CSE 341 HW4 REPORT

- ## Part 1

FACTS FORMAT

room(RoomId, Capacity , [Equipments])

course(CourseId, Instructor, [Hours],Capacity,Room,Students)

instructor(Instructor, [Courses],[Needs])

student(StudentId,[Courses],[IsHandicapped])

Conflict Checker Rule

```
conflict(C1,C2):-
    course(C1,_,Hours1,_,Room1,_),
    course(C2,_,Hours2,_,Room2,_),
    (Room1 = Room2) ,nl,
    format('Courses Room is same ~w',[Room1]),
    nl,
    ((subset(Hours1,Hours2));
    (subset(Hours2, Hours1))),
    format('There is a conflict between ~w (Hours : ~w ) and ~w (Hours : ~w ). Course Hourses overlap.',[C1, Hours1,C2,Hours2]).
```

```
| ?- conflict(cse241,cse321).

Courses Room is same z10
There is a conflict between cse241 (Hours : [8,9,10,11] ) and cse321 (Hours : [10,11] ). Course Hourses overlap.
```

```
| ?- conflict(cse341,cse101).

no
```

Enrollment Rule

```
enroll(Student, Course):-
    course(Course,_,_,Cap,Room,Students),
    room(Room,_,Equipment),
    length(Students,Count),
    ((Count +1) =<  Cap),
    student(Student, Lectures, Handicapped),
    member(Handicapped,Equipment),
    member(Course,Lectures),
    format('~w can be enrolled to ~w',[Id, Course]).
```

```
| ?- enroll(ali,cse341).
ali can be enrolled to cse341
```

Since Ali has handicap,

```
student(ali,[cse241,cse341],handicapped).
```

and Cse341 is held in Z23, and capacity is not full,

```
course(cse341,yakup_genc,[11-12],50,z23,[]).
```

and Z23 is suitable for the handicapped students,

```
room(z23,80,[projector,smartboard,handicapped,none]).
```

Ali can be enrolled to CSE341.

Assignment of Room and Classes Rule

```
assignment(Room,Course) :-
    room(Room,Cap,Equipment),
    course(Course,Instructor,Hours,CourseCap,RoomId,_),
    instructor(Id,Courses,Needs),
    member(Course,Courses),
    subset(Needs,Equipment),
    CourseCap =< Cap,
    RoomId = Room,

    (format('~w can be assign to ~w ',[Room, Course])).
```

-Check which room can be assigned to which classes.

```
| ?- assignment(z23,Y).
z23 can be assign to cse102

Y = cse102 ? ;
z23 can be assign to cse341

Y = cse341 ? ;
z23 can be assign to cse331

Y = cse331 ? ;

(1 ms) no
```

```
| ?- assignment(z10,Y).
z10 can be assign to cse241

Y = cse241 ? ;
z10 can be assign to cse321

Y = cse321 ? ;

no
```

-Check which room can be assigned to a given class.

```
| ?- assignment(X,cse341).
z23 can be assign to cse341

X = z23 ? ;

no
```

```
| ?- assignment(X,cse102).
z23 can be assign to cse102

X = z23 ? ;

(1 ms) no
```

-Check which classes a student can be assigned.

```
(1 ms) no
| ?- enroll(ali,Y).
ali can be enrolled to cse341

Y = cse341 ? ;

no
```

```
| ?- enroll(hasmet,Y).
hasmet can be enrolled to cse341

Y = cse341 ? ;
hasmet can be enrolled to cse331

Y = cse331 ? ;
hasmet can be enrolled to cse321

Y = cse321 ? ;
```

Adding Student , Course and Room Rules

```
addStudent(Id,Courses,Handicap) :- assertz(student(Id,Courses,Handicap)).
addCourse(Id,Lecturer,Hours,Capacity,Room) :- addCourse(Id,Lecturer,Hours,Capacity,Room,_).
addRoom(Id,Capacity,Equipment) :- addRoom(Id,Capacity,Equipment).
```

- **PART 2**

FACTS FORMAT

schedule(Source,Dest,Distance).

Ex:

```
schedule(ankara, rize, 5).
schedule(ankara, izmir, 6).
schedule(ankara, istanbul, 1).
schedule(ankara, van, 4).
schedule(ankara, diyarbakir, 8).
```

RULES

Rule for direct route between X and Y Cities. This rule is used to list all the connected cities for a given city.

```
run(X,Y,C,PATH,Visited) :-
    var(Y),
    schedule(X,Y,C).
```

Rules for directed route between X and Y Cities. This rule is used to find paths from source to target Cities. This is arbitrary implementation that I implemented.

```
run(X,Y,C,PATH,Visited) :-
    nonvar(Y),
    \+ member(Y,Visited),
    append(Visited,[Y],Temp),
    schedule(X,Y,C),

    PATH = [X,Y].
```

Rule for undirected route between X and Y Cities.

The deep first search is used for finding the all the path from source to distance.

```
run(X,Y,C,PATH, Visited) :-
    nonvar(Y),
    schedule(X,Z,C1),
    \+ member(Z,Visited),
    append(Visited,[Z],Temp),
    run(Z,Y,C2,P,Temp),
    nl,
    \+ member(Z,Visited),
    C is C1 + C2,
    PATH = [X | P].
```

Connection rule that shows desired query answer.

```
connection(X,Y,C):-
    Visited = [X],
    run(X,Y,C,PATH,Visited),
    (var(PATH) ->
        true;
        write(PATH)
    ).
```

TESTS

-List all the connected cities for a given city.

```
| ?- connection(canakkale,X,C).

C = 6
X = erzincan ? ;

no
```

```
| ?- connection(ankara,X,C).

C = 5
X = rize ? ;

C = 6
X = izmir ? ;

C = 1
X = istanbul ? ;

C = 4
X = van ? ;

C = 8
X = diyarbakir ? ;

no
```

-Direct route between two given cities.

```
| ?- connection(canakkale,erzincan,C).
[canakkale,erzincan]

C = 6 ? ;

no
```

-Find all the path to source to destination.

```
| ?- connection(canakkale,ankara,C).


[canakkale,erzincan,antalya,izmir,ankara]

C = 17 ? ;



[canakkale,erzincan,antalya,izmir,istanbul,ankara]

C = 14 ? ;



[canakkale,erzincan,antalya,izmir,istanbul,rize,ankara]

C = 22 ? ;


[canakkale,erzincan,antalya,diyarbakir,ankara]

C = 21 ? ;

no
```