CSE 414 Midterm Exam
Autumn 2018
October 31, 2018

- Please read all instructions (including these) carefully.
- Write your name and UW student number below.
- **This is a <u>closed-book exam</u>. You are allowed one page of note sheets that you can write on both sides.**
- No electronic devices are allowed, including **cell phones** used merely as watches. Silence your cell phones and place them in your bag.
- Solutions will be graded on correctness and ***clarity***. Each problem has a relatively simple and straightforward solution. Partial solutions will be graded for partial credit.
- There are 9 pages in this exam, not including this one.
- Please write your answers in the boxed space provided on the exam, and clearly mark your solutions. You may use the blank pages as scratch paper. **Do not use any additional scratch paper.**
- *Relax. You are here to learn. Good luck!*

Relational algebra operators:

Union ∪ Difference − Selection σ Projection π Join ⋈
Rename ρ Duplicate elimination δ Grouping and aggregation γ Sorting τ

By writing your name below, you certify that you have not received any unpermitted aid for this exam, and that you will not disclose the contents of the exam to anyone in the class who has not taken it.

NAME: _____

STUDENT NUMBER: _____

Problem	Points	Problem	Points
1	10	4	15
2	50		
3	25	**Total**	100

Problem 1: Warm up (10 points total)

Select either True or False for each of the following questions. For each question you get 1 points for answering it correctly, -0.5 point for an incorrect answer, and 0 point for no answer. The minimum you will get for this entire problem is 0.

a) Relational algebra and SQL are declarative languages.

True False

b) There can exist at most one key for a relation.

True False

c) A SQL statement using only SELECT, FROM, and WHERE keywords is always monotone.
(Solutions note: aggregates and subqueries can still make these not monotone.)

True False

d) Datalog sacrifices the properties of physical data independence to implement recursion.

True False

e) Every relation in a database must have a schema.

True False

f) The difference between sets and bags is that sets don't allow repeated values.

True False

g) Foreign keys must reference a primary key in another table.

True False

(Solutions note: both accepted for this class since we didn't cover the UNIQUE constraint)

h) SQL can express more complex queries than relational algebra.
(Solutions note: meaning of complex is unclear here so both answers acceptable. Often DBMS systems implement SQL keywords that can't be expressed with the RA we've learned.)

True False

i) In a table with primary key (A, B), every value of A must be unique.

True False

j) Extended relational algebra operators are for database systems with bag semantics.

True False

Problem 2: SQL (50 points total)

We will work with the following schema for a database of movies in this exam.

ACTOR (pid, fname, lname, gender)
MOVIE (mid, name, year, revenue)
DIRECTORS (did, fname, lname)
CASTS (pid, mid, role)
MOVIE_DIRECTORS (did, mid)

A tuple in the `Casts` relation represents that a person from the `Actor` table with `pid`, starred (was cast) in a `Movie` with `mid`. `Movie_Directors` represents a person from `Directors` with `did`, who directed a `Movie` with `mid`.

ACTOR.pid**, MOVIE.**mid**, DIRECTOR.**did = primary keys of the corresponding tables
CASTS.pid = foreign key to **ACTOR**.pid
MOVIE_DIRECTORS.did = foreign key to **DIRECTORS**.did
CASTS.mid, **MOVIE_DIRECTORS**.mid= foreign keys to **MOVIE**.mid

You can assume none of the tables contain `NULL` values. You may use subqueries for these questions. When writing your queries you can assume that the system is case-insensitive like SQLite.

a) (10 points) Write the SQL create table statements for both `Movie` and `Casts`, making sure to enforce the foreign key references from the `Casts` relation to `Actor` and `Movie`. Use `VARCHAR(n)` for character strings of length `n`.

```
CREATE TABLE Movie (
      mid INT PRIMARY KEY,
      name VARCHAR(150),
      year INT
      revenue FLOAT
);

CREATE TABLE Casts (
      pid INT REFERENCES Actor(id),
      mid INT REFERENCES Movie(id),
      role VARCHAR(50)
);
```

(Solutions note: an actor can play multiple roles in one movie, so (pid, mid, role) is not a primary key)

Schema repeated here for your reference:

ACTOR (pid, fname, lname, gender)

MOVIE (mid, name, year, revenue)
DIRECTORS (did, fname, lname)
CASTS (pid, mid, role)
MOVIE_DIRECTORS (did, mid)

b) (10 points) Return the first and last names of all the actors who were cast in the movie "The Third Man".

```
SELECT x.fname, x.lname
FROM ACTOR x, CASTS c, MOVIE y
WHERE x.pid = c.pid AND
      c.mid = y.mid AND
      y.name = 'The Third Man';
```

c) (10 points) List all directors who directed at least 200 movies, in descending order of the number of movies they directed. Return the directors' first and last names and the number of movies each of them directed.

```
SELECT x.fname, x.lname, COUNT(*) AS c
FROM Directors x, Movie_Directors y
WHERE x.did = y.did
GROUP BY x.did, x.fname, x.lname
HAVING COUNT(*) >= 200
ORDER BY c DESC;
```

Schema repeated here for your reference:

ACTOR (pid, fname, lname, gender)

MOVIE (mid, name, year, revenue)
DIRECTORS (did, fname, lname)
CASTS (pid, mid, role)
MOVIE_DIRECTORS (did, mid)

d) (10 points) For each year, count the number of movies in that year that had only female actors. Remember the meaning of a universal quantifier: a movie without any actors is also a movie with only female actors (since there are no male actors in that movie.) Return the year and the number of movies in that year.

```
SELECT z.year, count(*)
FROM Movie z
WHERE NOT EXISTS (SELECT *
                  FROM Actor x, Casts c
                  WHERE x.pid = c.pid AND
                        c.mid = z.mid AND
                        x.gender != 'F')
GROUP BY z.year;
```

e) (10 points) The highest grossing movie of the year is the one with the largest `revenue` among all movies of that year. Find all movies that have a revenue more than the highest grossing movie of the year 2000. These movies can be from any year. Return the name of each movie and its revenue.

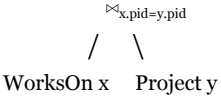
```
SELECT m1.name, m1.revenue
FROM Movie m1
WHERE m1.revenue > ALL (SELECT MAX(m2.revenue)
                        FROM Movie m2
                        WHERE m2.year = 2000)
```

Problem 3: Relational Algebra (25 points total)

We will use the same schema as problem 2, repeated here for your reference:

ACTOR (pid, fname, lname, gender)
MOVIE (mid, name, year, revenue)
DIRECTORS (did, fname, lname)
CASTS (pid, mid, role)
MOVIE_DIRECTORS (did, mid)

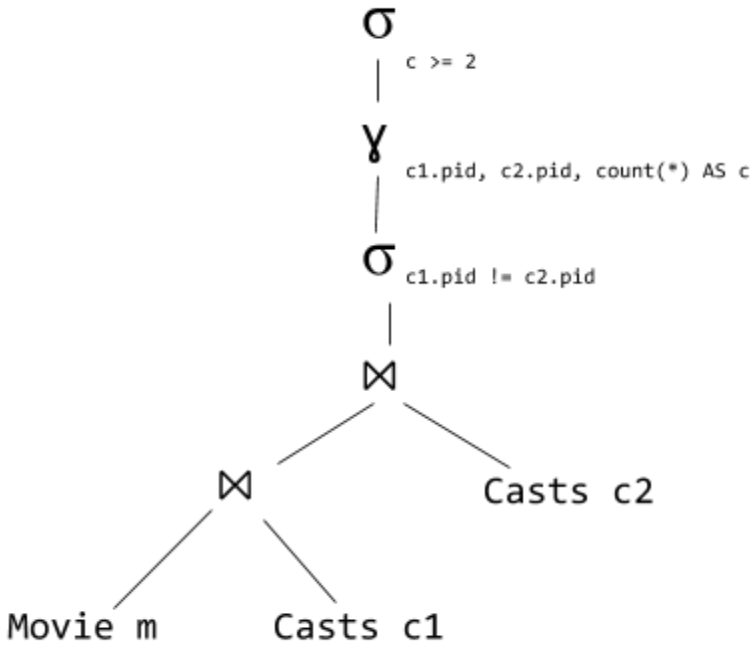
Note: you may use aliases in the query plan to avoid renaming, e.g.



a) (10 points) The following query returns pairs of actors who have starred in at least two movies together:

```
SELECT c1.pid, c2.pid, COUNT(*) AS c
FROM Movie m, Casts c1, Casts c2
WHERE m.mid = c1.mid AND
      m.mid = c2.mid AND
      c1.pid != c2.pid
GROUP BY c1.pid, c2.pid
HAVING COUNT(*) >= 2
```

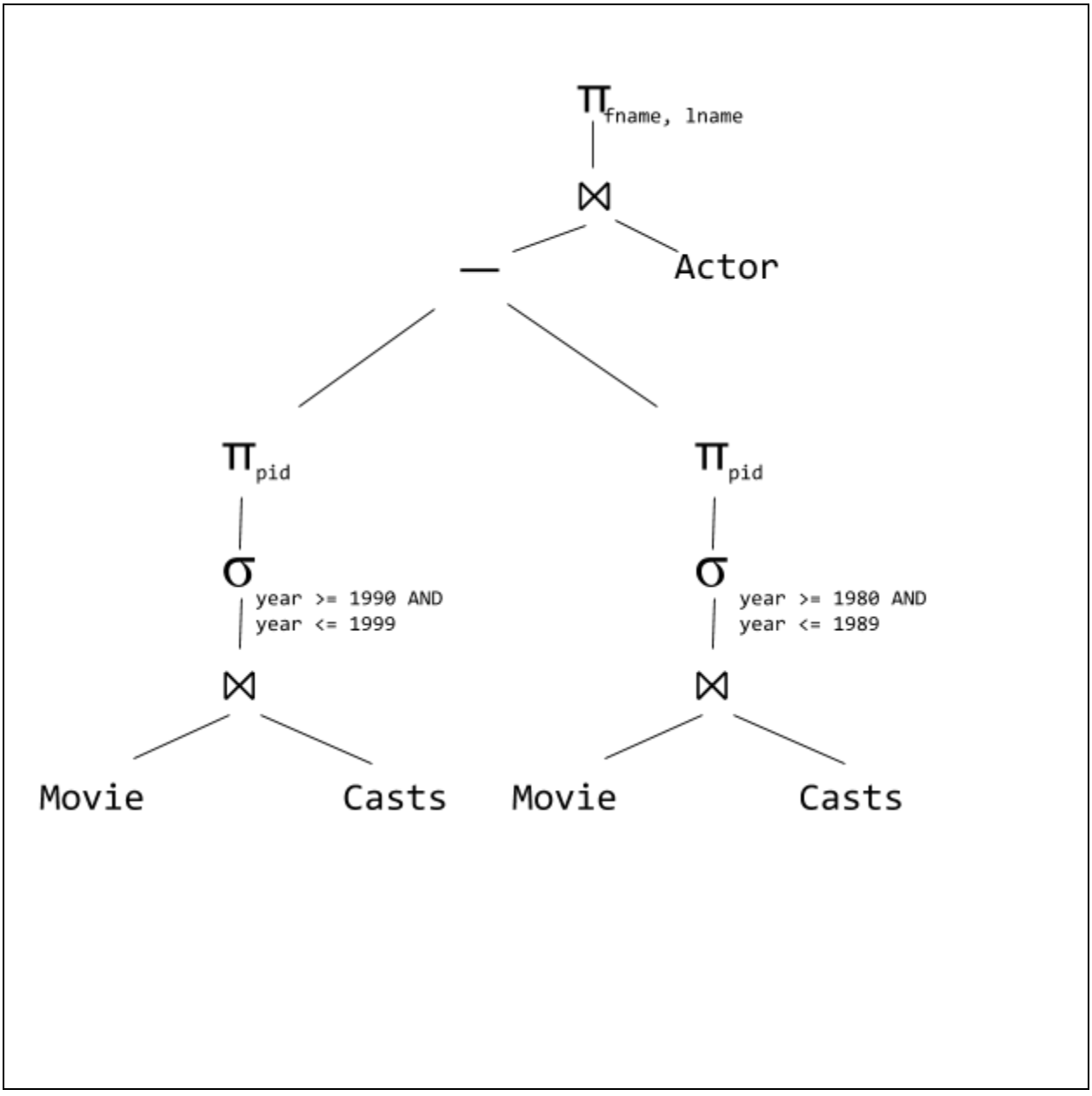
Write a Relational Algebra expression in the form of a logical query plan (you may draw a tree) that is equivalent to the SQL query.



Schema repeated here for your reference:

ACTOR (pid, fname, lname, gender)
MOVIE (mid, name, year, revenue)
DIRECTORS (did, fname, lname)
CASTS (pid, mid, role)
MOVIE_DIRECTORS (did, mid)

b) (15 points) Write a relational algebra query plan (you may draw a tree) that returns the first and last names of all actors who were cast in at least one movie from the 1990s, but were not cast in any movie in the 2000s. Specifically movies in the 1990s have `year` >= 1990 and <= 1999, and movies in the 2000s have year >= 2000 and <= 2009.
Use the set difference operator: —



Problem 4: Datalog (15 points total)

Consider the following database schema. Relation `Item` lists objects with their unique id (`oid`), category, and price. Relation `Gift` has one tuple for every person `pid` that offered a gift to a recipient `rid` . `Gift.oid` references `Item.oid`.

```
Item(oid, category)
Gift(pid, rid, oid)
```

The following datalog query returns the identifier of all people who offered or received a movie as a gift but never offered nor received a book. (The line numbers on the right are not part of the query.)

```
MoviePeople(x) :- Gift(x,_,o), Item(o,'movie')          (1)
MoviePeople(x) :- Gift(_,x,o), Item(o,'movie')          (2)
BookPeople(x) :- Gift(x,_,o), Item(o,'book')            (3)
BookPeople(x) :- Gift(_,x,o), Item(o,'book')            (4)
Answer(x) :- MoviePeople(x), NOT BookPeople(x)          (5)
```

a) (5 points) Given the following input facts, write all the facts in the `Answer` output of this query in the box below.

```
Gift('Bob', 'Joe'  , 3)
Gift('Bob', 'Alice' , 2)
Gift('Bob', 'Mary'  , 3)
Gift('Joe', 'Mary'  , 3)
Gift('Mary', 'Alice'  , 2)
Gift('Alice', 'Bob'  , 1)
Item(1, 'book')
Item(2, 'ring')
Item(3, 'movie')
```

```
Answer('Mary')
Answer('Joe')
```

b) (5 points)

Is this query monotone? Circle one: YES / NO

If the query is not monotonic, write one fact to add to the database (either `Gift` or `Item`) that would demonstrate that the query is not monotone.

(for example):

Adding

```
Gift('Alice', 'Mary', 1)
```

to the input facts would remove `Answer('Mary')` from the output.

c) (5 points)

Is this query safe? Circle one: YES / NO

(every variable in a rule appears in some positive, non-aggregated relational atom)

If the query is unsafe, write the number of the rule from above that makes it unsafe:

_____