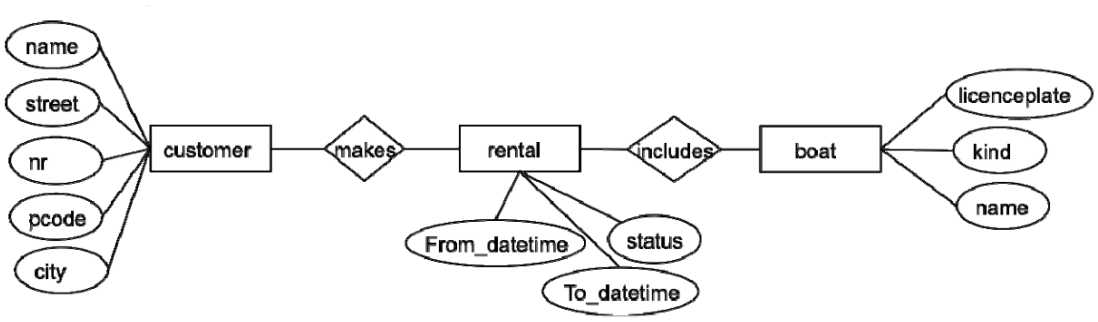
a) II only.
b) III only.
c) II and III only.
d) I and II only.

Explain why?

Design an ER diagram with an entity set E that is over constrained, that is, for which it is not possible for E to have any entities. Use as many entities and relationships as required.



Karl Andersson has a boat rental. As summer season approaches, he thinks of the mess of paper that he has to deal with every year. It would be so much more convenient to have everything organized digitally. His friend comes up with a first draft of an ER model shown in the Figure below.



Every boat has a license plate that identifies the boat, a name, and a type (sailing, motor boat etc.). Furthermore, Karl stores the customer's name and address (pcode is the postal code). Every rental has a number. A boat is rented from a certain date and time to a certain date and time. The status stores the information whether the boat has been returned or not.

(a) Mark the primary key in the ER model (adjust the model if necessary) and give a short explanation, why the primary keys are possible to use.

Customer: nr (customer number is unique)

Makes: the PK would be a combination of PK from Customer and unique information from Rental. As it stands Rental does not have a unique combination for a key. It would need another attribute that would be unique. Lets say rental_nr

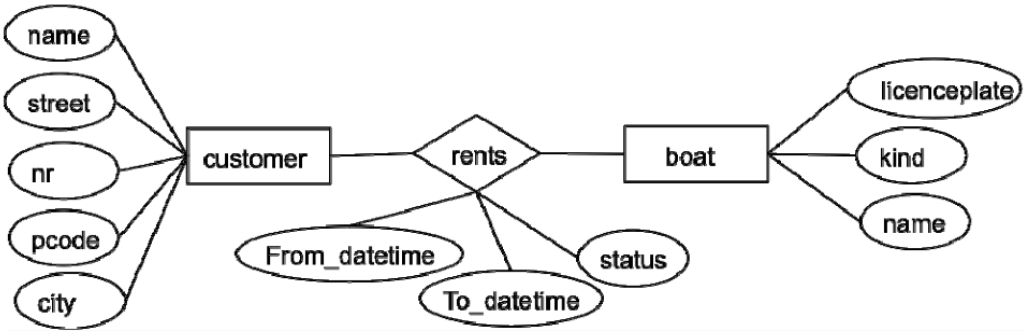
Includes (rental_nr,licenceplate)

Boat: licenceplate

(b) Add cardinality ratios and briefly explain your decision.

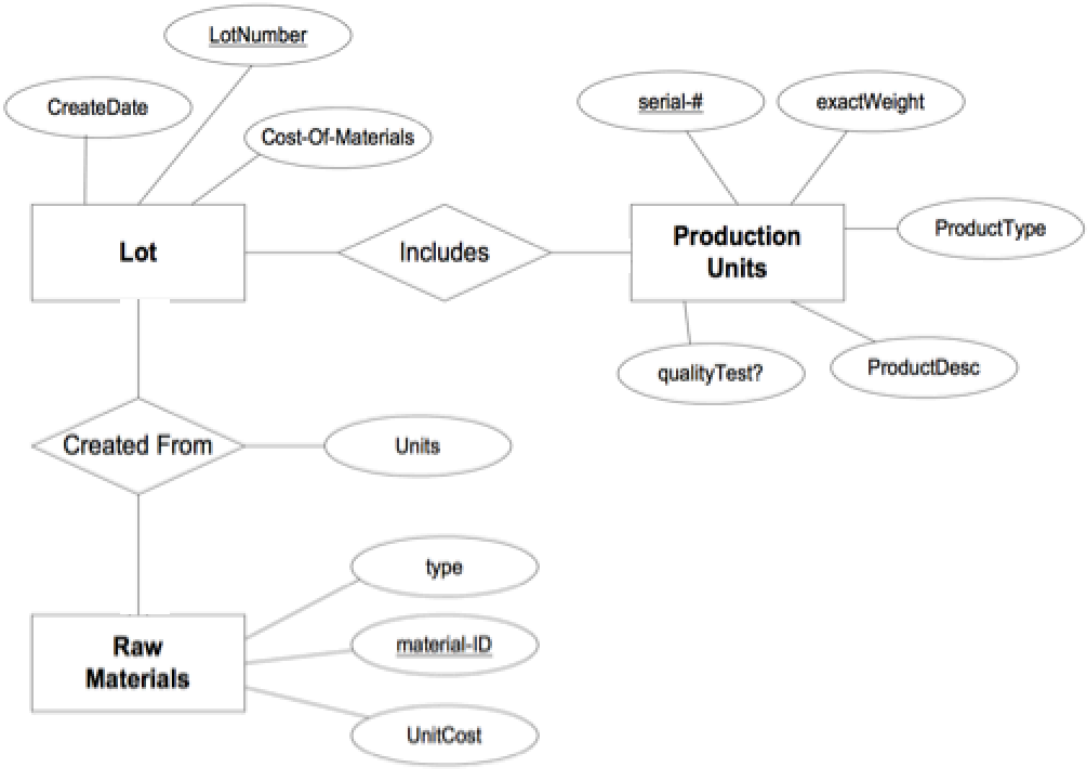
Makes (cutomer,rental): one to many
Includes (rental, boat): one to many

(c) Karl's other friend suggests a much easier model, as shown in the following figure instead of the one shown earlier. What is your reaction and supporting argument?



how would you differentiate between two rentals made on the same dates by the same customer but on separate rental agreemtents?

Production tracking is important in many manufacturing environments (e.g., the pharmaceuticals industry, children's toys, etc.). The following ER diagram captures important information in the tracking of production. Specifically, the ER diagram captures relationships between production lots (or batches), individual production units, and raw materials.



a. Convert the ER diagram into a relational database schema. Be certain to indicate primary keys and referential integrity constraints.

Lot (createDate, LotNumber, Cost-of-Material)
RawMaterial(type,material-ID,UnitCost)
ProductionUnits(serial-#,exactWeight,ProductType,ProductDesc,qualityTest?)
Includes(LotNumber,serial-#)
CreatedFrom(material-ID,LotNumber,Units)

b. identify an attribute in the above ER diagram that might represent a composite attribute, and explain why/how it might represent a composite attribute

ProductDesc

c. Suppose the current ER diagram has the following relationship, "raw materials are used in 0 to many lots." Please explain, in the context of the manufacturing environment, how the meaning changed if the minimal cardinality is changed to "1" (i.e., the relationship becomes "raw materials are used in 1 to many lots.")

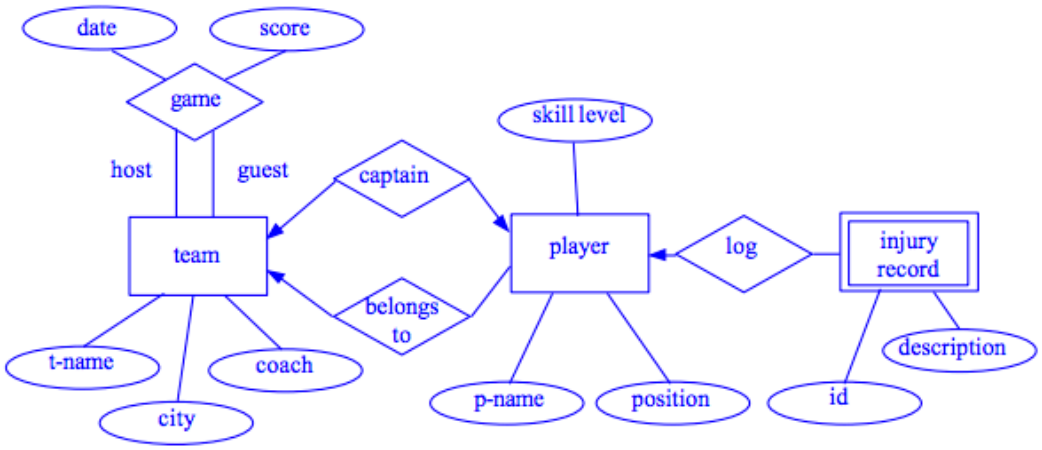
You will not find a raw material that is not used at least in one lot.

Suppose you are given the following requirements for a simple database for the National Hockey League (NHL):

- the NHL has many teams,
- each team has a name, a city, a coach, a captain, and a set of players,
- each player belongs to only one team,
- each player has a name, a position (such as *left wing* or *goalie*), a skill level, and a set of injury records,
- a team captain is also a player,
- a game is played between two teams (referred to as host_team and guest_team) and has a date (such as *May 11th, 1999*) and a score (such as *4 to 2*).

Construct a clean and concise ER diagram for the NHL database. List your assumptions and clearly indicate the cardinality constraints as well as any role indicators in your ER diagram.

Here is one sample solution. Note that other diagrams are possible depending on assumptions.



For the following relation schema:

employee(employee-name, street, city)
works(employee-name, company-name, salary)
company(company-name, city)
manages(employee-name, manager-name)

Give an expression in SQL for each of the following queries:

a) Find the names, street address, and cities of residence for all employees who work for 'First Bank Corporation' and earn more than $10,000.

```
select employee.employee-name, employee.street, employee.city from
employee, works
where employee.employee-name=works.employee-name
        and company-name = 'First Bank Corporation' and salary > 10000
```

b) Find the names of all employees in the database who live in the same cities as the companies for which they work

```
select e.employee-name
from employee e, works w, company c
where e.employee-name = w.employee-name and e.city = c.city
and w.company-name = c.company-name
```

c) Find the names of all employees in the database who live in the same cities and on the same streets as do their managers.

```
select p.employee-name
from employee p, employee r, manages m
where p.employee-name = m.employee-name and m.manager-name =
r.employee-name
and p.street = r.street and p.city = r.city
```

d) Find the names of all employees in the database who do not work for 'First Bank Corporation'. Assume that all people work for exactly one company.

```
select employee-name
from works
where company-name <> 'First Bank Corporation'
```

e) Find the names of all employees in the database who earn more than every employee of 'Small Bank Corporation'. Assume that all people work for at most one company.

```
select employee-name
```

```
from works
where salary > all (select salary
from works
where company-name = 'Small Bank Corporation')
```

f) Find the names of all employees who earn more than the average salary of all employees of their company. Assume that all people work for at most one company.

```
select employee-name
from works t
where salary >(select avg(salary) from works s
where t.company-name = s.company-name)
```

g) Find the name of the company that has the smallest payroll.

```
select company-name
from works
group by company-name
having sum(salary) <= all (select sum(salary)
from works
group by company-name)
```

Consider the following relational database:

employee(e-name, street, city)
works(e-name, c-name, salary)
company(c-name, city)
manages(e-name, m-name)

For each of the following queries, give an expression in relational algebra form.

a) Find the names and cities of residence of all employees who work for the First Bank Corporation.

$$\Pi_{e\text{-}name,\ city}(employee \bowtie (\sigma_{c\text{-}name = \text{'First Bank Corporation'}}(works)))$$

b) Find the names, street address, and cities of all employees who work for First Bank Corporation and earn more than $10,000 per annum. Assume each person works for at most one company.

$$\Pi_{e\text{-}name,\ street,\ city}(\sigma_{(c\text{-}name = \text{'First Bank Corporation'} \wedge salary > 10000)} works \bowtie employee)$$

DO YOU KNOW ANOTHER WAY?

c) Find the names of all employees in this database who do not work for the First Bank Corporation. Assume that all people work for exactly one company.

$$\Pi_{e\text{-}name}(\sigma_{c\text{-}name \neq \text{'First Bank Corporation'}}(works))$$

d) Find the name of all employees who earn more than every employee of Small Bank Corporation. Assume that all people work for at most one company.

$$\Pi_{e\text{-}name}(works) - (\Pi_{works.e\text{-}name} (works \bowtie_{(works.salary \leq works2.salary \wedge works2.c\text{-}name = \text{'Small Bank Corporation'})} \rho_{works2}(works)))$$

e) Assume the companies may be located in several cities. Find all companies located in every city in which Small Bank Corporation is located.

$$\Pi_{c\text{-}name}(company \div (\Pi_{city}(\sigma_{\text{c-name = 'Small Bank Corporation'}}(company))))$$

Just as a reminder:

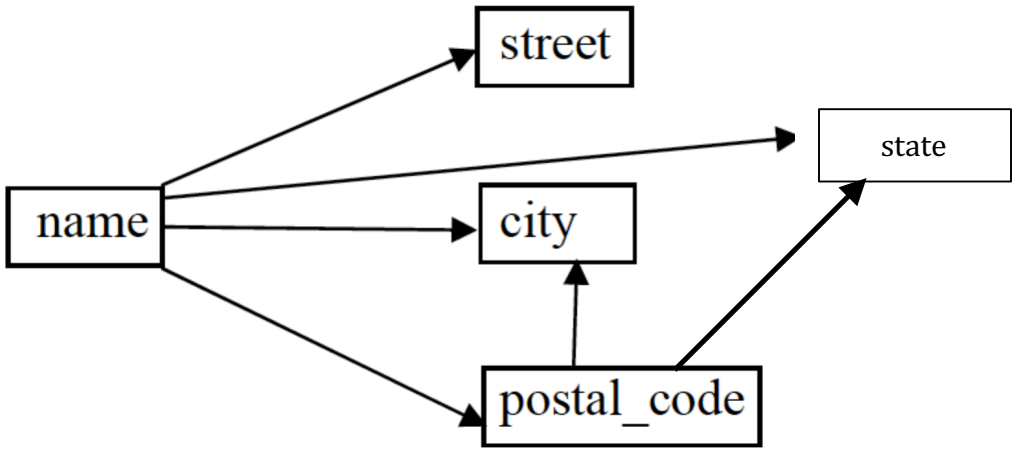
r ÷ s is used when we wish to express queries with "all":

Ex. "Which persons have a loyal customer's card at ALL the clothing boutiques in town X?"
"Which persons have a bank account at ALL the banks in the country?"
"Which students are registered on ALL the courses given by Soini?"
"Which students are registered on ALL the courses that are taught in period 1?"
"Which boys are registered on those courses that are taken by ALL the girls?"
"Which girls are registered on ALL the courses taken by student nr. 40101?"

A relation *NADDR* is defined as follows.
NADDR = (*name*, *street*, *city*, *state*, *postal_code*)
where *name* is unique, and for any given postal code, there is just one city and state

a. Give a set of FDs for this relation.
b. What are the candidate keys?
c. Is *NADDR* in3NF? 2NF? Explain why?
d. If *NADDR* is not in 3NF, normalize it into 3NF relations.

a)



b) The candidate key is name

c) It is in 2NF as there is dependency between non-key attributes as postal-code determines city and state

d) NADDR = (name, street, city, state, postal_code) will be decomposed into
NADDR1 = (name, street, postal_code)
Postal = (postal_code, city, state)

Consider this table.

A	B	C	D	E	F	G
3	small	black	new	4	Lee	inch
6	small	gold	old	4	Lee	foot
3	big	black	new	4	Lee	inch
3	big	black	old	4	Lee	inch
6	big	gold	old	5	Lee	foot

Given this data, determine which of the following statements are apparently true or false.

1. F —> A
2. A —> F
3. [A, B] —> F
4. C —> G
5. [A, B, D] —> [C, E, F, G]

1. False
2. True
3. True
4. True
5. True

6. Consider the following XML document:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet href="mystylesheet.xsl" type="text/xsl"?>
<microloans>
<loaner>
  <name>
    <first>Nandela</first>
    <last>Melson</last>
  </name>
  <address>Freedom Way 1, 23456 Johannesburg, South Africa</address>
  <loan>
    <amount>1000</amount>
    <payout-date>1990-01-01</payout-date>
    <repayment amount="100" date="1991-01-01"/>
    <repayment amount="100" date="1992-01-01"/>
  </loan>
  <loan>
    <amount>500</amount>
    <payout-date>1993-01-01</payout-date>
    <repayment amount="100" date="1991-01-01"/>
  </loan>
</loaner>
<loaner>
  <name>
    <first>Majeev</first>
    <last>Rotwani</last>
  </name>

  <address>Circle Strait 8, 98764 Bumbai, India</address>
</loaner>
</microloans>
```

a) Write an XPath expression that returns all of the name (`name` elements) in `loaners.xml`

//name

b) Write an XPath expression that returns all the names of borrowers, who have (had) at least one loan, which is to say, where there is a `loan` element

//loaner[loan]/name

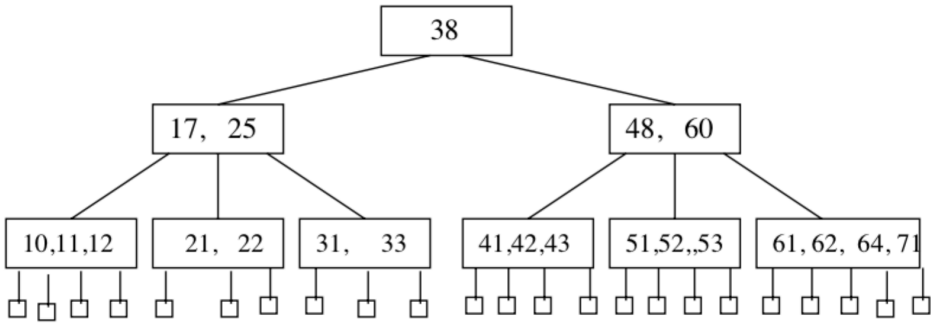
c) Consider the following XSL stylesheet:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

<xsl:template match="microloans">
   <html>
   <body>
     <xsl:apply-templates select="//loan"/>
   </body>
   </html>
</xsl:template>

<xsl:template match="loan">
   <xsl:apply-templates select="../name/last"/>,
   <xsl:apply-templates select="../name/first"/>:
   <xsl:apply-templates select="amount"/><br/>
</xsl:template>
</xsl:stylesheet>
```

State the output of applying this stylesheet on the XML document above.

<html>
<body>
Melson, Nandela: 1000
Melson, Nandela: 500
</body>
</html>

Consider the following B-Tree of order 5.



a) Insert 65 into the B-Tree

Split (61,62,64,65,71) into two node: (61,62) and (65,71) and move 64 up as the parent

b) Delete 21 from the original tree

Borrow from left sibling: move 12 up to the parent and move 17 down to where 21 was deleted from.

c) Delete 33 from the original tree

Once deleted the node will only have 31 in it. The siblings will not have sufficient values to lend values to this node, so a merge will happen. The merged node will contain (21, 22, 25,31).