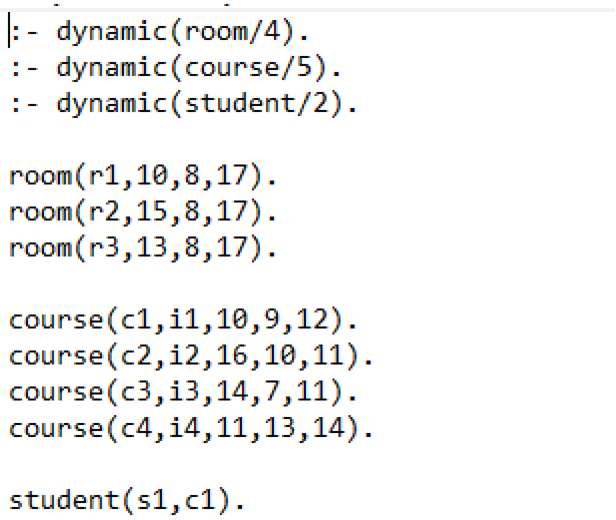
Part 1:

Source Code file: part1.pl

Code:

Knowledge Base:

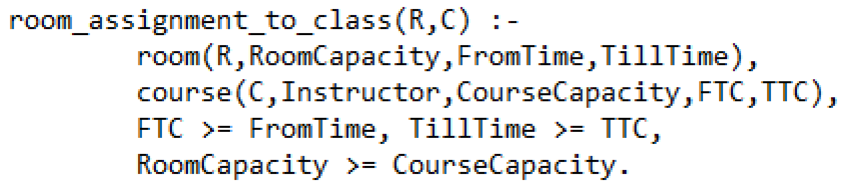


```
:- dynamic(room/4).
:- dynamic(course/5).
:- dynamic(student/2).

room(r1,10,8,17).
room(r2,15,8,17).
room(r3,13,8,17).

course(c1,i1,10,9,12).
course(c2,i2,16,10,11).
course(c3,i3,14,7,11).
course(c4,i4,11,13,14).

student(s1,c1).
```

Room, course and student are determined as dynamic to be able to add new students, rooms or courses to knowledge base.

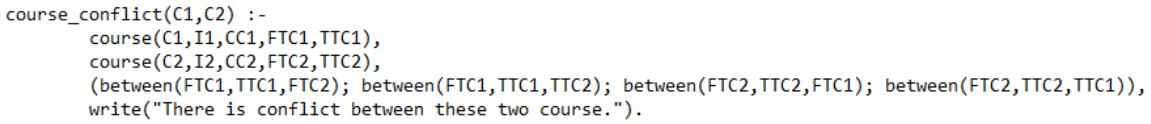
I added some initial facts to try my rules.

Rules:



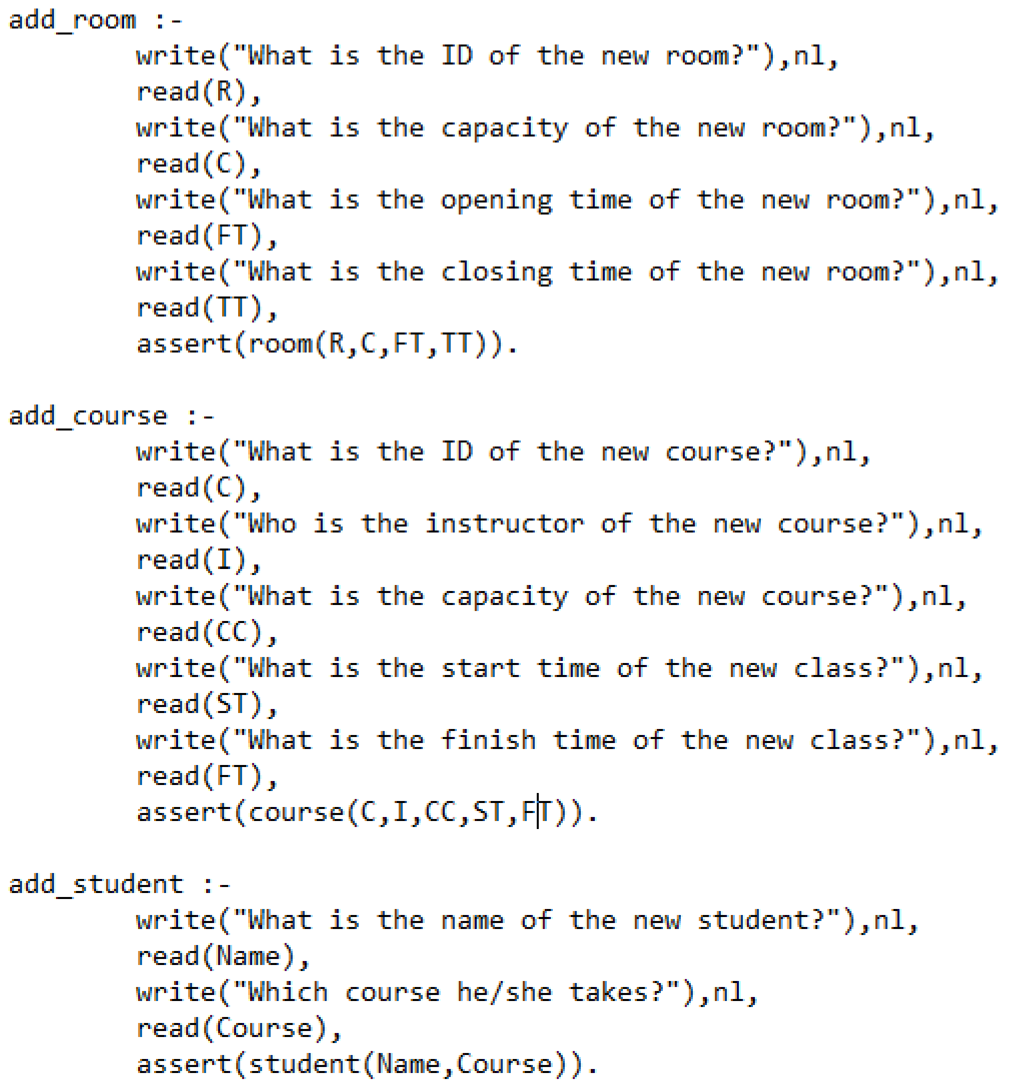
```
room_assignment_to_class(R,C) :-
        room(R,RoomCapacity,FromTime,TillTime),
        course(C,Instructor,CourseCapacity,FTC,TTC),
        FTC >= FromTime, TillTime >= TTC,
        RoomCapacity >= CourseCapacity.
```

Compares rooms and courses capacity and times to determine assignment is valid.



```
course_conflict(C1,C2) :-
        course(C1,I1,CC1,FTC1,TTC1),
        course(C2,I2,CC2,FTC2,TTC2),
        (between(FTC1,TTC1,FTC2); between(FTC1,TTC1,TTC2); between(FTC2,TTC2,FTC1); between(FTC2,TTC2,TTC1)),
        write("There is conflict between these two course.").
```

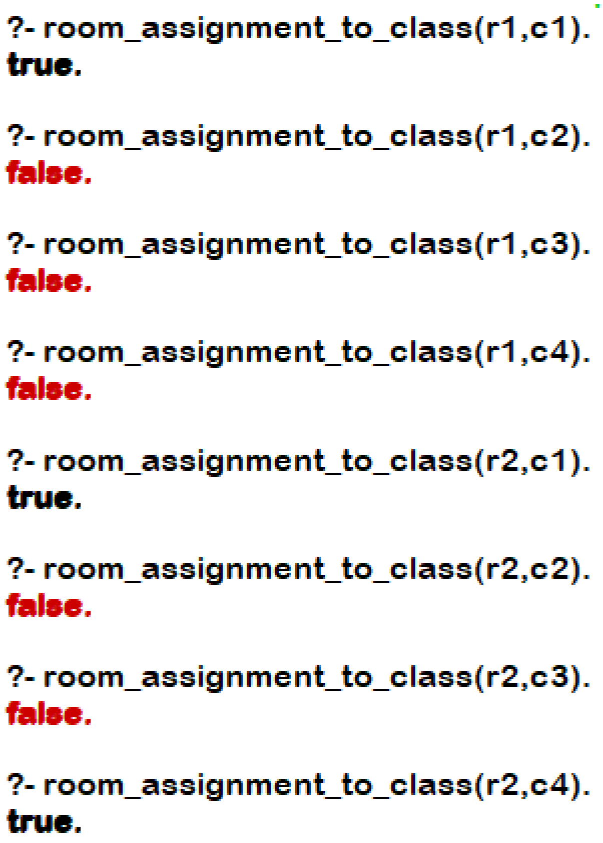
Compares course times of two different courses to control conflicts.



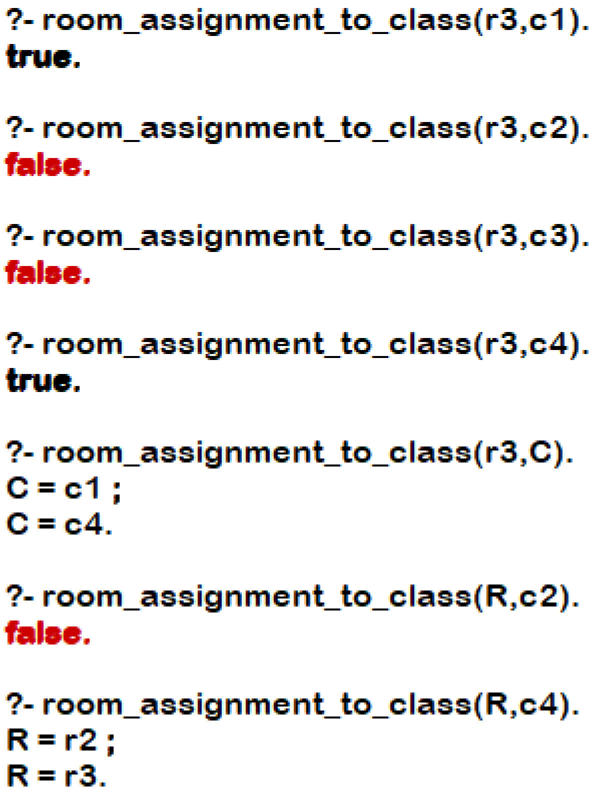
```
add_room :-
        write("What is the ID of the new room?"),nl,
        read(R),
        write("What is the capacity of the new room?"),nl,
        read(C),
        write("What is the opening time of the new room?"),nl,
        read(FT),
        write("What is the closing time of the new room?"),nl,
        read(TT),
        assert(room(R,C,FT,TT)).

add_course :-
        write("What is the ID of the new course?"),nl,
        read(C),
        write("Who is the instructor of the new course?"),nl,
        read(I),
        write("What is the capacity of the new course?"),nl,
        read(CC),
        write("What is the start time of the new class?"),nl,
        read(ST),
        write("What is the finish time of the new class?"),nl,
        read(FT),
        assert(course(C,I,CC,ST,FT)).

add_student :-
        write("What is the name of the new student?"),nl,
        read(Name),
        write("Which course he/she takes?"),nl,
        read(Course),
        assert(student(Name,Course)).
```

Asks questions to user who wants to add new course/student/room to determine course/student/rooms properties. Then adds new course/student/room to knowledge base

Result:

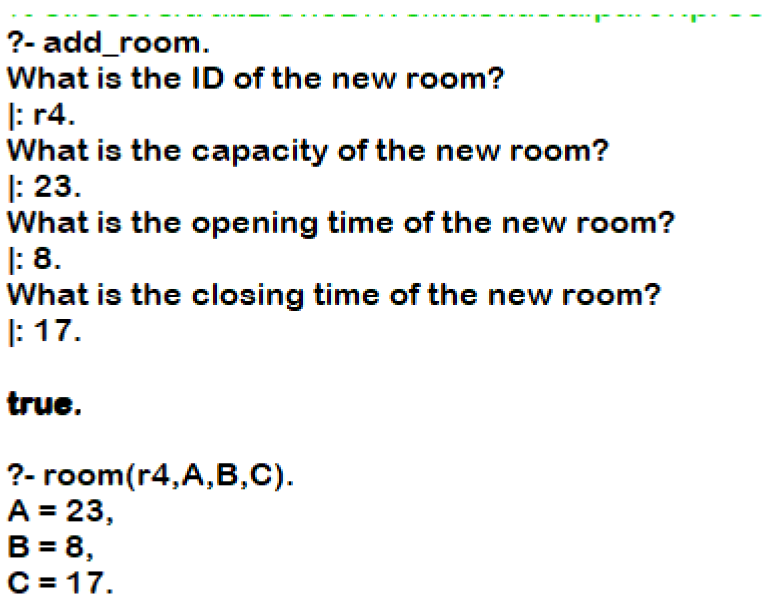


```
?- room_assignment_to_class(r1,c1).
true.

?- room_assignment_to_class(r1,c2).
false.

?- room_assignment_to_class(r1,c3).
false.

?- room_assignment_to_class(r1,c4).
false.

?- room_assignment_to_class(r2,c1).
true.

?- room_assignment_to_class(r2,c2).
false.

?- room_assignment_to_class(r2,c3).
false.

?- room_assignment_to_class(r2,c4).
true.
```

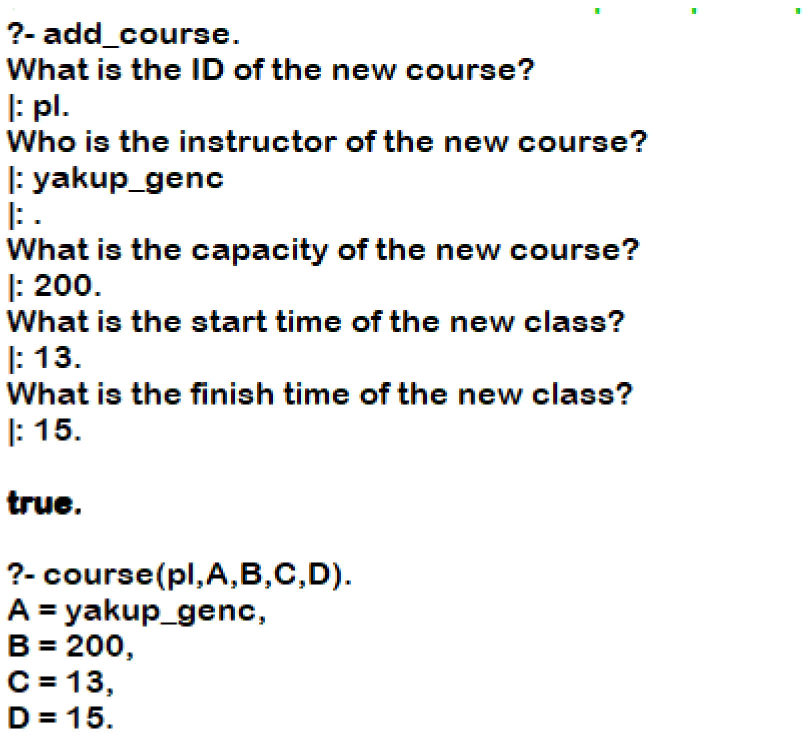


```
?- room_assignment_to_class(r3,c1).
true.

?- room_assignment_to_class(r3,c2).
false.

?- room_assignment_to_class(r3,c3).
false.

?- room_assignment_to_class(r3,c4).
true.

?- room_assignment_to_class(r3,C).
C = c1 ;
C = c4.

?- room_assignment_to_class(R,c2).
false.

?- room_assignment_to_class(R,c4).
R = r2 ;
R = r3.
```

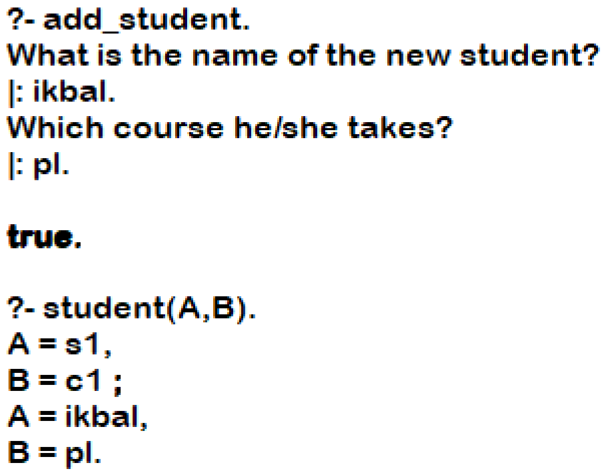
```
?- course_conflict(c1,c2).
There is conflict between these two course.
true .

?- course_conflict(c1,c3).
There is conflict between these two course.
true
Unknown action:  (h for help)
Action?
Unknown action:  (h for help)
Action? .

?- course_conflict(c1,c4).
false.

?- course_conflict(c2,c4).
false.

?- course_conflict(c2,c3).
There is conflict between these two course.
true .


?- course_conflict(c1,C).
There is conflict between these two course.
C = c1 ;
There is conflict between these two course.
C = c1 ;
There is conflict between these two course.
C = c1 ;
There is conflict between these two course.
C = c1 ;
There is conflict between these two course.
C = c2 ;
There is conflict between these two course.
C = c2 ;
There is conflict between these two course.
C = c3 ;
There is conflict between these two course.
C = c3 ;
false.
```



```
?- add_room.
What is the ID of the new room?
|: r4.
What is the capacity of the new room?
|: 23.
What is the opening time of the new room?
|: 8.
What is the closing time of the new room?
|: 17.

true.

?- room(r4,A,B,C).
A = 23,
B = 8,
C = 17.
```



```
?- add_course.
What is the ID of the new course?
|: pl.
Who is the instructor of the new course?
|: yakup_genc
|: .
What is the capacity of the new course?
|: 200.
What is the start time of the new class?
|: 13.
What is the finish time of the new class?
|: 15.

true.

?- course(pl,A,B,C,D).
A = yakup_genc,
B = 200,
C = 13,
D = 15.
```

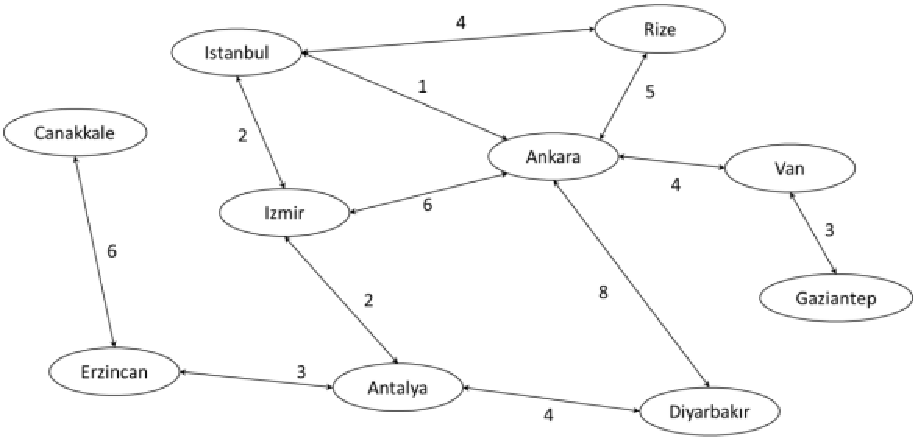


```
?- add_student.
What is the name of the new student?
|: ikbal.
Which course he/she takes?
|: pl.

true.

?- student(A,B).
A = s1,
B = c1 ;
A = ikbal,
B = pl.
```

Part 2:

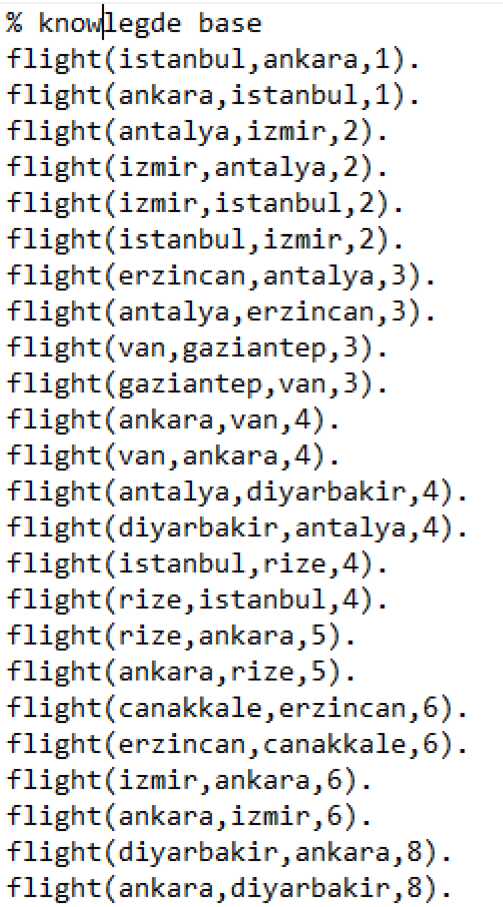
Source Code file: part2.pl

Flight Map:



Code:

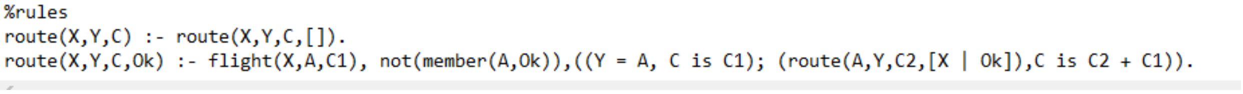
Knowledge Base:



```
% knowlegde base
flight(istanbul,ankara,1).
flight(ankara,istanbul,1).
flight(antalya,izmir,2).
flight(izmir,antalya,2).
flight(izmir,istanbul,2).
flight(istanbul,izmir,2).
flight(erzincan,antalya,3).
flight(antalya,erzincan,3).
flight(van,gaziantep,3).
flight(gaziantep,van,3).
flight(ankara,van,4).
flight(van,ankara,4).
flight(antalya,diyarbakir,4).
flight(diyarbakir,antalya,4).
flight(istanbul,rize,4).
flight(rize,istanbul,4).
flight(rize,ankara,5).
flight(ankara,rize,5).
flight(canakkale,erzincan,6).
flight(erzincan,canakkale,6).
flight(izmir,ankara,6).
flight(ankara,izmir,6).
flight(diyarbakir,ankara,8).
flight(ankara,diyarbakir,8).
```

I added all possible flights according to given flight map to knowledge base of my source code.

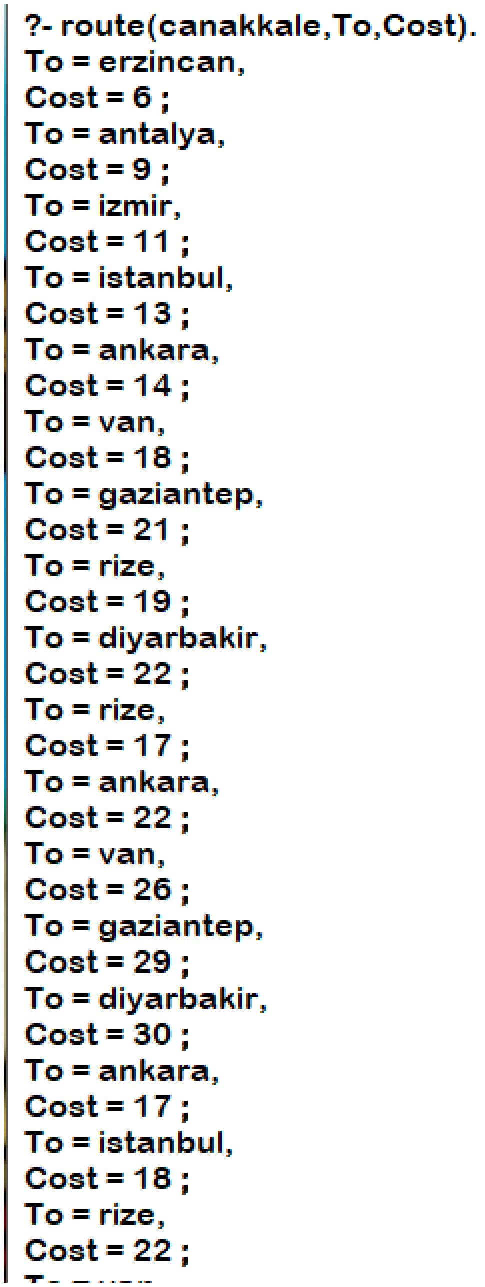
Rules:



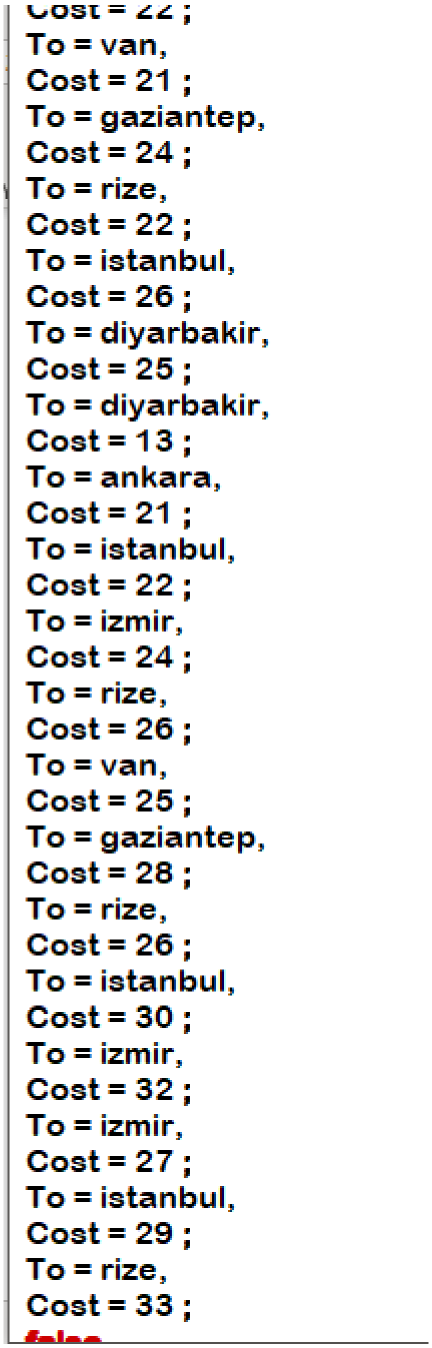
```
%rules
route(X,Y,C) :- route(X,Y,C,[]).
route(X,Y,C,Ok) :- flight(X,A,C1), not(member(A,Ok)),((Y = A, C is C1); (route(A,Y,C2,[X | Ok]),C is C2 + C1)).
```

Route to any location is a list of different possible ways, so a route query creates an empty route list. This list keeps intermediate stops between start location and finish location. By this way program ignores repeated answers.

Result:



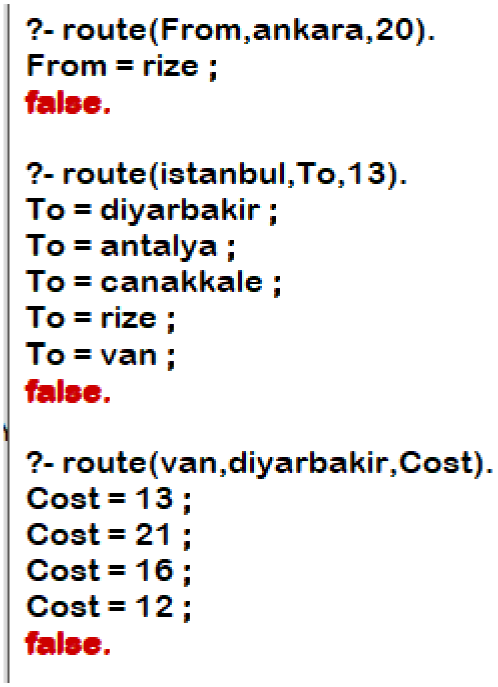
```
?- route(canakkale,To,Cost).
To = erzincan,
Cost = 6 ;
To = antalya,
Cost = 9 ;
To = izmir,
Cost = 11 ;
To = istanbul,
Cost = 13 ;
To = ankara,
Cost = 14 ;
To = van,
Cost = 18 ;
To = gaziantep,
Cost = 21 ;
To = rize,
Cost = 19 ;
To = diyarbakir,
Cost = 22 ;
To = rize,
Cost = 17 ;
To = ankara,
Cost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



```
Cost = 22 ;
To = van,
Cost = 21 ;
To = gaziantep,
Cost = 24 ;
To = rize,
Cost = 22 ;
To = istanbul,
Cost = 26 ;
To = diyarbakir,
Cost = 25 ;
To = diyarbakir,
Cost = 13 ;
To = ankara,
Cost = 21 ;
To = istanbul,
Cost = 22 ;
To = izmir,
Cost = 24 ;
To = rize,
Cost = 26 ;
To = van,
Cost = 25 ;
To = gaziantep,
Cost = 28 ;
To = rize,
Cost = 26 ;
To = istanbul,
Cost = 30 ;
To = izmir,
Cost = 32 ;
To = izmir,
Cost = 27 ;
To = istanbul,
Cost = 29 ;
To = rize,
Cost = 33 ;
false
```

Results show us all possible alternative routes for every location from Çanakkale by their costs.

And there is some different queries with answers:



```
?- route(From,ankara,20).
From = rize ;
false.

?- route(istanbul,To,13).
To = diyarbakir ;
To = antalya ;
To = canakkale ;
To = rize ;
To = van ;
false.

?- route(van,diyarbakir,Cost).
Cost = 13 ;
Cost = 21 ;
Cost = 16 ;
Cost = 12 ;
false.
```