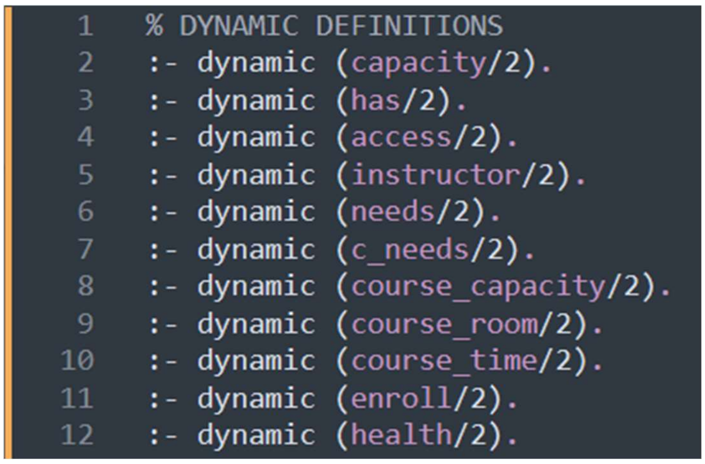
Part-1)

To be able to add new facts to the system, i convert procedures to dynamic instead of static.

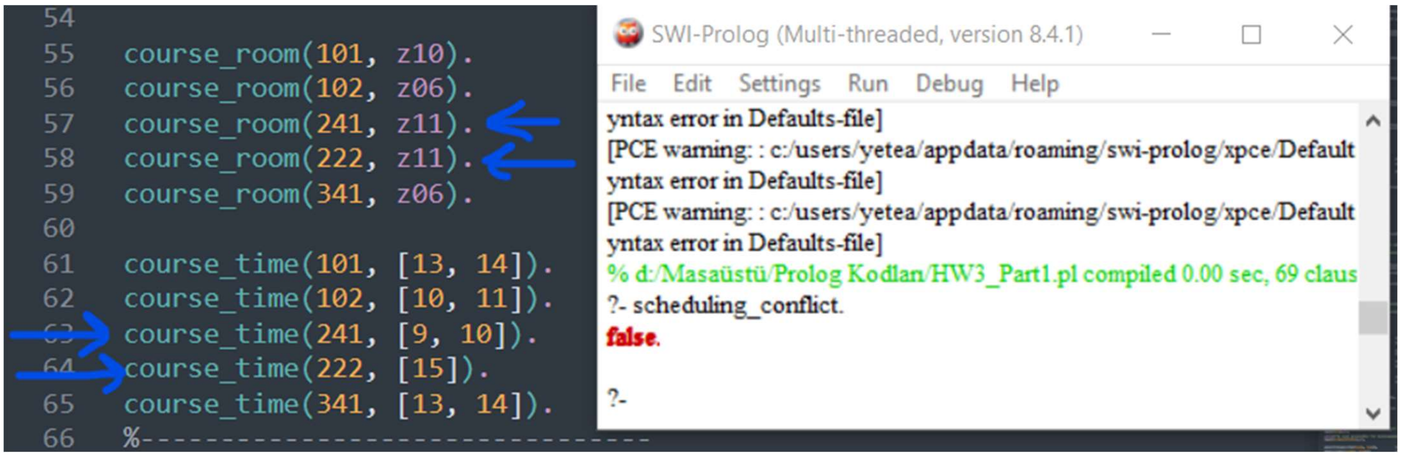
I write a lot of facts for test each situation. Because of that, my code became a bit long.



```
% DYNAMIC DEFINITIONS
:- dynamic (capacity/2).
:- dynamic (has/2).
:- dynamic (access/2).
:- dynamic (instructor/2).
:- dynamic (needs/2).
:- dynamic (c_needs/2).
:- dynamic (course_capacity/2).
:- dynamic (course_room/2).
:- dynamic (course_time/2).
:- dynamic (enroll/2).
:- dynamic (health/2).
```

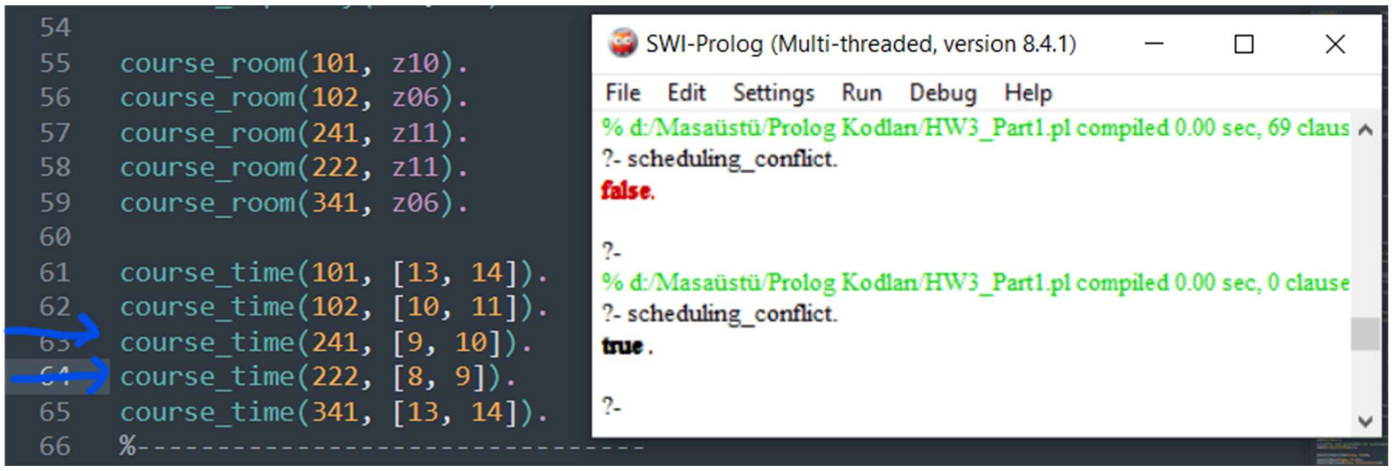
a) Check whether there is any scheduling conflict.

Firstly, I set schedules properly and when I check is there any scheduling conflict, program gives "false". Secondly, I changed the time of the 222 course to [8, 9]. These courses were using the same class before, now their time is conflicted too.



```

course_room(101, z10).
course_room(102, z06).
course_room(241, z11).
course_room(222, z11).
course_room(341, z06).

course_time(101, [13, 14]).
course_time(102, [10, 11]).
course_time(241, [9, 10]).
course_time(222, [15]).
course_time(341, [13, 14]).
%-------------------------------
```

```
yntax error in Defaults-file]
[PCE warning: : c:/users/yetea/appdata/roaming/swi-prolog/xpce/Default
yntax error in Defaults-file]
[PCE warning: : c:/users/yetea/appdata/roaming/swi-prolog/xpce/Default
yntax error in Defaults-file]
% d:/Masaüstü/Prolog Kodları/HW3_Part1.pl compiled 0.00 sec, 69 claus
?- scheduling_conflict.
false.

?-
```

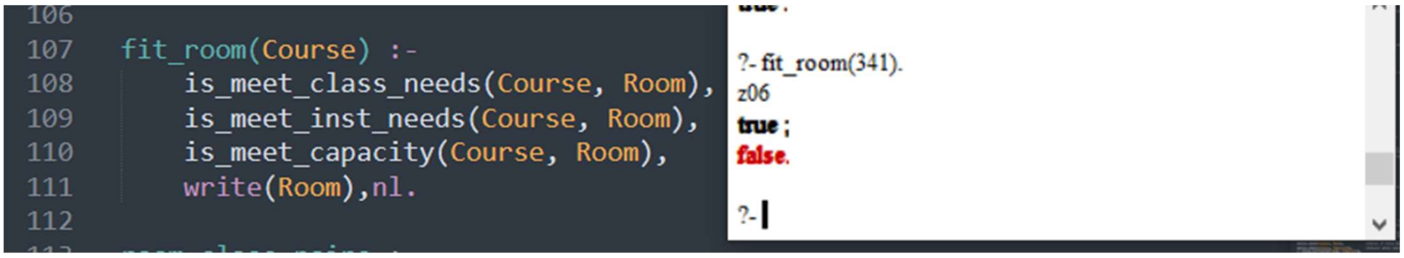


```

course_room(101, z10).
course_room(102, z06).
course_room(241, z11).
course_room(222, z11).
course_room(341, z06).

course_time(101, [13, 14]).
course_time(102, [10, 11]).
course_time(241, [9, 10]).
course_time(222, [8, 9]).
course_time(341, [13, 14]).
%-------------------------------
```

```
% d:/Masaüstü/Prolog Kodları/HW3_Part1.pl compiled 0.00 sec, 69 claus
?- scheduling_conflict.
false.

?-
% d:/Masaüstü/Prolog Kodları/HW3_Part1.pl compiled 0.00 sec, 0 clause
?- scheduling_conflict.
true .

?-
```

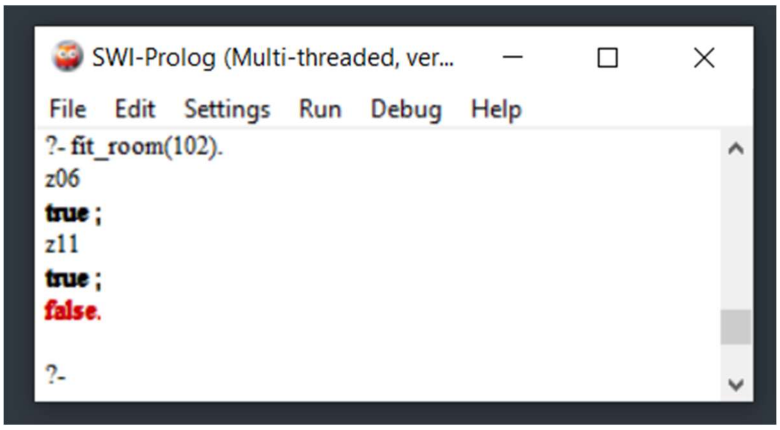
b) Check which room can be assigned to a given class.

When I want to find suitable rooms for 341 course, program gives only z06 as a solution, because only z06's capacity is bigger than 341's capacity.



```
fit_room(Course) :-
    is_meet_class_needs(Course, Room),
    is_meet_inst_needs(Course, Room),
    is_meet_capacity(Course, Room),
    write(Room),nl.
```

```
?- fit_room(341).
z06
true ;
false.

?-
```


SWI-Prolog (Multi-threaded, ver...
File Edit Settings Run Debug Help

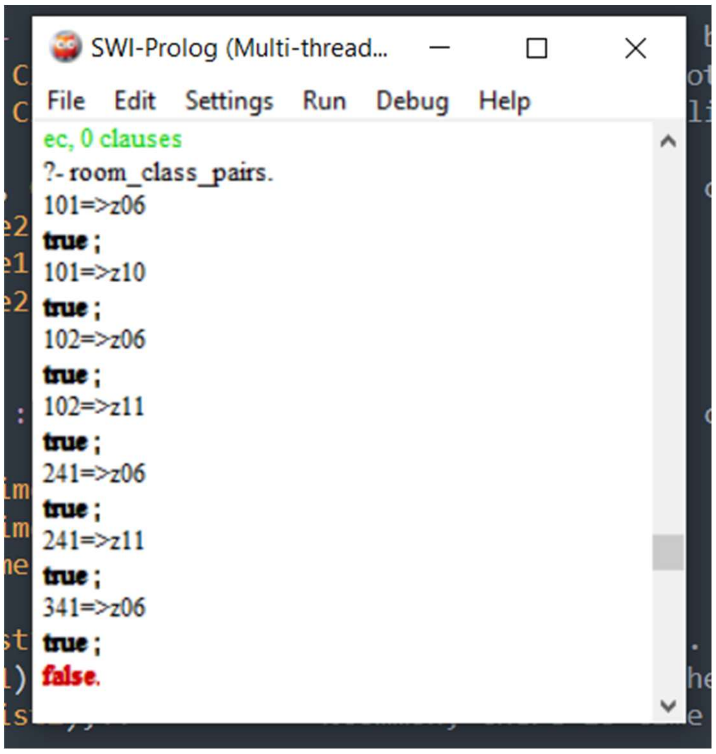
```
?- fit_room(102).
z06
true ;
z11
true ;
false.

?-
```

Z06 and Z11 are fit for 102 course, Z10 is not. Because 102 course needs smart board but z10 has only projector.

c) Check which room can be assigned to which classes.

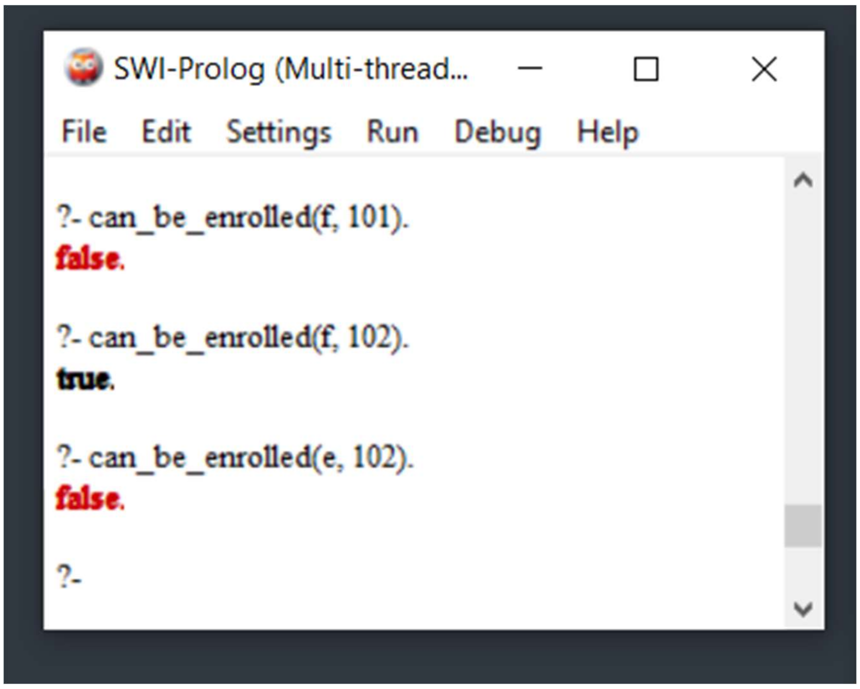
In this part, I check if is room meet class needs, is room meet classes instructor's needs and is room meet classes capacity. And if answer is yes, I print this room&class pair. I do same process for all classes.

These are my program's suitable room&class pairs.


SWI-Prolog (Multi-thread...
File Edit Settings Run Debug Help

```
ec, 0 clauses
?- room_class_pairs.
101=>z06
true ;
101=>z10
true ;
102=>z06
true ;
102=>z11
true ;
241=>z06
true ;
241=>z11
true ;
341=>z06
true ;
false.
```

d)Check whether a student can be enrolled to a given class

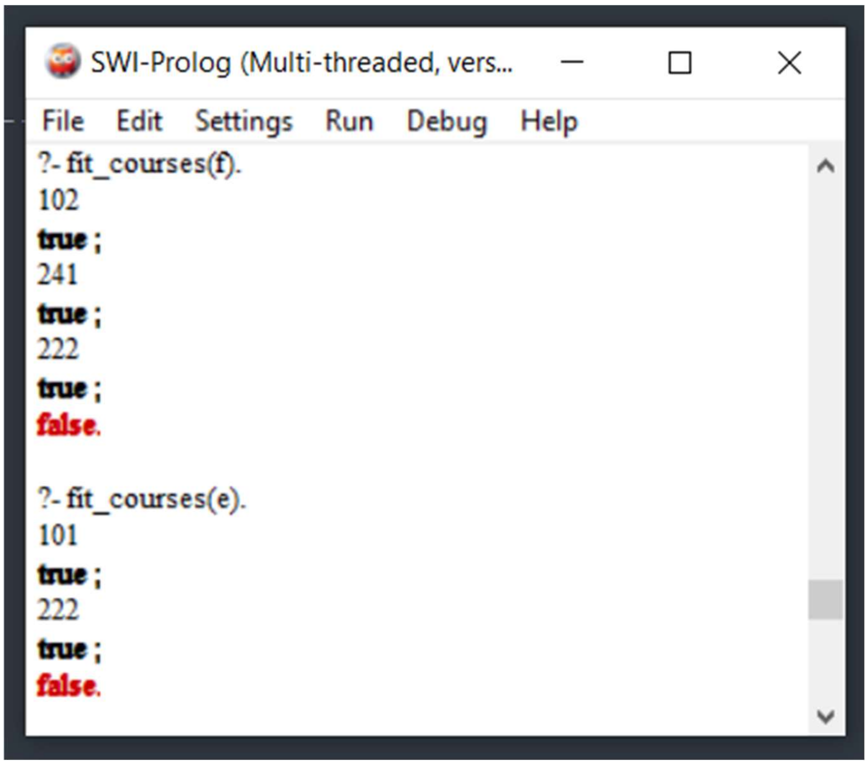

SWI-Prolog (Multi-thread...
File Edit Settings Run Debug Help

```
?- can_be_enrolled(f, 101).
false.

?- can_be_enrolled(f, 102).
true.

?- can_be_enrolled(e, 102).
false.

?-
```

Student ‘f’ can’t be enrolled to 101 courses, because 101 and 341 are at the same time. But he/she can be enrolled to 102 courses, there is no obstacle to attend course.
Student ‘e’ can’t be enrolled to 102 courses, because he/she is handicapped, and 102 courses is held on z06. Z06 is not accessible for handicapped students.

e) Check which classes a student can be assigned.

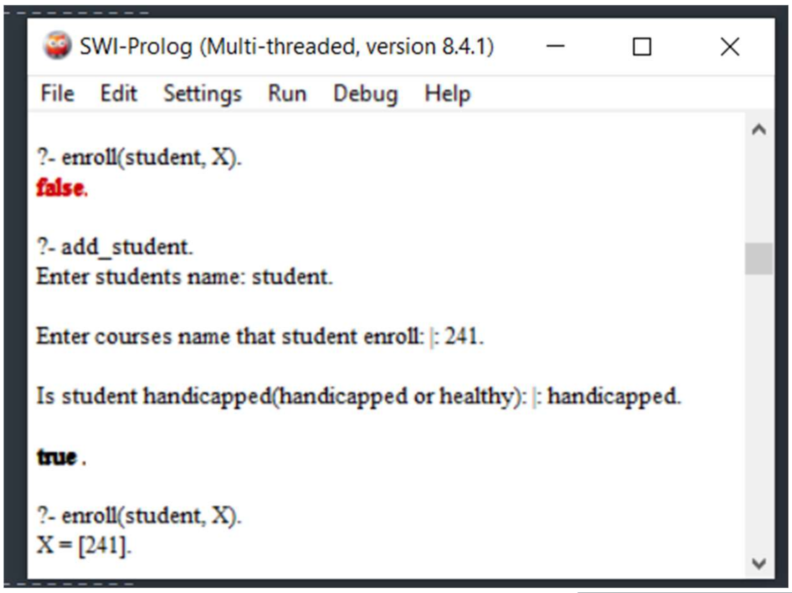
This part of code finds all courses that given student can be enrolled.


SWI-Prolog (Multi-threaded, vers...
File Edit Settings Run Debug Help

```
?- fit_courses(f).
102
true ;
241
true ;
222
true ;
false.

?- fit_courses(e).
101
true ;
222
true ;
false.
```

f) Adding Student, Course and Room to the System.

In this part, we can add new student, course or room to the system. If you want to add new student to the system, program wants some information about student, like his/her name, name of the course he/she taken and is student handicapped. But program doesn’t add these facts immediately. Before adding these facts, checks if this student can be enrolled to this course. Because maybe student is handicapped, and the course room doesn’t fit for handicapped people. Or if you want to add new course to the system, program ask you for the name of the course, instructor of the course, what is needed for the course (like projector), capacity of the course, course room and course time. But as I mentioned before, system doesn’t add these facts immediately. For example, program will look for instructor’s schedule, if there is conflict between new courses time and instructor’s schedule, it won’t add this fact and will print false as a result. Or if course room doesn’t meet courses needs, program won’t assign this room as courses room.

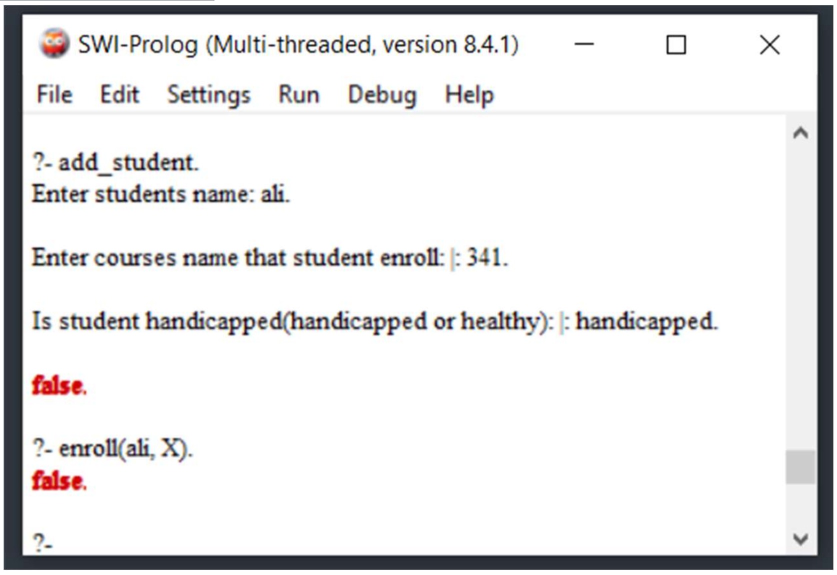

SWI-Prolog (Multi-threaded, version 8.4.1)
File Edit Settings Run Debug Help

```
?- enroll(student, X).
false.

?- add_student.
Enter students name: student.

Enter courses name that student enroll: |: 241.

Is student handicapped(handicapped or healthy): |: handicapped.

true .

?- enroll(student, X).
X = [241].
```

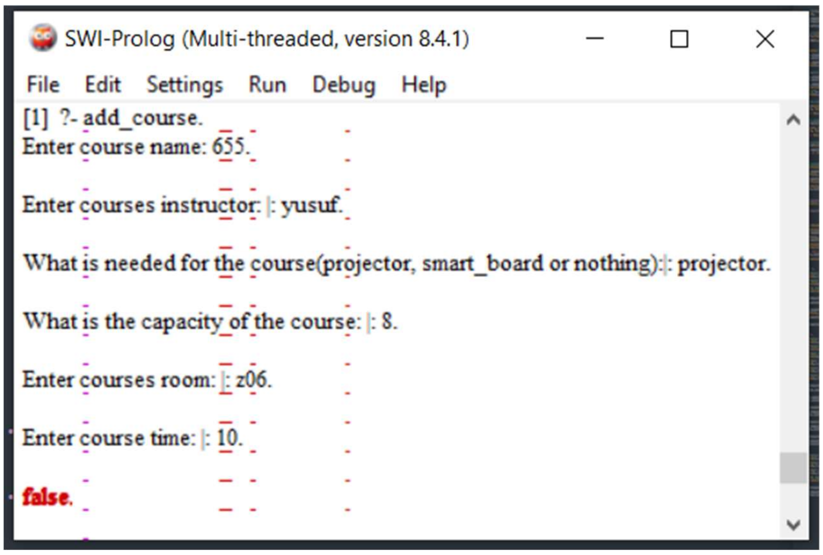
Here is some examples.

I added new student to the system. It checks given information internally and If there is no problem for the student to take the course, program assign this course to student.

In second example, Ali is handicapped student and course's room is not accessible for handicapped students. Therefore, program prints “false” and doesn't assign this course to Ali. When we check, we can see that Ali have no courses.


SWI-Prolog (Multi-threaded, version 8.4.1)
File Edit Settings Run Debug Help

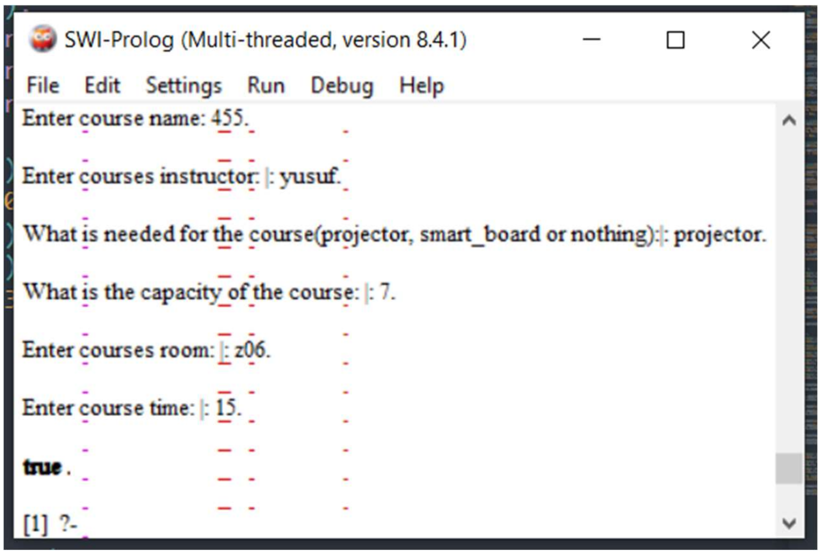
```
?- add_student.
Enter students name: ali.

Enter courses name that student enroll: |: 341.

Is student handicapped(handicapped or healthy): |: handicapped.

false.

?- enroll(ali, X).
false.

?-
```



SWI-Prolog (Multi-threaded, version 8.4.1)
File Edit Settings Run Debug Help

```
[1] ?- add_course.
Enter course name: 655.

Enter courses instructor: |: yusuf.

What is needed for the course(projector, smart_board or nothing):|: projector.

What is the capacity of the course: |: 8.

Enter courses room: |: z06.

Enter course time: |: 10.

false.
```

It prints "false”, and it won’t assign instructor to course, because sir Yusuf’s course hours conflicted with new course’s hours.

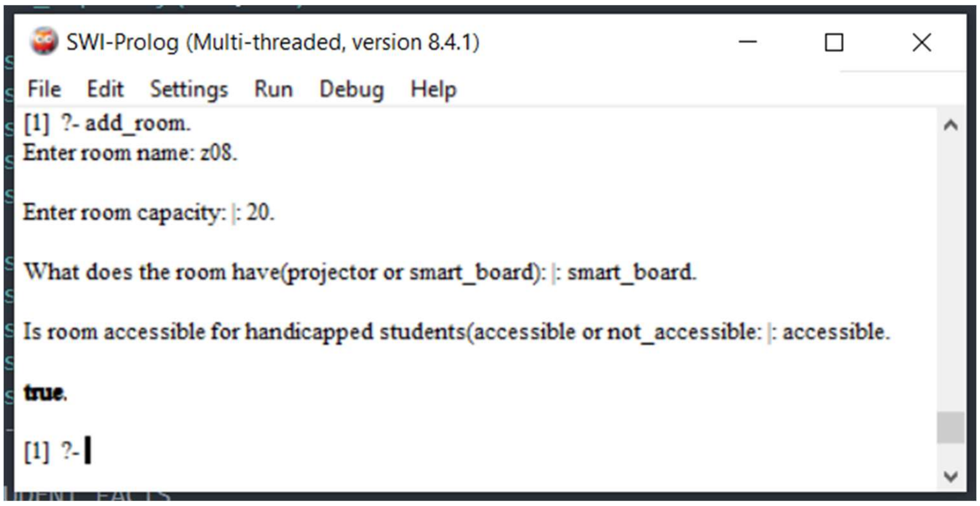


SWI-Prolog (Multi-threaded, version 8.4.1)
File Edit Settings Run Debug Help

```
Enter course name: 455.

Enter courses instructor: |: yusuf.

What is needed for the course(projector, smart_board or nothing):|: projector.

What is the capacity of the course: |: 7.

Enter courses room: |: z06.

Enter course time: |: 15.

true .

[1] ?-
```

Its ok now.

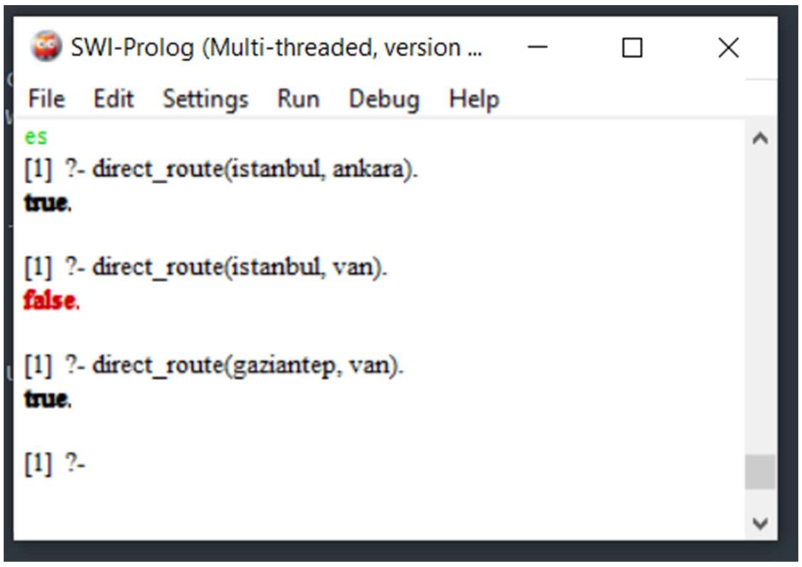
And its adding room



SWI-Prolog (Multi-threaded, version 8.4.1)
File Edit Settings Run Debug Help

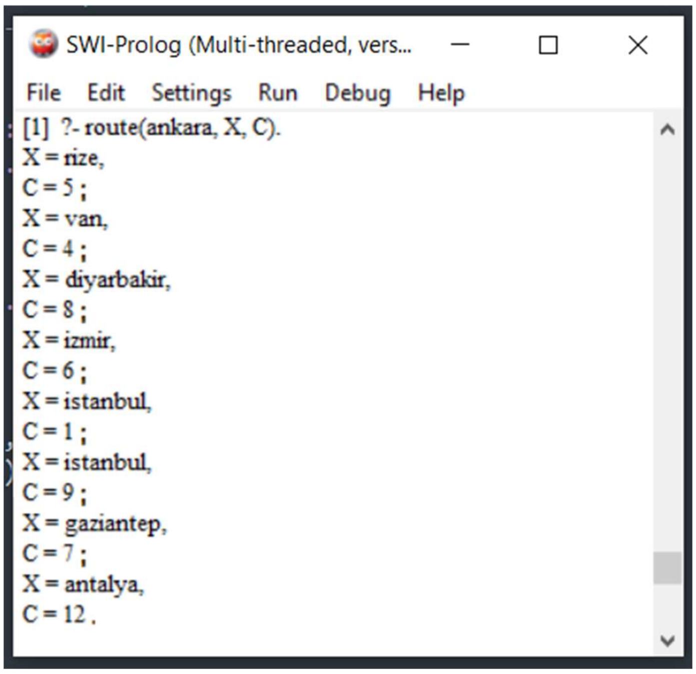
```
[1] ?- add_room.
Enter room name: z08.

Enter room capacity: |: 20.

What does the room have(projector or smart_board): |: smart_board.

Is room accessible for handicapped students(accessible or not_accessible: |: accessible.

true.

[1] ?-
```

Part-2)


SWI-Prolog (Multi-threaded, version ...
File Edit Settings Run Debug Help

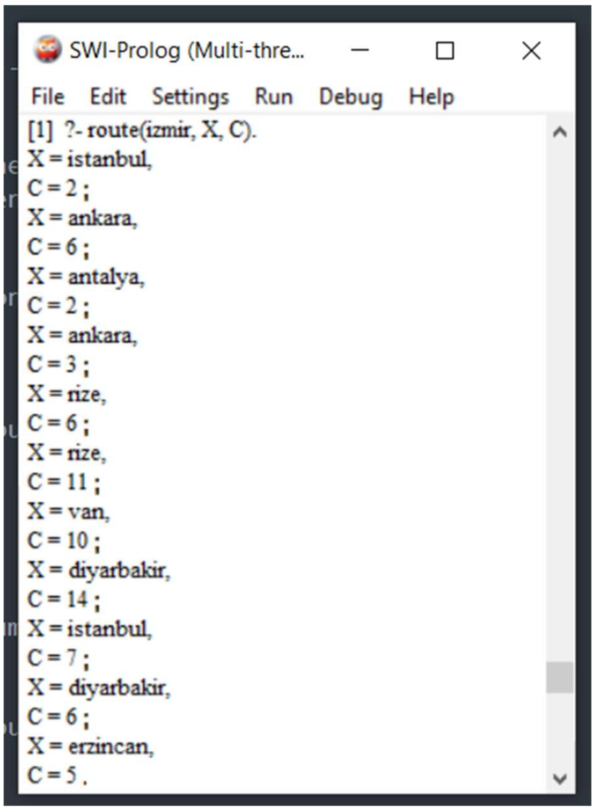
```
es
[1] ?- direct_route(istanbul, ankara).
true.

[1] ?- direct_route(istanbul, van).
false.

[1] ?- direct_route(gaziantep, van).
true.

[1] ?-
```

Checks if there is direct route between given two cities.


SWI-Prolog (Multi-threaded, vers...
File Edit Settings Run Debug Help

```
[1] ?- route(ankara, X, C).
X = rize,
C = 5 ;
X = van,
C = 4 ;
X = diyarbakir,
C = 8 ;
X = izmir,
C = 6 ;
X = istanbul,
C = 1 ;
X = istanbul,
C = 9 ;
X = gaziantep,
C = 7 ;
X = antalya,
C = 12 .
```

it can list all the connected cities for a given city.



```
[1] ?- route(izmir, X, C).
X = istanbul,
C = 2 ;
X = ankara,
C = 6 ;
X = antalya,
C = 2 ;
X = ankara,
C = 3 ;
X = rize,
C = 6 ;
X = rize,
C = 11 ;
X = van,
C = 10 ;
X = diyarbakir,
C = 14 ;
X = istanbul,
C = 7 ;
X = diyarbakir,
C = 6 ;
X = erzincan,
C = 5 .
```

Route will work until press enter, because there are infinity combinations to go somewhere from given city. Or in other words, plane can always flight to somewhere from somewhere with same passenger, until its broken.