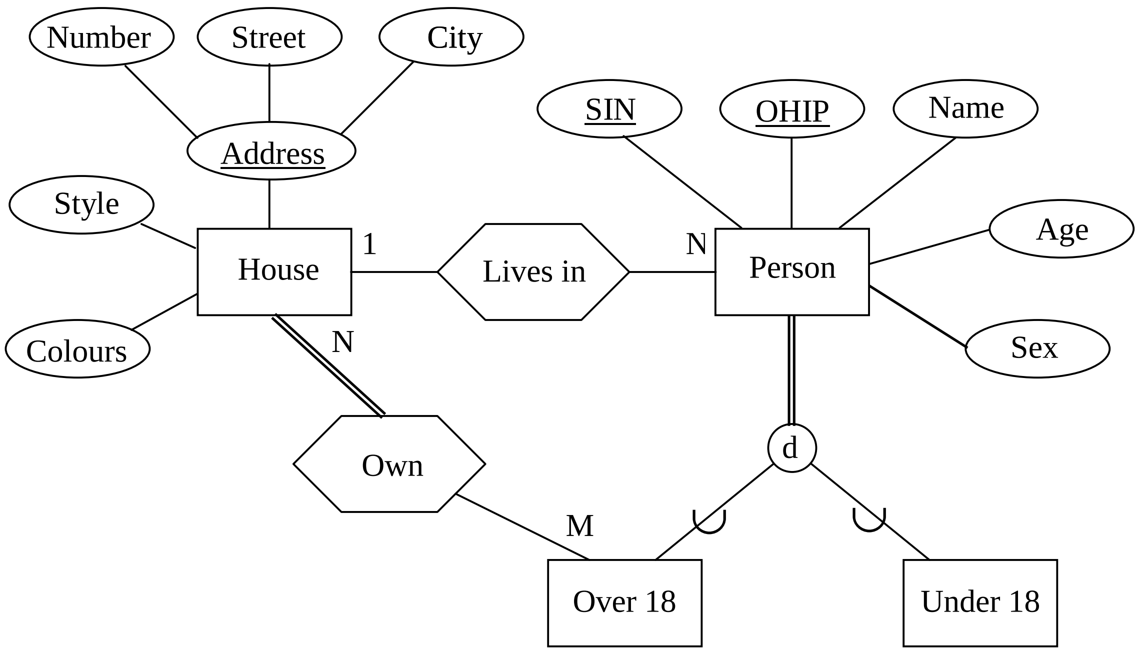
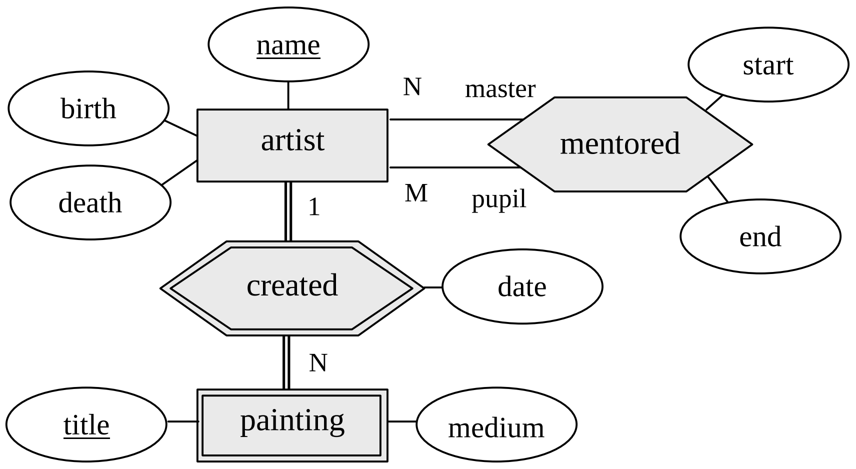
Write all your answers neatly in the answer booklet provided.
Be sure your name and id number are on the cover of the booklet.

1. [12 marks] Use an Entity-Relationship Diagram to depict the following enterprise. Explain any additional assumptions you make and list any aspects you were not able to depict.
   - A house is identified by a three-part address consisting of a number, street, and city. Each house also has a style (e.g., bungalow) and a set of colours.
   - A person is identified by a social insurance number *or* by an OHIP number. Each person also has a name, and age, and a sex.
   - Persons who are at least 18 years old may own zero or more houses, and every house is owned by at least one person.
   - Any person (regardless of age) lives in at most one house as his/her principal residence, and a house can have zero or more persons living there.



Not able to show that Over 18 iff age ≥ 18.

2. [6 marks] Translate the following E-R Diagram into an equivalent relational database scheme, following the procedures given in class.



artist(<u>aname</u>, birth, death)

painting(<u>title</u>, medium, date, aname)

mentored(<u>mastername</u>, <u>pupilname</u>, start, end)

Note that in marking your answers to Questions 3 through 6 below, we will be highly tolerant of minor syntax errors for which your intention is clear.

3. [6 marks] Write an appropriate DDL statement to declare the following relation in SQL:

Car (*LicenceNum*: string, *Province*: string, *Make*: string, *Model*: string, *Year*: integer, *VIN*: string)

where values of *Year* must lie between 1900 and 2002; *Make* is a foreign key into a relation called Manufacturer (having an attribute also named *Make* as *its* primary key); and *VIN* is required to take a unique value for each Car tuple.

```
Create Table Car (
   LicenseNum char(7) not null,
   Province   char(20) not null,
   Make       char(25) foreign key References Manufacturer,
   Model      char(40),
   Year       int,
   VIN        char(15) unique,
   check (Year between 1900 and 2002),
   primary key (licenseNum, Province));
```

4. [4 marks] Translate the following relational calculus query into relational algebra, assuming the existence of relations R(A,B,C) and S(D,E):

$$\{ t \mid t \in R \wedge t[A] = 5 \wedge \exists x\, (x \in S \wedge (t[B] = x[D] \vee t[C] = x[D])) \}$$

$\Pi_{A,B,C} ((\sigma_{A=5}(R) \bowtie_{B=D} S) \cup (\sigma_{A=5}(R) \bowtie_{C=D} S))$

5. [4 marks] Translate the following QBE query into SQL:

R	*X*	*Y*	*Z*
	P.	P._y1	_z2

S	*A*	*B*
	_y1	beta
		_z2

Conditions
_z2 > _y1

```
select X,Y
from R, S
where Y=A and B='beta'
and exists (select *
            from S S2
            where S2.B=Z)
and Z > Y;
```

6. [3 marks each] Consider the following relations as discussed in class:

Emp (eno, ename, title, city)
Proj (pno, pname, budget, city)
Works (eno, pno, resp, dur)
Pay (title, salary)

For each part of this question (considered independently of the other parts), write a single SQL statement that accomplishes the given requirements. (If you believe the English-language query to be ambiguous, rewrite it to clarify the interpretation you have chosen.)

a. For each city, how many projects are located in that city and what is the total budget over all projects in the city?

```
select city, count(pno), sum(budget)
from Proj
group by city;
```

b. For each project, what fraction of the budget is spent (in total) on salaries for the people working on that project? Sort your answer by the value of the budget.

```
select P.pno, pname, sal/budget
from Proj P,
     (select pno, sum(salary) as sal
      from Works, Emp, Pay
      where Works.eno=Emp.eno
      and Emp.title=Pay.title
      group by pno) as Q
where P.pno=Q.pno
order by budget;
```

An alternative solution is the following

```
select P.pno, pname, sum(salary)/budget
from Proj P, Works W, Emp E, Pay
```

```
where P.pno=W.pno
and W.eno=E.eno
and E.title=Pay.title
group by P.pno,budget
order by budget;
```

c. Remove the work assignment of *all* persons to any projects for which more than any 2 persons share the same responsibility.

```
delete from works
where pno in
     (select W.pno
      from Works W
      group by W.pno, W.resp
     having count(*)>2);
```

d. List *all* projects located in Toronto and include for each one the number of persons working on the project.

```
select Proj.pno, pname, count(*)
from Proj left outerjoin Works on (Proj.pno=Works.pno)
where city='Toronto'
group by Proj.pno,pname;
```

e. Insert into the Works relation all tuples that meet the following domain relational calculus expression:

$$\{ <e,p,r,d> \mid \exists\ en, pn, b, c\ (<e,en,r,c> \in \text{Emp} \wedge <p,pn,b,c> \in \text{Proj} \wedge d = 0) \}$$

```
Insert into Works
   (select eno, pno, title, 0
    from Emp, Proj
    where Emp.city=Proj.city);
```

7. [3 marks] With respect to embedded SQL, *briefly* explain the difference between *declaring* a cursor, *opening* a cursor, and *fetching* from a cursor.

Declaring a cursor associates a cursor name with an SQL select statement. Opening the cursor executes the associated statement and initializes the iterator to range over the resulting tuples. Fetching from a cursor returns the next tuple's values from the iterator and moves the iterator on to the next element.

[50 marks in total]