CSE 532 Assignment #2

Group Members:
1. Jatin Garg (111462753)
2. Rahul Bhansali (111401451)

Questions 1.) (30 points) Let us consider the following relations/predicates:
- Courses(<u>number</u>, quarter, instructor)
- Prereqs(<u>course</u>, <u>prerequisite</u>)
- Students(<u>id</u>, name, address)
- Enrolls(<u>studentID</u>, <u>course</u>, quarter, grade)

Note that courses are represented by their unique numbers in relations Prereqs and Enrolls; students are represented by their ID's. Prereqs table contains only direct prerequisite. For example, if CS101 is a prerequisite of CS200, and CS200 is a prerequisite of CS342, then only the pairs (CS200, CS101) and (CS342, CS200) would appear in Prereqs. We refer to CS101 as an indirect prerequisite of CS342. A prerequisite is considered satisfied only if the students gets an A or A- grade in the course.

1. Write a datalog program that finds all (student, course, quarter) triplets (x,y,q) such that x enrolls in y in quarter q but did not satisfy (i.e., didn't enrol or enrolled but got a B+ or lower grade) one or more of y's direct/indirect prerequisites in a **prior** quarter. In other words, you are trying to create a table of "violations"; a triplet is a violation if either a direct or an indirect prerequisite of a course has not been satisfied in a prior quarter. See #4 below for an example.

Logic:
1. We find all pairs (A,B) such that B is a prerequisite for A (direct or indirect).
2. We find the complement of the Enrolls table i.e. all pairs (S, C) such that a student S is not enrolled for course C.
3. We find all violation tuples, and it can arise from either of these conditions:
    a. Student got a grade lesser than ‘A-’ (i.e. B+ or below) in the course.
    b. Student is enrolled in a prerequisite course for the course after the course’s quarter.
    c. Student didn’t enroll for a prerequisite for the course.

Datalog Program:

1. AllPrereq(c, p) ← Prereqs(c,p)
2. AllPrereq(c, p) ← AllPrereq(c,x) , Prereqs(x,p), c<>p

3. Grade(g) ←Enrolls(s, c, q, g)
4. NotEnrolls(s, p) ← NOT Enrolls(s, p, q, g) , Courses(p, q, instructor) , Students(s, name, address), Grade(g)

5. NotSatisfy(s, c, q) ← AllPrereq(c,p) , Enrolls(s, c, q, g) , Enrolls(s, p, q', g') , g' <= 'B+', c<>p
6. NotSatisfy(s, c, q) ← AllPrereq(c,p) , Enrolls(s, c, q, g) , Enrolls(s, p, q' , g') , q' >=q, c<>p
7. NotSatisfy(s, c, q) ← AllPrereq(c,p) , Enrolls(s, c, q, g) , NotEnrolls(s, p), c<>p

8. Answer(name, c, q) ← Students(s, name, address) , NotSatisfy(s, c, q)

Question 2) (40 points) Consider a database for the Paris Metro and Bus lines, consisting of two relations *Metro(Station, Next)* and *Bus(Station, Next)*. Write the following questions in Datalog:

1. Find pairs of stations *(m,n)*, such that *m* can be reached from *n* by a bus path (using only buses) AND a metro path (using only metros).

   - BusPath(a,b) ← Bus(a,b)
   - BusPath(a,b) ← BusPath(a,c) , Bus(c,b)
   - Metro(a,b) ← Metro(a,b)
   - MetroPath(a,b) ← MetroPath(a,c) , Metro(c,b)
   - Answer(m,n) ← BusPath(n,m), MetroPath(n,m)

2. We say that the metro is useless in a bus path from *m* to *n* if by taking the metro at any intermediate point *k* one can return to *k*, but not reach any other (intermediate) station along the path. Find the pair of stations *(m,n)* such that the metro is useless in all the bus paths connecting *m* and *n*.

   BusPath(a,b) ← Bus(a,b)
   BusPath(a,b) ← BusPath(a,c) , Bus(c,b)

   Metro(a,b) ← Metro(a,b)
   MetroPath(a,b) ← MetroPath(a,c) , Metro(c,b)

   NotAnswer(a,b) ← BusPath(a,c1), BusPath(c1,c2), BusPath(c2,b), BusPath(a,b),
   MetroPath(c1, c2), c1<>c2, c1<>a, c1<>b, c2<>a, c2<>b, a<>b

   Answer(a,b) ← NOT NotAnswer(a,b) , BusPath(a,b)

Question 3.) (30 points) Suppose we have EDB relations: *frequents(Drinker, Bar)* , serves(Bar, Beer), and likes(Drinker, Beer). The relations have obvious meanings. Define the following predicates using safe datalog rules.

1. *veryHappy(D)* if every bar that drinker *D* frequents serves at least one beer he likes. You may assume that every drinker frequents at least one bar.
2. *sad(D)* if drinker *D* frequents all bars that serve no beer he likes.

1.) First we calculated all bars which the user frequents and that bar sells at least 1 beer that the drinker likes. After that a drinker is called sad when he frequents a bar that is not in Satisfy(). After taking negation of sad drinkers, we get our happy drinkers.

- Satisfy(D,Ba) ← serves(Ba, Be) , likes(D,Be)
- Sad(D) ← frequents(D,Ba) , NOT Satisfy(D, Ba)
- Drinker(D) ← *frequents(D,Ba)*
- veryHappy(D) ← NOT Sad(D), Drinker(D)

2.) First, we are finding LikesBar(D,Ba) that sells atleast 1 beer that the user likes. After that we calculated NotLikesBar(D, Ba), which does not sells any beer that the user likes. After that we calculated bars that the user does not frequents. Now if the user does not frequent at least 1 bar that the user likes, he is called as happy. We took the negation of that to find the sad drinkers.

- LikesBar(D,Ba) ← serves(Ba,Be) , likes(D,Be)
- Drinker(D) ← *frequents(D,Ba)*
- Bars(Ba) ← serves*(Ba,Be)*
- NotLikesBar(D, Ba) ← NOT LikesBar(D,Ba), Drinker(D), Bars(Ba)
- NFrequents(D,Ba) ← NOT frequents(D,Ba), Drinker(D), Bars(Ba)
- Happy(D) ← NotLikesBar(D,Ba), NFrequents(D,Ba)
- sad(D) ← Drinker(D), NOT Happy(D)