Gebze Technical University
CSE341 - Programming Languages

Homework 3 - Report

Ahmet Tuğkan Ayhan
1901042692

Part 1

Facts

* In part 1, there are 5 different facts.
* First fact is **room.**
* If we consider the fact at **line 7** it says that, there is a room with ID **z23** and this room **capacity** is 50 and it has courses between **hours** 10-11.
* For the equipments this room has **projector and smartboard.** It also **hasAccess** for handicapped students.
* Second fact is **occupancy.**
* If we consider the fact at **line 13** it says that, the room with id **z23** is busy between hours **10-11** by course **cse321**
* Third fact is **course**. If we consider the fact at **line 19,** it says that course name is **cse341** and it's instructor is **'Yakup Genc'.** The capacity of the course is 4 people and course is held between **hours 8-9** at room **z08**. **Special need** for the course is **smartBoard.**
* Fourth fact is **instructor.**
* If we consider the fact at **line 25,** it says that the instructor name is **'Yakup Genc'** and he gives the course **cse341.** His preference for the courses is **smartBoard.**
* Fifth fact is **student.**
* If we consider the fact at **line 31,** it says that there is a student with name **'A'** and it takes courses **cse341, cse321 and cse331.** This student is also handicapped.

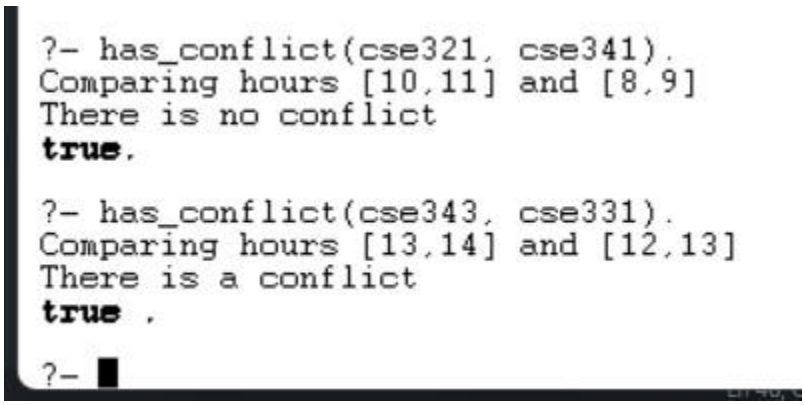
```
6  %    ID   Capacity    Hours           Special Equip.          Handicapped
7  room(z23  , 50 ,     [10,11]     ,  [projector,smartBoard]  ,   hasAccess).
8  room(z08  , 30 ,     [8,9,13,14] ,  [smartBoard]            ,   noAccess).
9  room(z10  , 40 ,     [12,13]     ,  [none]                  ,   hasAccess).
10 room(z11  , 40 ,     [15,16,17]  ,  [projector,smartBoard]  ,   noAccess).
11
12 %       Room ID  Hours     Course
13 occupancy(z23, [10,11] ,   cse321).
14 occupancy(z08, [8,9]   ,   cse341).
15 occupancy(z08, [13,14] ,   cse343).
16 occupancy(z10, [12,13] ,   cse331).
17
18 %     Course ID   Instructor        Capacity   Course Hours      Rooms  SpecialNeed
19 course(cse341, 'Yakup Genc'         , 4,        [8,9]         ,  [z08], smartBoard).
20 course(cse321, 'Didem Gozupek'      , 5,        [10,11]       ,  [z23], projector).
21 course(cse331, 'Alp Arslan Bayrakci', 3,        [12,13]       ,  [z10], none).
22 course(cse343, 'Habil Kalkan'       , 2,        [13,14]       ,  [z08], smartBoard).
23
24 %          Instructor Name         Courses      Preference
25 instructor('Yakup Genc'          , [cse341]   , smartBoard).
26 instructor('Didem Gozupek'       , [cse321]   , smartBoard).
27 instructor('Alp Arslan Bayrakci' , [cse331]   , none).
28 instructor('Habil Kalkan'        , [cse343]   , smartBoard).
29
30 %    Student ID       Courses          Handicapped or not
31 student('A' , [cse341, cse321, cse331] , handicapped   ).
32 student('C' , [cse331, cse321]         , notHandicapped).
33 student('D' , [cse343]                 , handicapped   ).
34 student('H' , [cse331]                 , handicapped   ).
35 student('L' , [cse341, cse321, cse343] , notHandicapped).
```

Rules

* There is 13 rules in part1 and some of them are looks very similar with eact other.

has_conflict (query1)

* If course hours overlap with each other, it means there is a conflict.



which_room (query2)

```
?- which_room(cse341).
Room                              : z23
Course Instructor                 : Yakup Genc
Instructor preference             : smartBoard
Course special need               : smartBoard
Room z23 has these equipments     : [projector,smartBoard]
Room has courses at these hours   : [10,11]
Course cse341 held at these hours : [8,9]
-> Equipments are enough and there are no time conflicts.
-> Course cse341 can held at room z23
true ;
Room                              : z11
Course Instructor                 : Yakup Genc
Instructor preference             : smartBoard
Course special need               : smartBoard
Room z11 has these equipments     : [projector,smartBoard]
Room has courses at these hours   : [15,16,17]
Course cse341 held at these hours : [8,9]
-> Equipments are enough and there are no time conflicts.
-> Course cse341 can held at room z11
true ;
```

which_room_to_where (query3)

```
?- which_room_to_where().
Room                              : z23
Course Instructor                 : Yakup Genc
Instructor preference             : smartBoard
Course special need               : smartBoard
Room z23 has these equipments     : [projector,smartBoard]
Room has courses at these hours   : [10,11]
Course cse341 held at these hours : [8,9]
-> Equipments are enough and there are no time conflicts.
-> Course cse341 can held at room z23
true ;
Room                              : z11
Course Instructor                 : Yakup Genc
Instructor preference             : smartBoard
Course special need               : smartBoard
Room z11 has these equipments     : [projector,smartBoard]
Room has courses at these hours   : [15,16,17]
Course cse341 held at these hours : [8,9]
-> Equipments are enough and there are no time conflicts.
-> Course cse341 can held at room z11
true ;
Room                              : z11
Course Instructor                 : Didem Gozupek
Instructor preference             : smartBoard
Course special need               : projector
Room z11 has these equipments     : [projector,smartBoard]
Room has courses at these hours   : [15,16,17]
Course cse321 held at these hours : [10,11]
-> Equipments are enough and there are no time conflicts.
-> Course cse321 can held at room z11
true ;
Room                              : z23
Course Instructor                 : Habil Kalkan
Instructor preference             : smartBoard
Course special need               : smartBoard
Room z23 has these equipments     : [projector,smartBoard]
Room has courses at these hours   : [10,11]
Course cse343 held at these hours : [13,14]
-> Equipments are enough and there are no time conflicts.
-> Course cse343 can held at room z23
true ;
Room                              : z11
Course Instructor                 : Habil Kalkan
Instructor preference             : smartBoard
Course special need               : smartBoard
Room z11 has these equipments     : [projector,smartBoard]
Room has courses at these hours   : [15,16,17]
Course cse343 held at these hours : [13,14]
-> Equipments are enough and there are no time conflicts.
-> Course cse343 can held at room z11
true.

?- ;
```

can_enroll (query4)

```
?- can_enroll('D', cse321).
Student         : D
Condition       : handicapped
Course          : cse321
Course held at  : z23
Room Status     : hasAccess
-> Student D can enroll to course cse321 at room z23
true .
```

```
?- can_enroll('C', cse321).
Student         : C
Condition       : notHandicapped
Course          : cse321
Course held at  : z23
-> Student C can enroll to course cse321 at room z23
true .

?-
```

which_classes_can_be_assigned (query5)

```
?- which_classes_can_be_assigned('A').
Student         : A
Condition       : handicapped
Course          : cse321
Course held at  : z23
Room Status     : hasAccess
-> Student A can enroll to course cse321 at room z23
true ;
Student         : A
Condition       : handicapped
Course          : cse331
Course held at  : z10
Room Status     : hasAccess
-> Student A can enroll to course cse331 at room z10
true .
```

```
?- which_classes_can_be_assigned('C').
Student         : C
Condition       : notHandicapped
Course          : cse341
Course held at  : z08
-> Student C can enroll to course cse341 at room z08
true ;
Student         : C
Condition       : notHandicapped
Course          : cse321
Course held at  : z23
-> Student C can enroll to course cse321 at room z23
true ;
Student         : C
Condition       : notHandicapped
Course          : cse331
Course held at  : z10
-> Student C can enroll to course cse331 at room z10
true ;
Student         : C
Condition       : notHandicapped
Course          : cse343
Course held at  : z08
-> Student C can enroll to course cse343 at room z08
true .

?-
```

add_student

```
?- listing(student).
:- dynamic student/3.

student('A', [cse341, cse321, cse331], handicapped).
student('C', [cse331, cse321], notHandicapped).
student('D', [cse343], handicapped).
student('H', [cse331], handicapped).
student('L', [cse341, cse321, cse343], notHandicapped).

true.

?- add_student('B', cse331, handicapped).
true.

?- listing(student).
:- dynamic student/3.

student('A', [cse341, cse321, cse331], handicapped).
student('C', [cse331, cse321], notHandicapped).
student('D', [cse343], handicapped).
student('H', [cse331], handicapped).
student('L', [cse341, cse321, cse343], notHandicapped).
student('B', [cse331], handicapped).

true.

?- which_classes_can_be_assigned('B').
Student        : B
Condition      : handicapped
Course         : cse321
Course held at : z23
Room Status    : hasAccess
-> Student B can enroll to course cse321 at room z23
true ;
Student        : B
Condition      : handicapped
Course         : cse331
Course held at : z10
Room Status    : hasAccess
-> Student B can enroll to course cse331 at room z10
true .

?-
```

add_course

```
?- listing(course).
:- dynamic course/6.

course(cse341, 'Yakup Genc', 4, [8, 9], [z08], smartBoard).
course(cse321, 'Didem Gozupek', 5, [10, 11], [z23], projector).
course(cse331, 'Alp Arslan Bayrakci', 3, [12, 13], [z10], none).
course(cse343, 'Habil Kalkan', 2, [13, 14], [z08], smartBoard).

true.

?- add_course(cse999, 'Tugkan Ayhan', 7, [9,10], [z11], projector)
.
true.

?- listing(course).
:- dynamic course/6.

course(cse341, 'Yakup Genc', 4, [8, 9], [z08], smartBoard).
course(cse321, 'Didem Gozupek', 5, [10, 11], [z23], projector).
course(cse331, 'Alp Arslan Bayrakci', 3, [12, 13], [z10], none).
course(cse343, 'Habil Kalkan', 2, [13, 14], [z08], smartBoard).
course(cse999, 'Tugkan Ayhan', 7, [9, 10], [z11], projector).

true.
```

add_room

```
?- listing(room).
:- dynamic room/5.

room(z23, 50, [10, 11], [projector, smartBoard], hasAccess).
room(z08, 30, [8, 9, 13, 14], [smartBoard], noAccess).
room(z10, 40, [12, 13], [none], hasAccess).
room(z11, 40, [15, 16, 17], [projector, smartBoard], noAccess).

true.

?- add_room(z30, 20, [11,12], [projector,smartBoard], hasAccess).
true.

?- listing(room).
:- dynamic room/5.

room(z23, 50, [10, 11], [projector, smartBoard], hasAccess).
room(z08, 30, [8, 9, 13, 14], [smartBoard], noAccess).
room(z10, 40, [12, 13], [none], hasAccess).
room(z11, 40, [15, 16, 17], [projector, smartBoard], noAccess).
room(z30, 20, [11, 12], [projector, smartBoard], hasAccess).

true.

?- which_room(cse341).
Room                              : z23
Course Instructor                 : Yakup Genc
Instructor preference             : smartBoard
Course special need               : smartBoard
Room z23 has these equipments     : [projector,smartBoard]
Room has courses at these hours   : [10,11]
Course cse341 held at these hours : [8,9]
-> Equipments are enough and there are no time conflicts.
-> Course cse341 can held at room z23
true ;
Room                              : z11
Course Instructor                 : Yakup Genc
Instructor preference             : smartBoard
Course special need               : smartBoard
Room z11 has these equipments     : [projector,smartBoard]
Room has courses at these hours   : [15,16,17]
Course cse341 held at these hours : [8,9]
-> Equipments are enough and there are no time conflicts.
-> Course cse341 can held at room z11
true ;
Room                              : z30
Course Instructor                 : Yakup Genc
Instructor preference             : smartBoard
Course special need               : smartBoard
Room z30 has these equipments     : [projector,smartBoard]
Room has courses at these hours   : [11,12]
Course cse341 held at these hours : [8,9]
-> Equipments are enough and there are no time conflicts.
-> Course cse341 can held at room z30
true.

?-
```

Part 2

Facts

* In part2, there is only two type of facts.
* First and the most important one is **flight**.
* This facts says that, there is a route from X to Y and the length of that route is C.
* It is used to make a graph traversal using a couple of rules.
* Second fact is **last_element**

```
64
65  last_element(X,[X]).
66
```

* It means that, if list Y(which is [X]) has only one element, then this last element is equal to X.
* This fact is used with **last_element** rule.

```
66
67  last_element(X,[_|Z]):-
68      last_element(X,Z).
```

* This rule recursively calls itself until every head element is removed.
* If there is only one element left, then it calls **last_element fact** and assigns it's value to X.
* For example if our list is [2,5,3,6] then the list turns into this:

-> [2,5,3,6]
-> [2] | [5,3,6] (2 is discarded)
-> [5] | [3,6] (5 is discarded)
-> [3] | [6] (3 is discarded)
-> [6] has one element then, X = [6]

* By doing so I can get last element of a list. I used this value while printing the route.

```
1   %% Knowledge-base
2   % canakkale
3   flight(canakkale, erzincan, 6).
4   % erzincan
5   flight(erzincan, canakkale, 6).
6   flight(erzincan, antalya, 3).
7   % antalya
8   flight(antalya, erzincan, 3).
9   flight(antalya, izmir, 2).
10  flight(antalya, diyarbakir, 4).
11  % izmir
12  flight(izmir, antalya, 2).
13  flight(izmir, istanbul, 2).
14  flight(izmir, ankara, 6).
15  % istanbul
16  flight(istanbul, izmir, 2).
17  flight(istanbul, ankara, 1).
18  flight(istanbul, rize, 4).
19  % ankara
20  flight(ankara, izmir, 6).
21  flight(ankara, istanbul, 1).
22  flight(ankara, rize, 5).
23  flight(ankara, van, 4).
24  flight(ankara, diyarbakir, 8).
25  % rize
26  flight(rize, istanbul, 4).
27  flight(rize, ankara, 5).
28  % diyarbakir
29  flight(diyarbakir, ankara, 8).
30  flight(diyarbakir, antalya, 4).
31  % van
32  flight(van, ankara, 4).
33  flight(van, gaziantep, 3).
34  % gaziantep
35  flight(gaziantep, van, 3).
36
```

Rules

There are total of 5 rules in part2, but since I explained one of the rules (which is **last_element**) I will explain the other 4.

```
36
37  %% Rules
38  route(X,Y,C):-
39      flight(X,Y,C).
40
41  route(X,Y,C):-
42      visit(X,Y,0,[X],C).
43
44  visit(X,Y,C,Visited,WantedCost):-
45      flight(X,Y,Cost),
46      not(member(Y, Visited)),
47      FinalCost is Cost + C,
48      FinalCost = WantedCost,
49      reverse(Visited, Path),
50      append(Path, [Y], FinalPath),
51      last_element(X2, Visited),
52      nl,
53      format('Start  : ~w',[X2]), nl,
54      format('Finish : ~w',[Y]), nl,
55      format('Path   : ~w',[FinalPath]), nl.
56
57
58  visit(X,Y,C,Visited,WantedCost):-
59      flight(X,Z,Cost),
60      Y \== Z,
61      not(member(Z, Visited)),
62      TotalCost is C + Cost,
63      visit(Z,Y, TotalCost, [Z|Visited],WantedCost).
64
65  last_element(X,[X]).
66
67  last_element(X,[_|Z]):-
68      last_element(X,Z).
69
```

* 1st rule **(line 38),** simply says that there is a **route** from X to Y with cost C if there is a **flight** from X to Y with cost C.
* This works only when there is no extra flight between X and Y cities.
* 2nd rule **(line 41)** solves that extra flight problem. It gives initial values to **visit** rule and calls it.
* It means if I can visit from X to Y with cost C, then there is a route from X to Y with cost C.
* Third rule **(line 44)** actually is a base case for fourth rule**(line 58)**. If fourth rule successfully finds a route from X to Y then it's first fact **(line 45)** returns true and it continues.
* If flight fact is true, then it must control if city Y is in the Visited places**(line 46)**. If so then it won't continue because there is no need to print same destination more than once.
* I added this condition because the recursive call which happens in fourt rule doesn't add last city to the **Visited** list. So I add it manually and if its already added, then to avoid adding it again I give that condition
* Since last city not added automatically, it's cost not added either. So I added it manually with **line 47**
* Then after getting final cost I compared it with given Cost value at **line 48** (it can be given also as _. if that happens it will directly return true at will print out FinalCost).

* If this comparison is false then it means user gave us a cost (for example: route(canakkale, van, 2)) and there is no route from canakkale to van with cost 2.

```
?- route(canakkale,van,2).
false.
```

* If the comparison is true then it means there is a route from canakkale to van with given cost.

```
?- route(canakkale,van,18).

Start  : canakkale
Finish : van
Path   : [canakkale,erzincan,antalya,izmir,istanbul,ankara,van]
true .

?-
```

* Lastly, if user gave us no cost value then it shows all routes from X to Y with any cost.

```
?- route(canakkale,van,Cost).

Start  : canakkale
Finish : van
Path   : [canakkale,erzincan,antalya,izmir,istanbul,ankara,van]
Cost = 18 ;

Start  : canakkale
Finish : van
Path   : [canakkale,erzincan,antalya,izmir,istanbul,rize,ankara,van]
Cost = 26 ;

Start  : canakkale
Finish : van
Path   : [canakkale,erzincan,antalya,izmir,ankara,van]
Cost = 21 ;

Start  : canakkale
Finish : van
Path   : [canakkale,erzincan,antalya,diyarbakir,ankara,van]
Cost = 25 ;
false.

?-
```

* To explain rest of the 3rd rule simply, I need to reverse the list since it would return [van, ..., .. ,.., canakkale] if I wouldn't reverse it. After reversing I added last element (which is van in this example) since it weren't in the list. Then to get value at "Start : canakkale" I used last_element rule. Finally, I printed the route with starting and finishing point + cost.

* In 4th rule, I said that if there is a flight from X to Z **and** if the city Z is not equal to Y **and** this city Z is not visited before **then** add flight cost of this city to **totalCost and** add this city to **Visited** list **and then** continue to visiting from that city until we find the city we want(which is Y).

Test Results

```
?- route(izmir,gaziantep,Cost).

Start  : izmir
Finish : gaziantep
Path   : [izmir,antalya,diyarbakir,ankara,van,gaziantep]
Cost = 21 ;

Start  : izmir
Finish : gaziantep
Path   : [izmir,istanbul,ankara,van,gaziantep]
Cost = 10 ;

Start  : izmir
Finish : gaziantep
Path   : [izmir,istanbul,rize,ankara,van,gaziantep]
Cost = 18 ;

Start  : izmir
Finish : gaziantep
Path   : [izmir,ankara,van,gaziantep]
Cost = 13 ;
false.

?- route(izmir,gaziantep,15).
false.

?- route(izmir,gaziantep,21).

Start  : izmir
Finish : gaziantep
Path   : [izmir,antalya,diyarbakir,ankara,van,gaziantep]
true .

?-
```

```
?- route(izmir,_,10).

Start  : izmir
Finish : gaziantep
Path   : [izmir,istanbul,ankara,van,gaziantep]
true ;

Start  : izmir
Finish : van
Path   : [izmir,ankara,van]
true ;
false.
```

```
?- route(izmir,_,Cost).
Cost = 2 ;
Cost = 2 ;
Cost = 6 ;

Start  : izmir
Finish : antalya
Path   : [izmir,antalya]
Cost = 2 ;

Start  : izmir
Finish : istanbul
Path   : [izmir,istanbul]
Cost = 2 ;

Start  : izmir
Finish : ankara
Path   : [izmir,ankara]
Cost = 6 ;

Start  : izmir
Finish : erzincan
Path   : [izmir,antalya,erzincan]
Cost = 5 ;

Start  : izmir
Finish : diyarbakir
Path   : [izmir,antalya,diyarbakir]
Cost = 6 ;

Start  : izmir
Finish : canakkale
Path   : [izmir,antalya,erzincan,canakkale]
Cost = 11 ;

Start  : izmir
Finish : ankara
Path   : [izmir,antalya,diyarbakir,ankara]
Cost = 14 ;

Start  : izmir
Finish : istanbul
Path   : [izmir,antalya,diyarbakir,ankara,istanbul]
Cost = 15 ;

Start  : izmir
Finish : rize
Path   : [izmir,antalya,diyarbakir,ankara,rize]
Cost = 19 ;

Start  : izmir
Finish : van
Path   : [izmir,antalya,diyarbakir,ankara,van]
Cost = 18
```