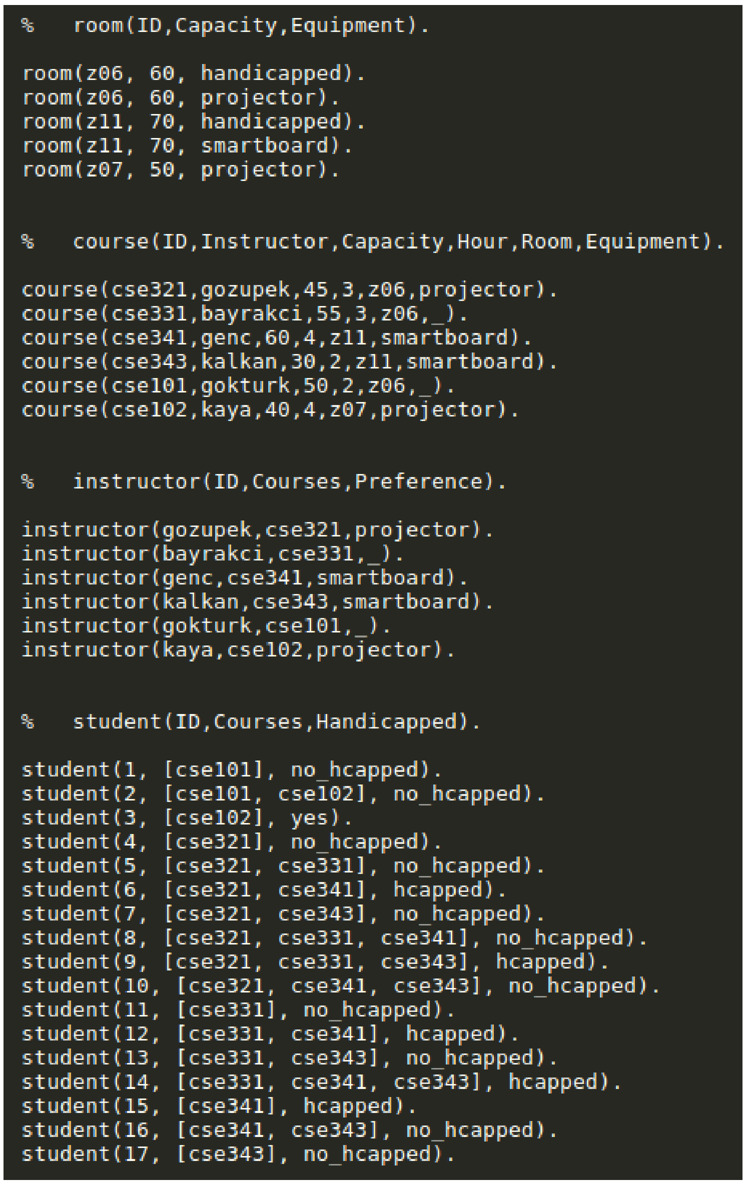
MEHMET ACAR

1801042095

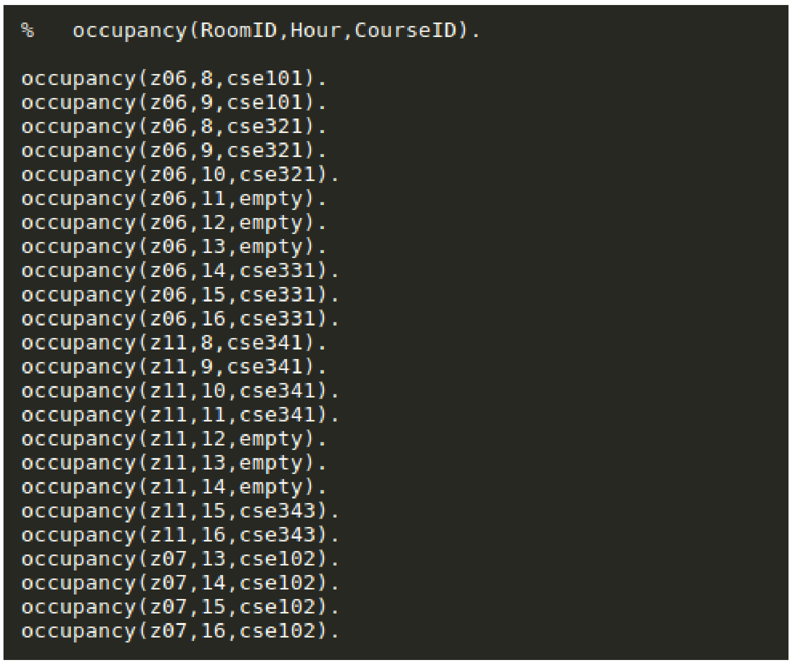
CSE341 PROGRAMMING LANGUAGES HW3 REPORT

Part1

In this part, first of all, I added rooms, courses and students with defining the facts.



```
%   room(ID,Capacity,Equipment).

room(z06, 60, handicapped).
room(z06, 60, projector).
room(z11, 70, handicapped).
room(z11, 70, smartboard).
room(z07, 50, projector).


%   course(ID,Instructor,Capacity,Hour,Room,Equipment).

course(cse321,gozupek,45,3,z06,projector).
course(cse331,bayrakci,55,3,z06,_).
course(cse341,genc,60,4,z11,smartboard).
course(cse343,kalkan,30,2,z11,smartboard).
course(cse101,gokturk,50,2,z06,_).
course(cse102,kaya,40,4,z07,projector).


%   instructor(ID,Courses,Preference).

instructor(gozupek,cse321,projector).
instructor(bayrakci,cse331,_).
instructor(genc,cse341,smartboard).
instructor(kalkan,cse343,smartboard).
instructor(gokturk,cse101,_).
instructor(kaya,cse102,projector).


%   student(ID,Courses,Handicapped).

student(1, [cse101], no_hcapped).
student(2, [cse101, cse102], no_hcapped).
student(3, [cse102], yes).
student(4, [cse321], no_hcapped).
student(5, [cse321, cse331], no_hcapped).
student(6, [cse321, cse341], hcapped).
student(7, [cse321, cse343], no_hcapped).
student(8, [cse321, cse331, cse341], no_hcapped).
student(9, [cse321, cse331, cse343], hcapped).
student(10, [cse321, cse341, cse343], no_hcapped).
student(11, [cse331], no_hcapped).
student(12, [cse331, cse341], hcapped).
student(13, [cse331, cse343], no_hcapped).
student(14, [cse331, cse341, cse343], hcapped).
student(15, [cse341], hcapped).
student(16, [cse341, cse343], no_hcapped).
student(17, [cse343], no_hcapped).
```



```
%   occupancy(RoomID,Hour,CourseID).

occupancy(z06,8,cse101).
occupancy(z06,9,cse101).
occupancy(z06,8,cse321).
occupancy(z06,9,cse321).
occupancy(z06,10,cse321).
occupancy(z06,11,empty).
occupancy(z06,12,empty).
occupancy(z06,13,empty).
occupancy(z06,14,cse331).
occupancy(z06,15,cse331).
occupancy(z06,16,cse331).
occupancy(z11,8,cse341).
occupancy(z11,9,cse341).
occupancy(z11,10,cse341).
occupancy(z11,11,cse341).
occupancy(z11,12,empty).
occupancy(z11,13,empty).
occupancy(z11,14,empty).
occupancy(z11,15,cse343).
occupancy(z11,16,cse343).
occupancy(z07,13,cse102).
occupancy(z07,14,cse102).
occupancy(z07,15,cse102).
occupancy(z07,16,cse102).
```

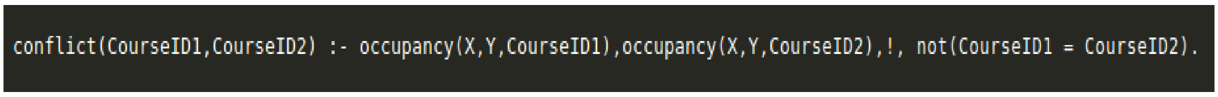
In order to see scheduling conflict between courses, I added CSE101 course's facts.

In order to show that handicapped students can not enroll the course which has not proper equipment in it's room, I added CSE102 course's facts.

conflict(CourseID1,CourseID2)

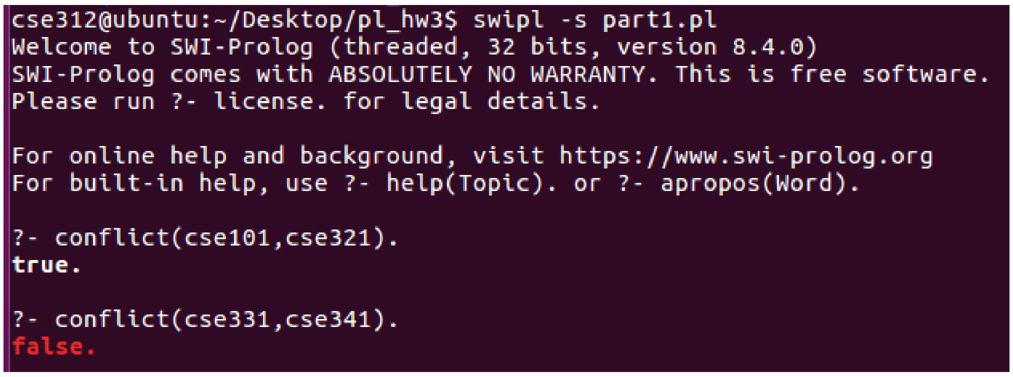
My conflict rule checks that given courses has same class with conflicting hours or not.

In order to do that, it uses occupancy facts. If it finds given courses in occupancy facts with same class and conflicting hours, it returns true. Otherwise, it returns false.



```
conflict(CourseID1,CourseID2) :- occupancy(X,Y,CourseID1),occupancy(X,Y,CourseID2),!, not(CourseID1 = CourseID2).
```

If we query conflict(cse101,cse321).   and   conflict(cse331,cse341).   , results are like below.



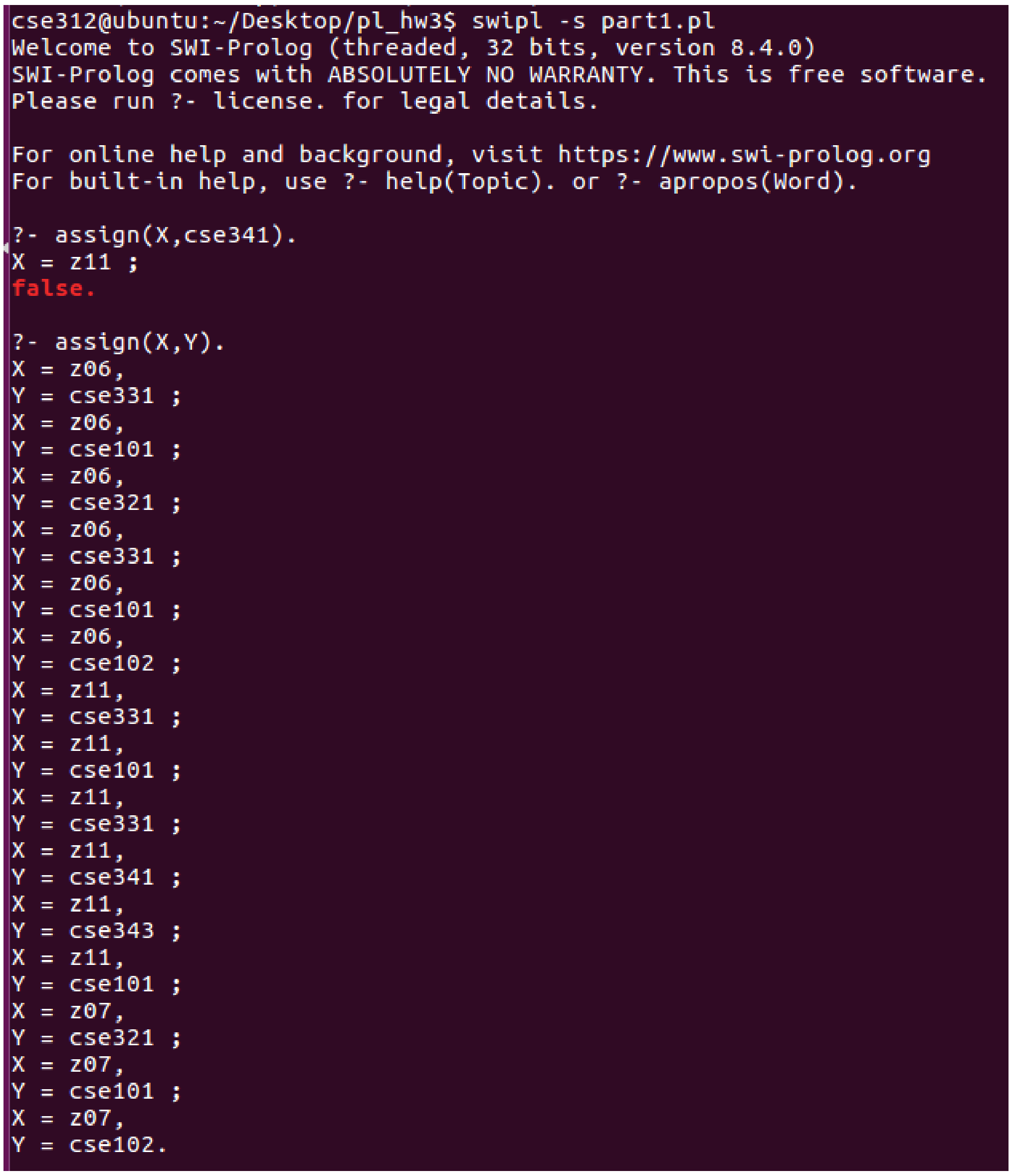
```
cse312@ubuntu:~/Desktop/pl_hw3$ swipl -s part1.pl
Welcome to SWI-Prolog (threaded, 32 bits, version 8.4.0)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- conflict(cse101,cse321).
true.

?- conflict(cse331,cse341).
false.
```

assign(RoomID,CourseID)

My assign rule checks which room can be assigned to given class or which room can be assigned to which class. In order to do that, I used room and course facts. If RoomId's equipment which is found from room facts and CourseID's equipment need which is found from course facts are same and if CourseID's capacity which is found from course facts is equal or less than RoomId's capacity which is found from room facts, assigning operation is successful and it returns RoomID or RoomId-CourseID pair according to user input in query. If CourseID is given and RoomID is unknown when query, it returns only RoomID. If RoomID and CourseID are unknown when query, it returns RoomID-CourseID pair.

```
assign(RoomID,CourseID) :- room(RoomID,X,Equipment), course(CourseID,_,Y,_,_,Equipment), Y=<X.
```

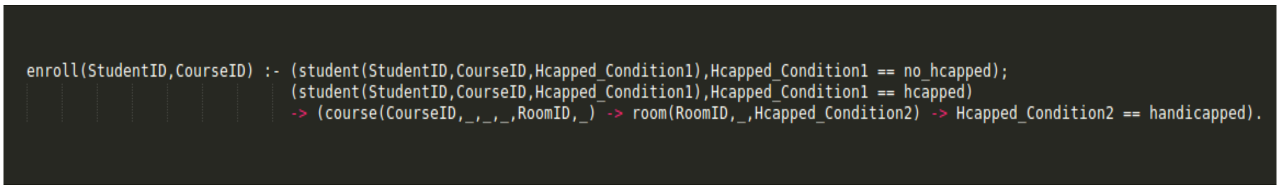
If we query assign(X,cse341). and assign(X,Y).   , results are like below.



```
cse312@ubuntu:~/Desktop/pl_hw3$ swipl -s part1.pl
Welcome to SWI-Prolog (threaded, 32 bits, version 8.4.0)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- assign(X,cse341).
X = z11 ;
false.

?- assign(X,Y).
X = z06,
Y = cse331 ;
X = z06,
Y = cse101 ;
X = z06,
Y = cse321 ;
X = z06,
Y = cse331 ;
X = z06,
Y = cse101 ;
X = z06,
Y = cse102 ;
X = z11,
Y = cse331 ;
X = z11,
Y = cse101 ;
X = z11,
Y = cse331 ;
X = z11,
Y = cse341 ;
X = z11,
Y = cse343 ;
X = z11,
Y = cse101 ;
X = z07,
Y = cse321 ;
X = z07,
Y = cse101 ;
X = z07,
Y = cse102.
```

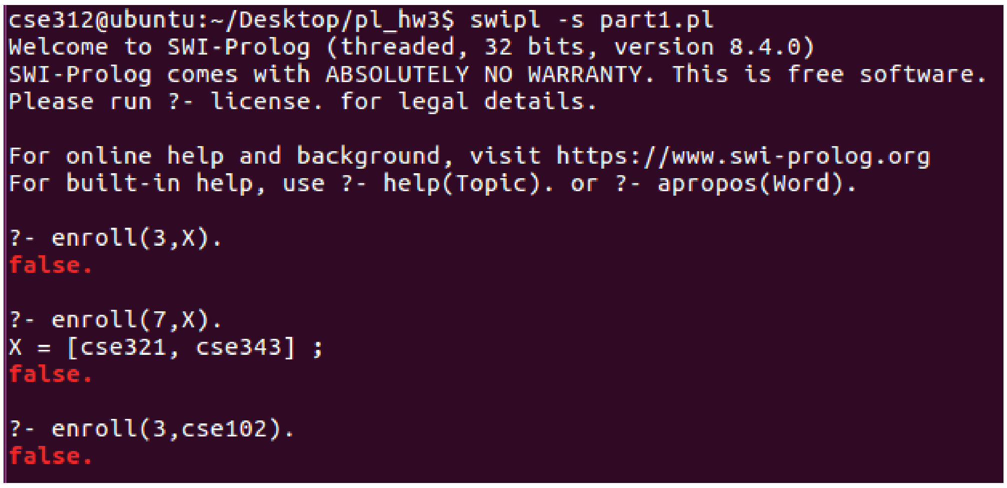
enroll(StudentID,CourseID)

My enroll rule checks whether a student can be enrolled to a given class or which classes a student can be assigned. In order to do that, I used student, course and room facts. If StudentID's condition is hcapped, we get RoomID which is CourseID's room. After that, if that RoomID has equipment for handicapped people, enrolling is successful and it prints true or CourseID according to user input in query. If StudentID is given and CourseID is unknown when query, it returns only CourseID. If StudentID and CourseID are given when query, it returns true or false.



```
enroll(StudentID,CourseID) :- (student(StudentID,CourseID,Hcapped_Condition1),Hcapped_Condition1 == no_hcapped);
                              (student(StudentID,CourseID,Hcapped_Condition1),Hcapped_Condition1 == hcapped)
                              -> (course(CourseID,_,_,_,RoomID,_) -> room(RoomID,_,Hcapped_Condition2) -> Hcapped_Condition2 == handicapped).
```

If we query enroll(3,X). and enroll(7,X). and enroll(3,cse102). , results are like below.

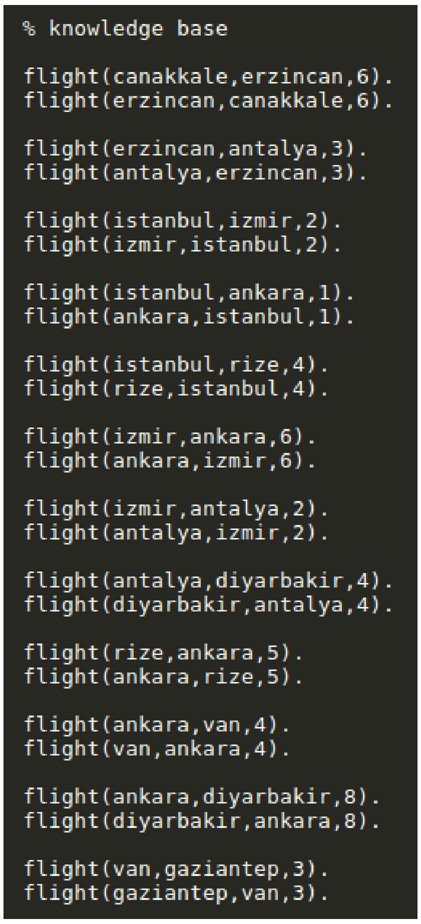


```
cse312@ubuntu:~/Desktop/pl_hw3$ swipl -s part1.pl
Welcome to SWI-Prolog (threaded, 32 bits, version 8.4.0)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- enroll(3,X).
false.

?- enroll(7,X).
X = [cse321, cse343] ;
false.

?- enroll(3,cse102).
false.
```

enroll(3,X). and enroll(3,cse102). returns false. Because, Student which has ID 3 is a handicapped person and in student fact, Student which has ID 3 has a CSE102 course. Cse102 course's room has not proper equipment for handicapped people. So that, it returns false.

PART2

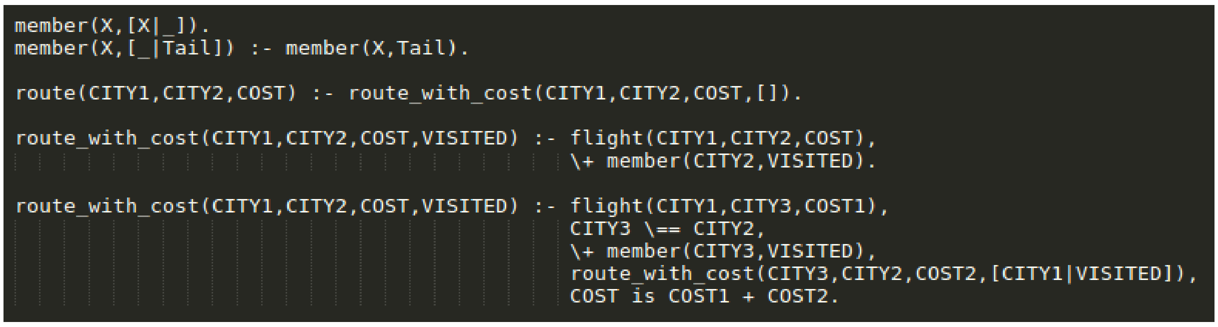
In this part, first of all, I defined the flight facts.



```
% knowledge base

flight(canakkale,erzincan,6).
flight(erzincan,canakkale,6).

flight(erzincan,antalya,3).
flight(antalya,erzincan,3).

flight(istanbul,izmir,2).
flight(izmir,istanbul,2).

flight(istanbul,ankara,1).
flight(ankara,istanbul,1).

flight(istanbul,rize,4).
flight(rize,istanbul,4).

flight(izmir,ankara,6).
flight(ankara,izmir,6).

flight(izmir,antalya,2).
flight(antalya,izmir,2).

flight(antalya,diyarbakir,4).
flight(diyarbakir,antalya,4).

flight(rize,ankara,5).
flight(ankara,rize,5).

flight(ankara,van,4).
flight(van,ankara,4).

flight(ankara,diyarbakir,8).
flight(diyarbakir,ankara,8).

flight(van,gaziantep,3).
flight(gaziantep,van,3).
```

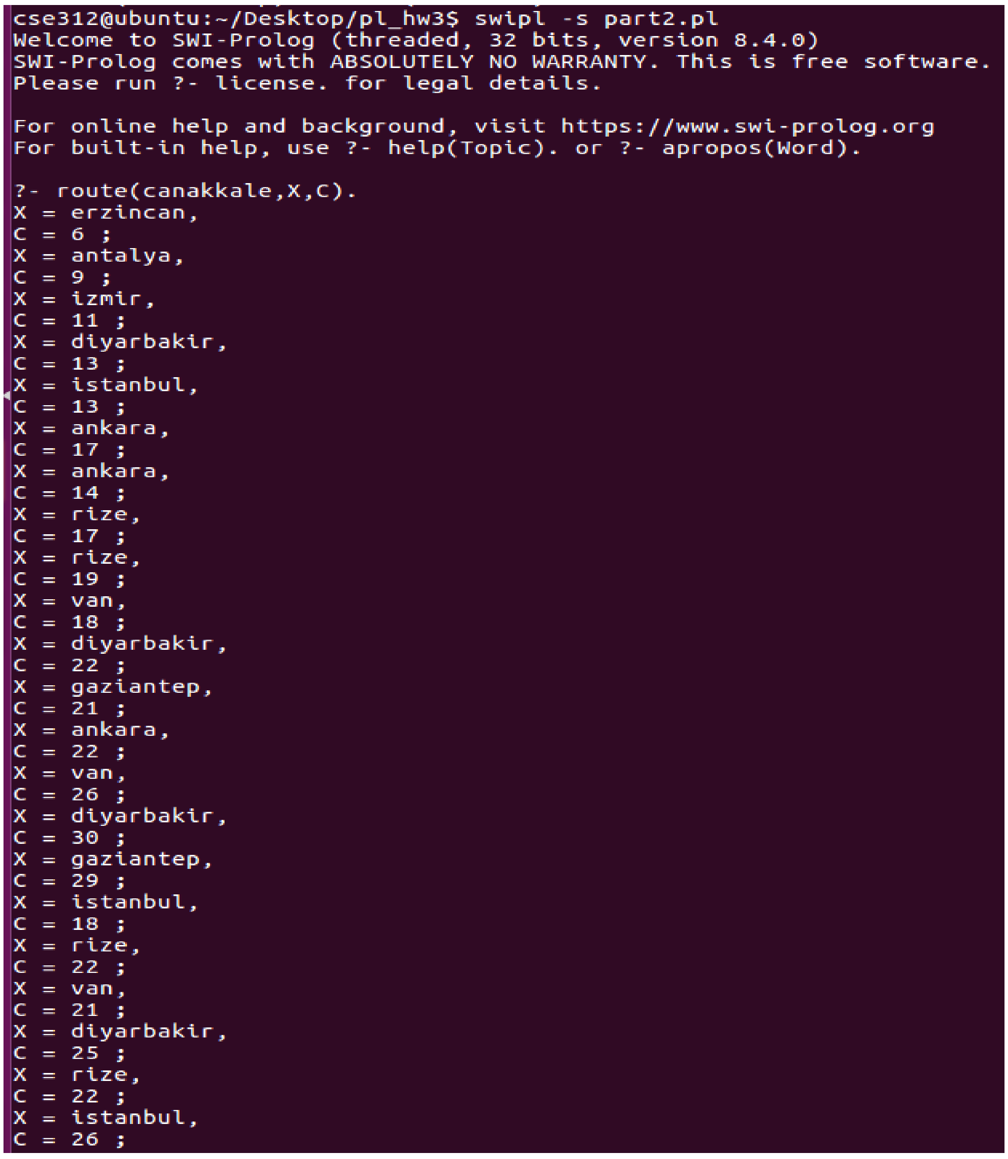
Then, I have a route rule. This rule indicates that we have path on the graph if we have route from one city to another.

In route_with_cost rule, we check that city is visited before or not with using member and we check destination place with CITY3 \== CITY2.

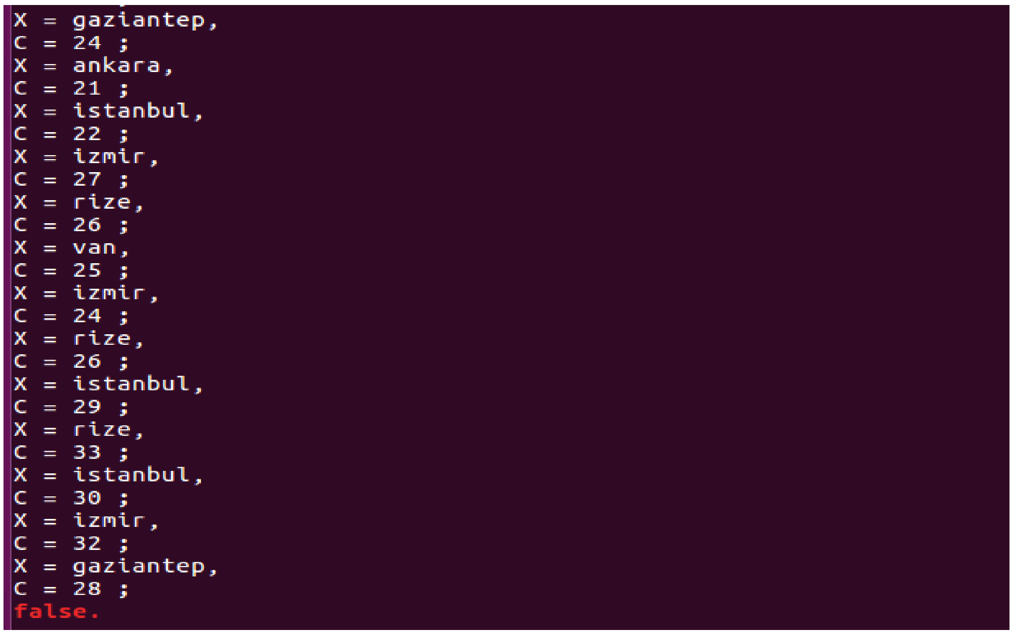


```
member(X,[X|_]).
member(X,[_|Tail]) :- member(X,Tail).

route(CITY1,CITY2,COST) :- route_with_cost(CITY1,CITY2,COST,[]).

route_with_cost(CITY1,CITY2,COST,VISITED) :- flight(CITY1,CITY2,COST),
                                             \+ member(CITY2,VISITED).

route_with_cost(CITY1,CITY2,COST,VISITED) :- flight(CITY1,CITY3,COST1),
                                             CITY3 \== CITY2,
                                             \+ member(CITY3,VISITED),
                                             route_with_cost(CITY3,CITY2,COST2,[CITY1|VISITED]),
                                             COST is COST1 + COST2.
```

If we query route(canakkale,X,C), result is like this.



```
cse312@ubuntu:~/Desktop/pl_hw3$ swipl -s part2.pl
Welcome to SWI-Prolog (threaded, 32 bits, version 8.4.0)
SWI-Prolog comes with ABSOLUTELY NO WARRANTY. This is free software.
Please run ?- license. for legal details.

For online help and background, visit https://www.swi-prolog.org
For built-in help, use ?- help(Topic). or ?- apropos(Word).

?- route(canakkale,X,C).
X = erzincan,
C = 6 ;
X = antalya,
C = 9 ;
X = izmir,
C = 11 ;
X = diyarbakir,
C = 13 ;
X = istanbul,
C = 13 ;
X = ankara,
C = 17 ;
X = ankara,
C = 14 ;
X = rize,
C = 17 ;
X = rize,
C = 19 ;
X = van,
C = 18 ;
X = diyarbakir,
C = 22 ;
X = gaziantep,
C = 21 ;
X = ankara,
C = 22 ;
X = van,
C = 26 ;
X = diyarbakir,
C = 30 ;
X = gaziantep,
C = 29 ;
X = istanbul,
C = 18 ;
X = rize,
C = 22 ;
X = van,
C = 21 ;
X = diyarbakir,
C = 25 ;
X = rize,
C = 22 ;
X = istanbul,
C = 26 ;
```



```
X = gaziantep,
C = 24 ;
X = ankara,
C = 21 ;
X = istanbul,
C = 22 ;
X = izmir,
C = 27 ;
X = rize,
C = 26 ;
X = van,
C = 25 ;
X = izmir,
C = 24 ;
X = rize,
C = 26 ;
X = istanbul,
C = 29 ;
X = rize,
C = 33 ;
X = istanbul,
C = 30 ;
X = izmir,
C = 32 ;
X = gaziantep,
C = 28 ;
false.
```