1. Bookstore schemas (15wi-Midterm):

Books (<u>bid</u> integer, name string, author string, genre string)
Stores (<u>sid</u> integer, name, city)
Catalog (<u>sid</u>, <u>bid</u>)
-- means that store sid sells book bid
-- sid and bid are foreign keys to Stores and Books, respectively

Sample tuples from the tables:
Books: (1, “Running Dog”, “Don Delillo”, “fiction”)
Stores: (15, “Top Dawgs”, “Seattle”)
Catalog: (1, 15)

No fact is null in the relations and you can assume set semantics for all the questions below. You can use = and ≠ to compare strings. Please only write safe queries.

a) Write a Datalog+negation query to find all stores in Seattle that only sell fiction.

```
NonFiction(bid):- Books(bid,n,a,g), g != “fiction”
NonAns(sid) :- Stores(sid,_,_), Catalog(sid, bid), NonFiction(bid)
Ans(n) :- Stores(sid, n, “Seattle”), !NonAns(sid)
```

b) Write a Datalog+negation query that find all the stores that only sell books by a single author.

```
NonAns (s) :- Catalog(s, b1), Catalog(s, b2), Books(b1, _, a1, _), Books(b2, _, a2, _), a1 ≠ a2
Ans(n) :- Stores(s, n, _), !NonAns(s)

-- We also accepted answers that compute stores that sell books that are written
-- by a single author.
```

2. A datalog rule is safe if and only if every variable appears in a relational atom. (16au-Midterm)

True **<u>False</u>**

3. All queries expressible in datalog (with recursion, but without negation and without aggregates) are monotone. (17au-Midterm)

**<u>True</u>** False

4. Write the following queries using the schema below (16au-Midterm):

```
Product (pid, name, cid)  -- cid is foreign key to Company.cid
Company (cid, cname, city)
Purchase(pid, custId, quantity, price)
-- pid is foreign key to Product.pid, custId is foreign key to Customer.custId
Customer(custId, name, city)
```

a) Write a safe Datalog+negation program that returns the PIDs of the products that have never been sold, along with the names of the companies that made them. Label your answer relation Ans. (13 points)

```
NonAnswers(n, p) :- Product(p, _, c), Company(c, n, _),
                Purchase(p, i, _, _), Customer(i, _, _)
All(n,p) :- Product(p, _, c), Company(c, n, _)
Ans(n,p) :- All(n,p), !NonAnswers(n,p)
```

## 3 Datalog

```
Employee(eid, name, salary)
Project(pid, title, budget)
WorksOn(eid, pid, year)
```

3. (15 points)

   Answer the questions below.

   (a) (5 points) Write a datalog program that returns all employees that worked only on the ``Compiler'' project. Your query should return the eid and name of each such employee.

**Solution:**

```
nonAnswer(eid) :- Employee(eid, -, -),
                  WorksOn(eid, pid, -),
                  Project(eid, t, -),
                  t!= 'Compiler'
Answer(eid, name) :- Employee(eid, name, -), !nonAnswer(eid)
```

2-3 points off for missing negation
1 point off for failing to return employees without any projects
2 points off for unsafe query

```
Employee(eid, name, salary)
Project(pid, title, budget)
WorksOn(eid, pid, year)
```

(b) (10 points) A project p1 influences a project p2, if there exists an employee who worked on p1 during some year, then worked on p2 during some later year. After a major bug was discovered in several projects, the company traced it down to a design flaw in the ``Compiler'' project, and now wants to retain only the projects that were not influenced by ``Compiler''. (Note ``Compiler'' *is* influenced by ``Compiler''.) Write a datalog program to find all projects who were not influenced by the ``Compiler'' project; return their `pid` and `title`.

**Solution:**

```
Infl(x) :- Project(x, 'Compiler', -)
Infl(z) :- Infl(x),
           WorksOn(eid, x, y1),
           WorksOn(eid, z, y2),
           y1 < y2
Answ(pid, title) :- Project(pid, title, -), !Infl(pid)
```

5 points off for missing recursion (almost nobody wrote a recursive query!)
2 points off for unsaafe
1 point off for missing $y1 < y2$
1 point off for missing negation