

# **CMPE 540 PRINCIPLES OF AI**

## **PROJECT REPORT**

Course Advisor Expert System : A PROLOG Project

by

M. Toygar KARADENİZ

Bogaziçi University

1996

## Table Of Contents

|  |           |
|--|-----------|
| <b>TABLE OF CONTENTS .....</b>             | <b>1</b>  |
| <b>INTRODUCTION .....</b>                  | <b>2</b>  |
| <b>DOMAIN OF EXPERTISE.....</b>            | <b>2</b>  |
| <b>ABOUT SYSTEM IMPLEMENTATION.....</b>    | <b>3</b>  |
| <b>COMMAND PROMPT .....</b>                | <b>4</b>  |
| <b>EXAMPLE DIALOGUES .....</b>             | <b>5</b>  |
| <b>CONCLUSION .....</b>                    | <b>10</b> |
| <b>APPENDIX - PROLOG SOURCE CODES.....</b> | <b>11</b> |

## **Introduction**

Maybe, the most boring part of the university life, is the registration process through which the students decide on their next semester courses, discuss them with their advisor and finally, take the relevant ones. This process is tiring for both the advisor and the students. The situation becomes even more terrible when the advisor is not much aware about the course schedule and the background of the student. At this point, it sounds good to have a system that automates that process and overcomes all the trouble.

The following report is an implementation for such a course advisor system. The mentioned system has been developed in a PC environment using SWI-Prolog logic language and expert system techniques. The question of why the expert system techniques are more suitable for such a system than any other technique is an open question. But, there is no doubt that if all the university advisors are course advising experts, then the process of assigning courses would be a simpler and shorter process.

## **Domain Of Expertise**

The general domain on which this expert system works, is mainly the database of courses, departments, students, backgrounds, prerequisites etc. Shortly, the domain consists of course schedule and the backgrounds of the students. The problem of course assignment is mainly the consideration of the database facts which are background and schedule records, deciding on new courses to assign to a particular student and checking the validity of these assignments one by one. So, the system uses a complex inference engine that keeps up with many information particles.

## About System Implementation

As mentioned before, Course Advisor Expert (CAE) System is implemented under SWI-Prolog. Theoretical expert system techniques of AI are tried to be used to a full extend. The system mainly consists of two modules.

- **Knowledge Base** : The first module is the knowledge base which is placed in the file named 'data.pl'. This knowledge base captures the data which is totally separable from the expert system inference rules. So, it has the relevant representations of the courses, students, backgrounds of students, schedule etc. plus some constants which are going to be used by the engine. These data are all declarative data and so are represented as Prolog facts. The knowledge base contains no rules.
- **Inference Engine** : The second module of the program is the inference engine by the help of which the system does all the decisions. Inference engine consists fully of inference rules and no facts. Some shell techniques are used in addition to inference rules, like the user prompt. Every other specific knowledge about the system is moved to the knowledge base to preserve the independence between knowledge and inference.

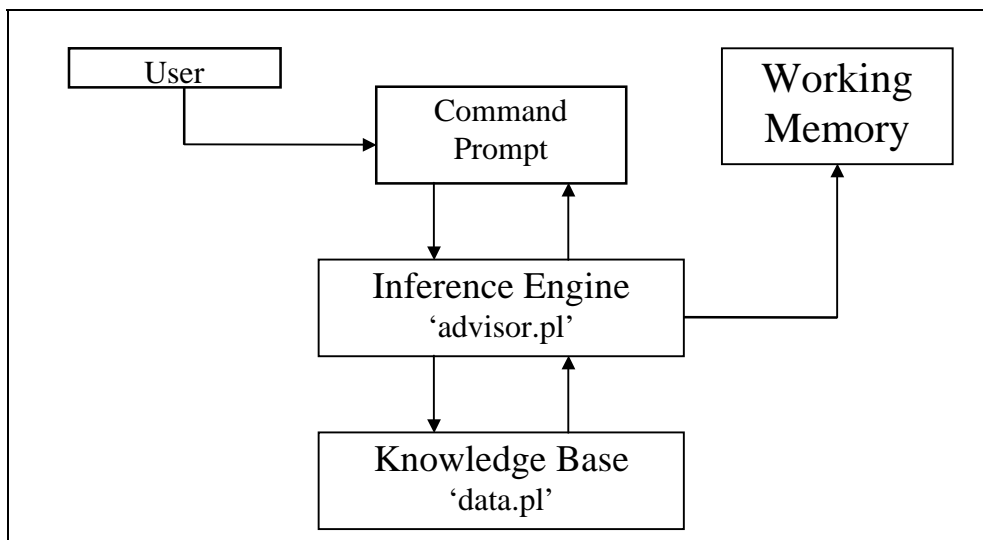


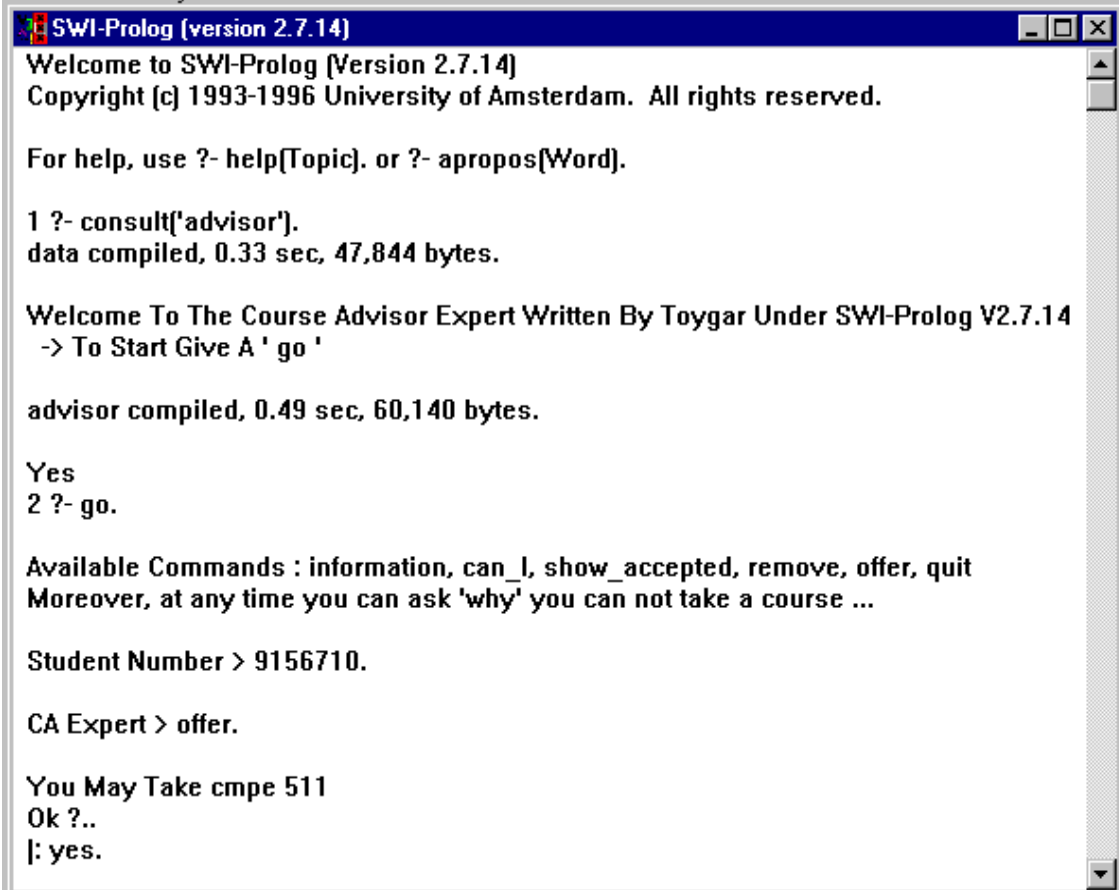
Figure 1

## Command Prompt

CAE system interacts with the user through a command loop. The user can enter '**information,**' '**can\_I,**' '**show\_accepted,**' '**remove,**' '**offer,**' and '**quit**' commands. Moreover, at any time the user can ask to the system '**why**' he/she can not take a specific course. The user commands do several operations summarized as follows,

- **information** : This command gives a two line information about the available commands to the user.
- **can\_I** : By the help of '**can\_I**' command the user can ask for taking a special course. If that course has no problem to be taken by the student, it is added to the taken courses so far.
- **show\_accepted** : This command shows the list of the taken courses for this semester so far.
- **remove** : Using this command, enables the user to give up a signed course.
- **offer** : This is the main command of the program used to offer the suitable courses to the students.
- **why** : This command is mainly used to ask the system why the user can not take a specific course. The system will respond to this command by a list of causes.
- **quit** : Entering this command will close the CAE system and return the control to the Prolog command loop.

## Example Dialogues



```
SWI-Prolog (version 2.7.14)
Welcome to SWI-Prolog (Version 2.7.14)
Copyright (c) 1993-1996 University of Amsterdam. All rights reserved.

For help, use ?- help[Topic]. or ?- apropos[Word].

1 ?- consult('advisor').
data compiled, 0.33 sec, 47,844 bytes.

Welcome To The Course Advisor Expert Written By Toygar Under SWI-Prolog V2.7.14
-> To Start Give A ' go '

advisor compiled, 0.49 sec, 60,140 bytes.

Yes
2 ?- go.

Available Commands : information, can_l, show_accepted, remove, offer, quit
Moreover, at any time you can ask 'why' you can not take a course ...

Student Number > 9156710.

CA Expert > offer.

You May Take cmpe 511
Ok ?..
|: yes.
```

Figure 2

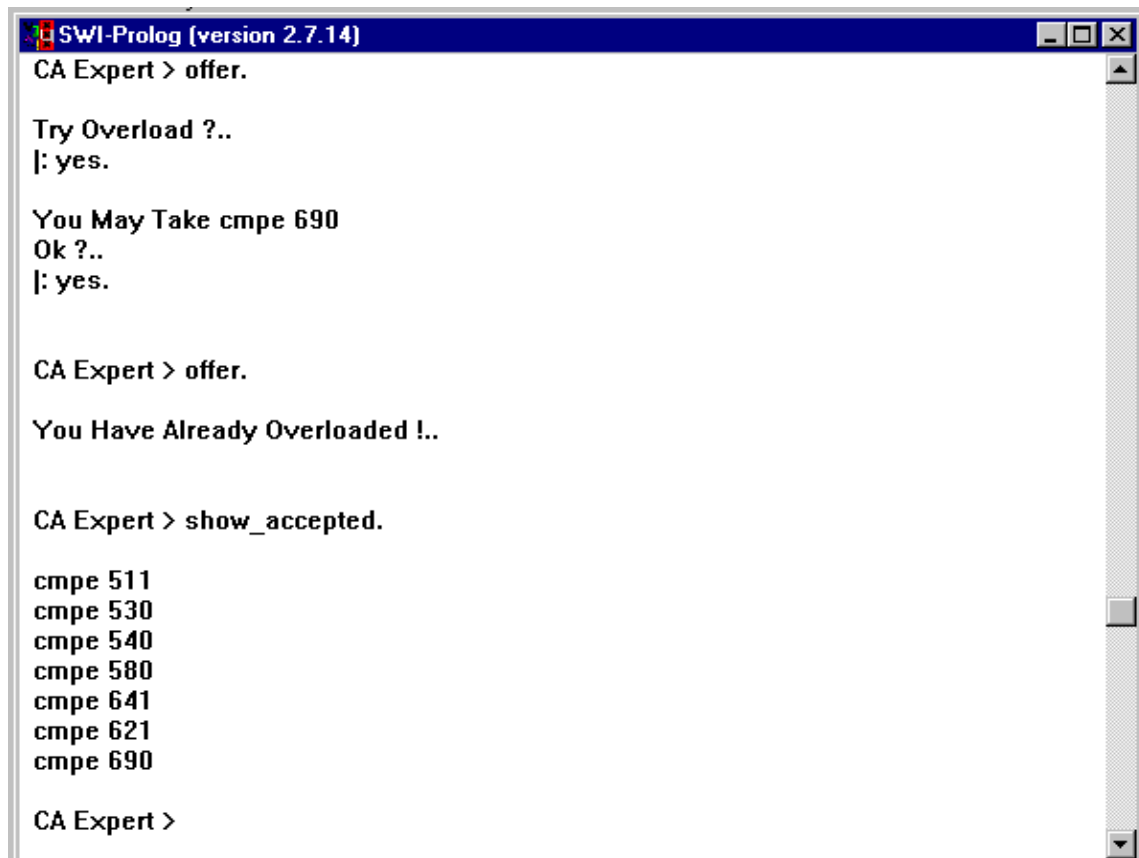


Figure 3

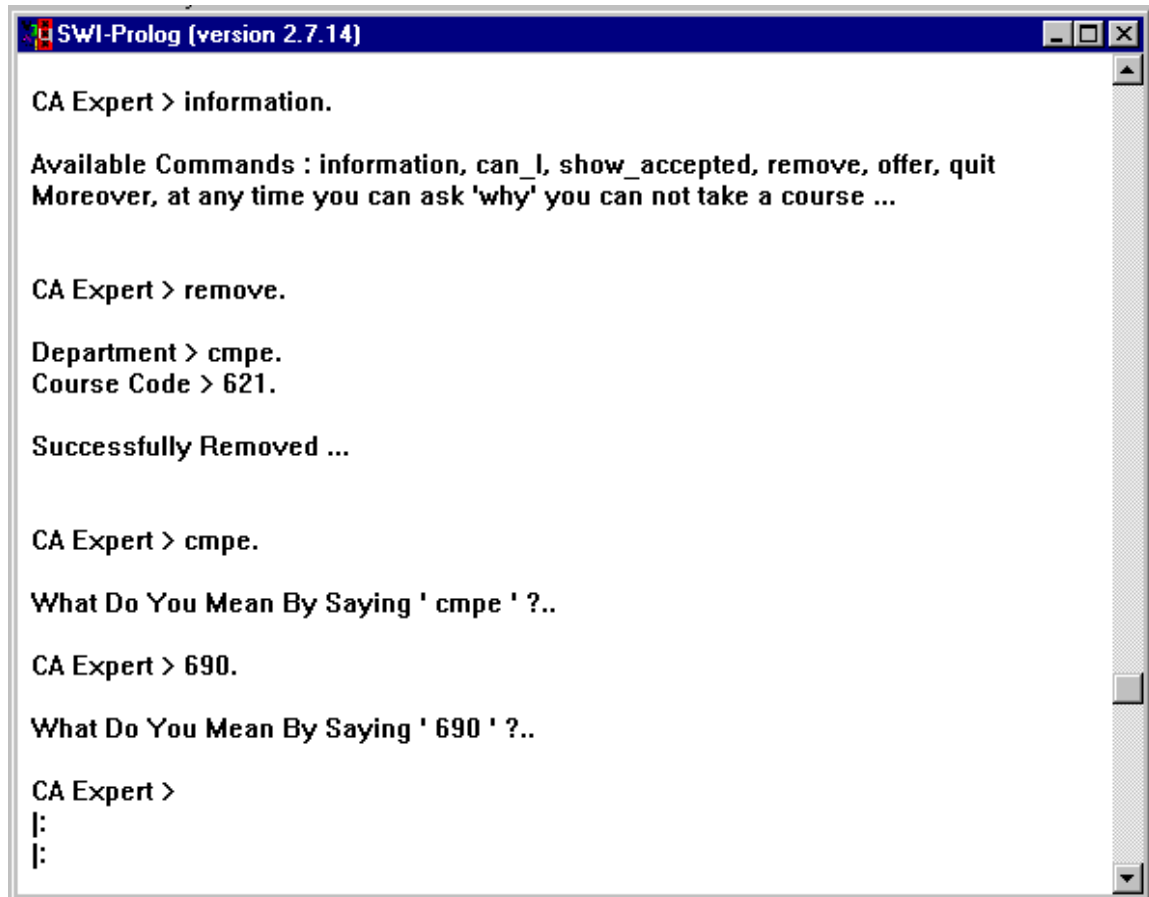
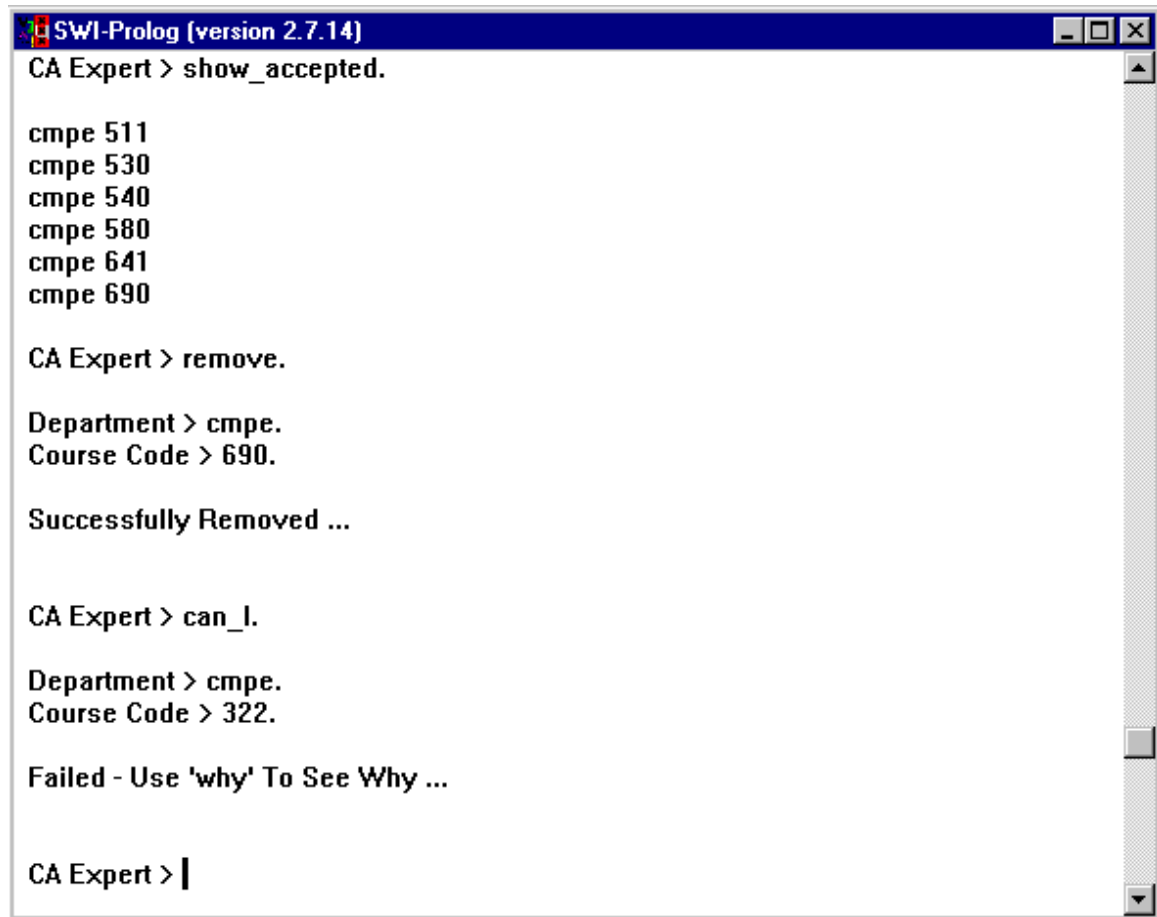


Figure 4



```
SWI-Prolog (version 2.7.14)
CA Expert > show_accepted.

cmpe 511
cmpe 530
cmpe 540
cmpe 580
cmpe 641
cmpe 690

CA Expert > remove.

Department > cmpe.
Course Code > 690.

Successfully Removed ...

CA Expert > can_l.

Department > cmpe.
Course Code > 322.

Failed - Use 'why' To See Why ...

CA Expert > |
```

Figure 5

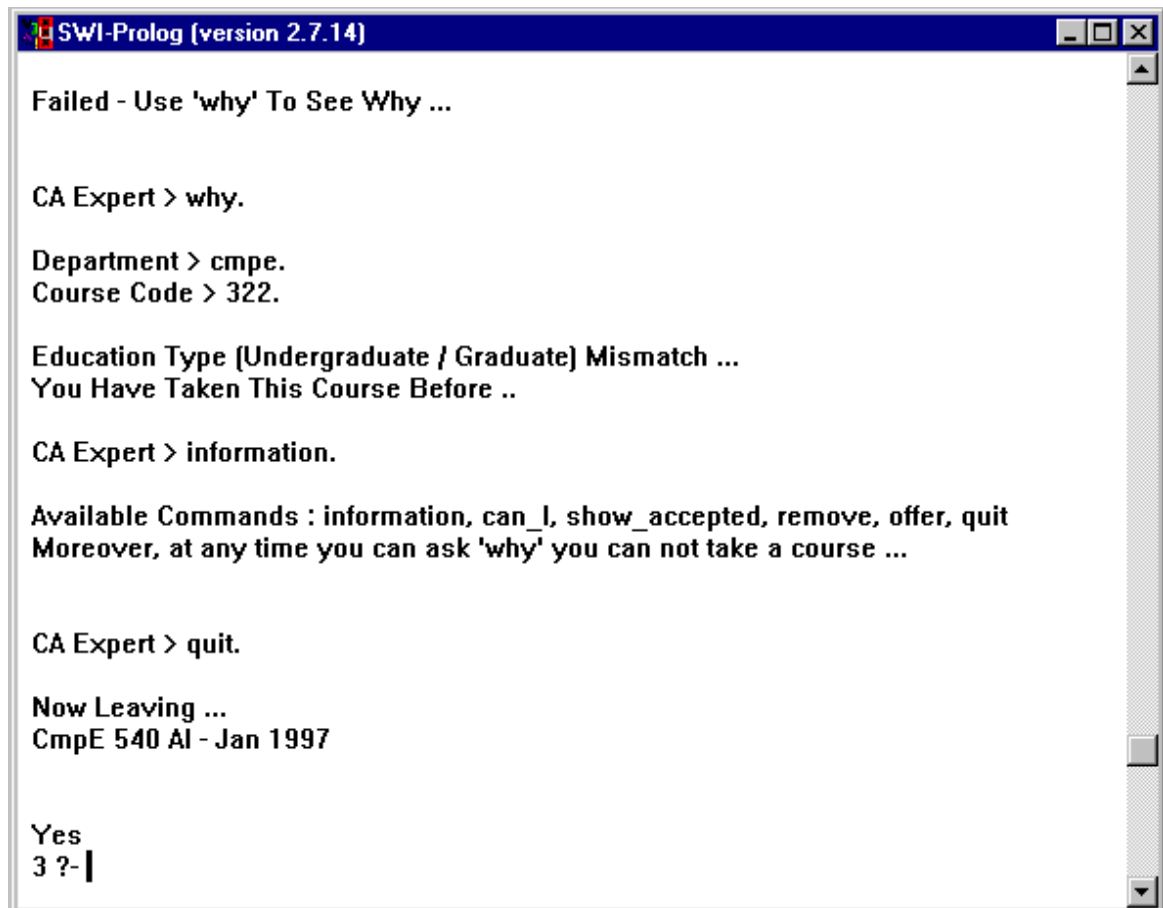


Figure 6

## Conclusion

The CAE system is not a very complicated system. But, it is not a toy system either, as it has various crucial points to consider. There are many details and enough complexity to make it a good project. Moreover, CAE system seems adequate to find a chance to try itself in its real domain, not with a simulated environment like or knowledge base. There is also one more point to mention. When programmed in Prolog, many of the programming details has been eliminated and we get a clear code. This is due to the fact that the problem is suitable to Prolog itself and expert system techniques of AI.

## Appendix - PROLOG Source Codes

```

data.pl -----
/* Top Of Database File */

below_ID(499). /* undergraduate course codes */
over_ID(400). /* graduate course codes */
overloading_gpa(3). /* minimum GPA for overloading */
overloading_semester(2). /* minimum semester for overloading */
not_overloaded(6). /* temporary constant */
overloaded(7). /* temporary constant */
lowest_grade(f). /* determines the lowest grade */
acceptable_grade(dc). /* determines the acceptable grades */
acceptable_grade(dd).
min_normal_GPA(2). /* minimum GPA for taking new courses */
max_hss(2). /* maximum number of HSS courses */
max_cc(6). /* maximum number of CC courses */
max_outdoor_cc(3). /* maximum number of other department CC courses */

/* some positive responses */

yes(yes).
yes(yeah).
yes(yup).

/* message constants */

text(1,'Welcome To The Course Advisor Expert Written By Toygar Under SWI-
Prolog V2.7.14').
text(2,' -> To Start Give A '' go '').
text(3,'Student Number > ').
text(4,'CA Expert > ').
text(5,'Available Commands : information, can_I, show_accepted, remove,
offer, quit').
text(6,'Moreover, at any time you can ask ''why'' you can not take a course
...').
text(7,'Now Leaving ...').
text(8,'CmpE 540 AI - Jan 1997').
text(9,'Department > ').
text(10,'Course Code > ').
text(11,'Successfully Added ...').
text(12,'Failed - Use ''why'' To See Why ...').
text(13,'Successfully Removed ...').
text(14,'Failed - Not Present In Your Course Set ...').
text(15,'No Problems Associated With Your Taking This Course !..').
text(16,'Already Present In Your Course Set ...').
text(17,'Course Type Mismatch ...').
text(18,'Time Collision Occured ...').
text(19,'Education Type (Undergraduate / Graduate) Mismatch ...').
text(20,'Prerequisties Not Completed ...').
text(21,'Corequisties Not Completed ...').
text(22,'You Have Taken This Course Before ..').
text(23,'What Do You Mean By Saying '' '').
text(24,'You May Take ').
text(25,'Ok ?..').
text(26,'You Have Already Overloaded !..').
text(27,'Try Overload ?..').
text(28,'Failed ...').
text(29,'Nothing Taken Yet !..').
text(30,'No Such Course !..').

/* current number of taken hss and cc courses */
hss_count(9156710,2).

```

```

hss_count(9200001,2).
hss_count(9400001,2).
cc_count(9156710,0).
cc_count(9200001,0).
cc_count(9400001,0).
cc_outdoor_count(9156710,0).
cc_outdoor_count(9200001,0).
cc_outdoor_count(9400001,0).

/* student backgrounds */

background(9156710,math,151,aa).
background(9156710,math,152,ba).
background(9156710,math,251,dc).
background(9156710,math,252,aa).
background(9156710,math,310,f).
background(9156710,psy,101,aa).
background(9156710,phil,131,aa).
background(9156710,phys,101,aa).
background(9156710,phys,102,dd).
background(9156710,phys,201,aa).
background(9156710,chem,101,bb).
background(9156710,chem,102,bb).
background(9156710,cmpe,150,bb).
background(9156710,cmpe,220,bb).
background(9156710,cmpe,223,aa).
background(9156710,cmpe,322,aa).
background(9156710,cmpe,344,aa).
background(9156710,cmpe,410,aa).
background(9156710,ie,310,bb).
background(9156710,ie,306,dd).
background(9156710,cmpe,240,aa).
background(9156710,cmpe,224,bb).
background(9156710,cmpe,321,aa).
background(9156710,cmpe,350,f).
background(9156710,pe,101,aa).
background(9156710,pe,102,aa).
background(9156710,tk,221,cb).
background(9156710,tk,222,cb).
background(9156710,htr,311,cb).
background(9156710,htr,312,aa).
background(9156710,eng,110,aa).
background(9156710,ec,201,bb).
background(9156710,ec,202,cc).
background(9156710,ee,211,bb).
background(9156710,ee,212,cc).
background(9156710,me,210,dc).
background(9200001,math,151,cb).
background(9200001,math,152,dc).
background(9200001,math,251,dd).
background(9200001,math,252,bb).
background(9200001,phys,101,f).
background(9200001,phys,102,cb).
background(9200001,chem,101,cb).
background(9200001,chem,102,ba).
background(9200001,cmpe,150,cc).
background(9200001,pe,101,aa).
background(9200001,pe,102,aa).
background(9200001,tk,221,cb).
background(9200001,tk,222,bb).
background(9200001,eng,110,cb).
background(9200001,ec,202,cb).
background(9200001,ie,201,dc).
background(9200001,ie,202,cc).
background(9200001,ie,256,cb).
background(9200001,soc,101,bb).
background(9200001,ad,102,dc).
background(9400001,math,151,aa).
background(9400001,math,152,aa).
background(9400001,psy,101,cb).
background(9400001,phys,101,bb).
background(9400001,phys,102,bb).
background(9400001,chem,101,bb).
background(9400001,chem,102,ba).
background(9400001,pe,101,aa).

```

```
background(9400001,pe,102,aa).
background(9400001,math,161,ba).
background(9400001,math,162,bb).
background(9400001,pols,101,aa).
```

```
/* courses */
```

```
course(math,151,1).
course(math,152,2).
course(math,155,2).
course(math,156,2).
course(math,161,1).
course(math,162,1).
course(math,242,3).
course(math,251,3).
course(math,252,4).
course(math,254,4).
course(math,291,5).
course(math,292,5).
course(math,310,6).
course(math,316,6).
course(math,320,6).
course(math,325,6).
course(math,331,6).
course(math,332,6).
course(math,341,6).
course(math,342,6).
course(math,364,7).
course(math,431,7).
course(math,471,7).
course(math,475,7).
course(math,497,7).
course(math,498,7).
course(cmpe,150,1).
course(cmpe,220,3).
course(cmpe,223,3).
course(cmpe,224,3).
course(cmpe,240,3).
course(cmpe,321,4).
course(cmpe,322,4).
course(cmpe,344,5).
course(cmpe,350,5).
course(cmpe,410,6).
course(cmpe,420,6).
course(cmpe,425,6).
course(cmpe,444,6).
course(cmpe,446,6).
course(cmpe,450,7).
course(cmpe,460,6).
course(cmpe,475,6).
course(cmpe,476,7).
course(cmpe,491,7).
course(cmpe,492,7).
course(cmpe,495,7).
course(cmpe,496,7).
course(cmpe,511,9).
course(cmpe,530,9).
course(cmpe,540,9).
course(cmpe,580,9).
course(cmpe,641,9).
course(cmpe,621,9).
course(cmpe,690,9).
course(cmpe,695,9).
course(ie,201,2).
course(ie,202,2).
course(ie,255,3).
course(ie,256,3).
course(ie,303,4).
course(ie,306,4).
course(ie,310,4).
course(ie,320,4).
course(ie,341,5).
course(ie,403,6).
course(ie,423,6).
course(ie,441,6).
```

```

course(ie,443,6).
course(ie,450,6).
course(ie,491,7).
course(ie,492,7).
course(ie,493,7).
course(ie,494,7).
course(ie,495,7).
course(psy,101,1).
course(phil,131,1).
course(phys,101,1).
course(phys,102,2).
course(phys,201,3).
course(phys,202,3).
course(chem,101,1).
course(chem,102,2).
course(ad,102,2).
course(ad,104,2).
course(ad,131,2).
course(ad,216,3).
course(ad,341,4).
course(ee,210,3).
course(ee,211,3).
course(ee,212,3).
course(ec,201,3).
course(ec,202,3).
course(ec,323,4).
course(pe,101,1).
course(pe,102,1).
course(tk,221,1).
course(tk,222,1).
course(htr,311,4).
course(htr,312,4).
course(eng,110,1).
course(me,210,2).
course(soc,101,1).
course(pols,101,1).

```

```

/* course schedule */

```

```

schedule(math,151,1,2,1,t,t,th).
schedule(math,152,3,4,3,m,m,w).
schedule(math,251,1,2,1,m,m,w).
schedule(math,252,3,4,3,m,m,w).
schedule(math,310,3,4,3,m,m,w).
schedule(psy,101,4,4,4,m,w,f).
schedule(phil,131,7,8,8,m,m,w).
schedule(phys,101,3,3,3,m,w,f).
schedule(phys,102,1,1,1,m,w,f).
schedule(phys,201,1,2,1,t,t,th).
schedule(chem,101,2,2,2,m,w,f).
schedule(chem,102,2,2,2,m,w,f).
schedule(cmpe,150,5,free,free,m,free,free).
schedule(cmpe,220,1,2,6,t,t,f).
schedule(cmpe,223,5,3,4,t,w,w).
schedule(cmpe,322,1,2,5,w,w,th).
schedule(cmpe,344,1,2,6,m,m,th).
schedule(cmpe,410,1,2,3,f,f,f).
schedule(cmpe,420,4,5,6,m,t,t).
schedule(cmpe,444,2,1,2,m,w,w).
schedule(cmpe,450,4,3,4,t,th,th).
schedule(cmpe,460,3,1,2,m,t,t).
schedule(cmpe,475,5,1,2,w,th,th).
schedule(cmpe,491,7,free,free,f,free,free).
schedule(cmpe,492,8,free,free,f,free,free).
schedule(cmpe,495,1,2,3,t,t,t).
schedule(ie,310,3,4,8,t,t,w).
schedule(ie,306,6,5,6,m,w,w).
schedule(cmpe,240,free,free,free,free,free,free).
schedule(cmpe,224,free,free,free,free,free,free).
schedule(cmpe,321,free,free,free,free,free,free).
schedule(cmpe,350,free,free,free,free,free,free).
schedule(pe,101,7,8,free,f,f,free).
schedule(pe,102,free,free,free,free,free,free).
schedule(tk,221,7,8,free,f,f,free).
schedule(tk,222,free,free,free,free,free,free).
schedule(htr,311,7,8,free,m,m,free).

```

```

schedule(htr,312,free,free,free,free,free,free).
schedule(eng,110,6,7,8,m,m,m).
schedule(ec,201,8,7,8,w,f,f).
schedule(ec,202,free,free,free,free,free,free).
schedule(ee,211,3,4,3,m,m,w).
schedule(ee,212,free,free,free,free,free,free).
schedule(me,210,6,3,4,m,f,f).
schedule(math,341,free,free,free,free,free,free).
schedule(cmpe,425,free,free,free,free,free,free).
schedule(cmpe,446,free,free,free,free,free,free).
schedule(cmpe,476,free,free,free,free,free,free).
schedule(cmpe,496,free,free,free,free,free,free).
schedule(cmpe,511,2,3,4,th,th,th).
schedule(cmpe,530,6,7,8,th,th,th).
schedule(cmpe,540,5,4,5,w,f,f).
schedule(cmpe,580,1,2,3,t,t,t).
schedule(cmpe,641,2,3,4,w,w,w).
schedule(cmpe,621,1,2,3,f,f,f).
schedule(cmpe,690,9,free,free,f,free,free).
schedule(cmpe,695,6,7,8,w,w,w).
schedule(ie,201,3,4,3,t,t,th).
schedule(ie,255,5,1,2,m,w,w).
schedule(ie,303,1,2,1,m,m,f).
schedule(ie,341,3,3,4,t,th,th).
schedule(ie,202,free,free,free,free,free,free).
schedule(ie,256,free,free,free,free,free,free).
schedule(ie,320,free,free,free,free,free,free).
schedule(ie,403,6,4,5,t,th,th).
schedule(ie,423,4,5,free,m,m,free).
schedule(ie,441,6,2,3,m,w,w).
schedule(ie,443,1,2,3,th,f,f).
schedule(ie,450,7,8,free,w,w,free).
schedule(ie,491,9,free,free,f,free,free).
schedule(ie,492,9,free,free,f,free,free).
schedule(ie,493,4,6,7,t,t,th).
schedule(ie,494,6,7,8,f,f,f).
schedule(ie,495,6,7,8,th,th,th).
schedule(math,155,1,2,1,t,t,th).
schedule(math,161,3,4,3,t,t,th).
schedule(math,162,7,8,7,m,m,w).
schedule(math,242,1,2,2,m,w,w).
schedule(math,291,1,2,1,t,t,th).
schedule(math,316,6,7,6,t,t,th).
schedule(math,325,3,4,3,t,t,th).
schedule(math,331,4,5,4,t,t,th).
schedule(math,342,5,6,5,m,m,w).
schedule(math,254,free,free,free,free,free,free).
schedule(math,292,free,free,free,free,free,free).
schedule(math,320,free,free,free,free,free,free).
schedule(math,332,free,free,free,free,free,free).
schedule(math,364,free,free,free,free,free,free).
schedule(math,471,1,2,1,m,m,w).
schedule(math,431,3,4,3,t,t,th).
schedule(math,475,6,7,6,m,m,w).
schedule(math,497,9,free,free,m,free,free).
schedule(math,498,9,free,free,t,free,free).
schedule(math,156,free,free,free,free,free,free).
schedule(phys,202,3,4,3,t,t,th).
schedule(ad,216,free,free,free,free,free,free).
schedule(ee,210,free,free,free,free,free,free).
schedule(soc,101,4,3,4,t,th,th).
schedule(ad,102,6,7,8,t,th,th).
schedule(ad,104,7,7,8,th,f,f).
schedule(ad,131,6,7,8,w,w,w).
schedule(pol,101,7,7,7,m,t,w).
schedule(ad,341,7,8,9,t,t,t).
schedule(ec,323,7,8,9,t,t,t).

```

```

/* known students */

```

```

student(9156710,cmpe,graduate,10,3.55).
student(9200001,ie,undergraduate,5,2,50).
student(9400001,math,undergraduate,3,3.20).
student(9500001,ie,undergraduate,1,0.00).
student(9500002,cmpe,undergraduate,1,0.00).

```

```
student(9528880,cmpe,undergraduate,1,0.00).
```

```
/* prerequisites */
```

```
prerequisite(math,251,math,152).
prerequisite(math,310,math,252).
prerequisite(phys,201,phys,102).
prerequisite(chem,102,chem,101).
prerequisite(cmpe,220,cmpe,150).
prerequisite(cmpe,223,cmpe,150).
prerequisite(cmpe,322,cmpe,224).
prerequisite(cmpe,344,cmpe,240).
prerequisite(cmpe,410,cmpe,220).
prerequisite(cmpe,410,cmpe,224).
prerequisite(cmpe,420,cmpe,322).
prerequisite(cmpe,444,cmpe,240).
prerequisite(cmpe,450,cmpe,321).
prerequisite(cmpe,460,cmpe,220).
prerequisite(cmpe,475,cmpe,322).
prerequisite(ie,310,math,251).
prerequisite(ie,306,ie,201).
prerequisite(cmpe,240,cmpe,150).
prerequisite(cmpe,224,cmpe,223).
prerequisite(cmpe,321,cmpe,224).
prerequisite(cmpe,350,cmpe,220).
prerequisite(math,341,math,252).
prerequisite(cmpe,446,cmpe,344).
prerequisite(ie,201,cmpe,150).
prerequisite(ie,255,math,152).
prerequisite(ie,303,ie,201).
prerequisite(ie,341,ie,201).
prerequisite(ie,202,ie,201).
prerequisite(ie,202,ie,255).
prerequisite(ie,256,ie,255).
prerequisite(ie,320,ie,202).
prerequisite(ie,403,ie,320).
prerequisite(ie,403,ie,341).
prerequisite(ie,423,ie,341).
prerequisite(ie,423,ie,320).
prerequisite(ie,441,ie,341).
prerequisite(ie,441,ie,320).
prerequisite(ie,443,ie,341).
prerequisite(ie,443,ie,320).
prerequisite(ie,450,ie,341).
prerequisite(ie,450,ie,320).
prerequisite(ie,491,ie,341).
prerequisite(ie,492,ie,320).
prerequisite(ie,493,ie,341).
prerequisite(ie,494,ie,341).
prerequisite(ie,495,ie,341).
prerequisite(math,242,math,162).
prerequisite(math,291,math,162).
prerequisite(math,316,math,252).
prerequisite(math,325,math,252).
prerequisite(math,331,math,252).
prerequisite(math,331,math,291).
prerequisite(math,342,math,252).
prerequisite(math,342,math,291).
prerequisite(math,320,math,242).
prerequisite(math,332,math,331).
prerequisite(math,364,math,331).
prerequisite(math,471,math,342).
prerequisite(math,431,math,342).
prerequisite(math,475,math,342).
prerequisite(math,152,math,151).
prerequisite(math,252,math,251).
prerequisite(phys,102,phys,101).
prerequisite(phys,202,phys,201).
```

```
/* corequisties */
```

```
corequist(cmpe,220,math,152).
corequist(ee,212,ee,211).
corequist(cmpe,425,cmpe,420).
```

```

corequist (cmpe,476,cmpe,491).
corequist (cmpe,641,cmpe,530).
corequist (cmpe,641,cmpe,540).
corequist (cmpe,621,cmpe,530).
corequist (cmpe,621,cmpe,580).
corequist (cmpe,690,cmpe,540).
corequist (cmpe,690,cmpe,580).
corequist (cmpe,695,cmpe,530).
corequist (math,162,math,161).
corequist (math,254,math,251).
corequist (math,292,math,252).
corequist (math,292,math,291).

```

```

/* courses allowed by a particular department */

```

```

offered_only(math,151,cmpe).
offered_only(math,151,math).
offered_only(math,151,ie).
offered_only(math,152,cmpe).
offered_only(math,152,math).
offered_only(math,152,ie).
offered_only(math,251,cmpe).
offered_only(math,251,math).
offered_only(math,251,ie).
offered_only(math,252,cmpe).
offered_only(math,252,math).
offered_only(math,252,ie).
offered_only(math,310,cmpe).
offered_only(math,310,math).
offered_only(math,310,ie).
offered_only(phys,101,cmpe).
offered_only(phys,101,ie).
offered_only(phys,102,cmpe).
offered_only(phys,102,ie).
offered_only(phys,201,cmpe).
offered_only(phys,201,ie).
offered_only(chem,101,cmpe).
offered_only(chem,101,ie).
offered_only(chem,102,cmpe).
offered_only(chem,102,ie).
offered_only(cmpe,150,cmpe).
offered_only(cmpe,220,cmpe).
offered_only(cmpe,223,cmpe).
offered_only(cmpe,322,cmpe).
offered_only(cmpe,344,cmpe).
offered_only(cmpe,410,cmpe).
offered_only(cmpe,420,cmpe).
offered_only(cmpe,444,cmpe).
offered_only(cmpe,450,cmpe).
offered_only(cmpe,492,cmpe).
offered_only(ie,310,cmpe).
offered_only(ie,310,ie).
offered_only(ie,306,cmpe).
offered_only(ie,306,ie).
offered_only(cmpe,240,cmpe).
offered_only(cmpe,224,cmpe).
offered_only(cmpe,321,cmpe).
offered_only(cmpe,350,cmpe).
offered_only(pe,101,cmpe).
offered_only(pe,101,ie).
offered_only(pe,101,math).
offered_only(pe,102,cmpe).
offered_only(pe,102,ie).
offered_only(pe,102,math).
offered_only(tk,221,cmpe).
offered_only(tk,221,ie).
offered_only(tk,221,math).
offered_only(tk,222,cmpe).
offered_only(tk,222,ie).
offered_only(tk,222,math).
offered_only(htr,311,ie).
offered_only(htr,311,cmpe).
offered_only(htr,312,ie).
offered_only(htr,312,math).
offered_only(eng,110,cmpe).
offered_only(eng,110,math).

```

```

offered_only(ec,201,cmpe).
offered_only(ec,202,cmpe).
offered_only(ee,211,cmpe).
offered_only(ee,212,cmpe).
offered_only(me,210,cmpe).
offered_only(math,341,cmpe).
offered_only(cmpe,511,cmpe).
offered_only(cmpe,530,cmpe).
offered_only(cmpe,540,cmpe).
offered_only(cmpe,580,cmpe).
offered_only(cmpe,641,cmpe).
offered_only(cmpe,621,cmpe).
offered_only(cmpe,690,cmpe).
offered_only(cmpe,695,cmpe).
offered_only(ie,201,ie).
offered_only(ie,255,ie).
offered_only(ie,303,ie).
offered_only(ie,341,ie).
offered_only(ie,202,ie).
offered_only(ie,256,ie).
offered_only(ie,320,ie).
offered_only(ie,403,ie).
offered_only(ie,423,ie).
offered_only(ie,441,ie).
offered_only(ie,443,ie).
offered_only(ie,450,ie).
offered_only(ie,491,ie).
offered_only(ie,492,ie).
offered_only(math,155,math).
offered_only(math,161,math).
offered_only(math,162,math).
offered_only(math,242,math).
offered_only(math,291,math).
offered_only(math,316,math).
offered_only(math,325,math).
offered_only(math,331,math).
offered_only(math,342,math).
offered_only(math,254,math).
offered_only(math,292,math).
offered_only(math,320,math).
offered_only(math,332,math).
offered_only(math,364,math).
offered_only(math,471,math).
offered_only(math,431,math).
offered_only(math,475,math).
offered_only(math,497,math).
offered_only(math,498,math).
offered_only(math,156,math).
offered_only(phys,202,ie).
offered_only(ee,210,ie).

```

```

/* defined hss courses */

```

```

hss(psy,101,cmpe).
hss(psy,101,ie).
hss(phil,131,cmpe).
hss(phil,131,ie).
hss(pols,101,cmpe).
hss(pols,101,ie).
hss(soc,101,cmpe).
hss(soc,101,ie).
hss(ad,102,cmpe).
hss(ad,102,ie).
hss(ad,104,cmpe).
hss(ad,131,cmpe).

```

```

/* defined cc courses */

```

```

cc(ad,341,cmpe).
cc(ad,341,ie).
cc(ec,323,cmpe).
cc(ec,323,ie).
cc(ad,216,cmpe).
cc(ad,216,ie).
cc(ie,493,ie).

```

```
cc(ie,494,ie).
cc(ie,495,ie).
cc(cmpe,425,cmpe).
cc(cmpe,446,cmpe).
cc(cmpe,476,cmpe).
cc(cmpe,495,cmpe).
cc(cmpe,496,cmpe).
cc(cmpe,460,cmpe).
cc(cmpe,475,cmpe).
cc(cmpe,491,cmpe).
```

```
/* Bottom Of Database File */
```

```
advisor.pl -----
```

```
/* Name      : M.Toygar Karadeniz          */
/* Course    : CmpE 540 AI                 */
/* Subject    : A Rule Based Course Advisor Expert */
```

```
/* consulting message */
:-
```

```
    consult('data'),retractall(offered_already(_,_,_)),
    assert(offered_already(dummy,dummy,dummy)),
    nl,text(1,Text_1),write(Text_1),
    nl,text(2,Text_2),write(Text_2),nl,nl.
```

```
/* go:
   main starting predicate of the program
   it implements a command prompt */
```

```
go :-
    info,text(3,Text_1),nl,write(Text_1),
    read(Student_No),
    (retract(offered_count(Student_No,_));true),
    assert(offered_count(Student_No,0)),
    repeat,text(4,Text_2),nl,write(Text_2),read(X),do(Student_No,X),nl,
    X==quit.
```

```
/* info:
   a little information predicate */
```

```
info :-
    nl,text(5,Text_1),write(Text_1),nl,
    text(6,Text_2),write(Text_2),nl.
```

```
/* do:
   input : constant Student_No
           constant command (quit,information,...)
   does the processing of the user command */
```

```
do(_,quit) :-
    nl,text(7,Text_1),write(Text_1),nl,text(8,Text_2),write(Text_2),nl.
```

```
do(_,information) :-
    info,!.
```

```
do(Student_No,can_I) :-
    check_count(Student_No),!.
```

```
do(Student_No,can_I) :-
    !,nl,text(9,Text_1),write(Text_1),read(Dept),
    text(10,Text_2),write(Text_2),read(ID),
    ((check(Student_No,Dept,ID),
    assert(offered_already(Student_No,Dept,ID)),
    offered_count(Student_No,Count),
    retract(offered_count(Student_No,Count)),
    succ(Count,New_Count),assert(offered_count(Student_No,New_Count)),
    nl,text(11,Text_3),write(Text_3),nl,!);
    (nl,text(12,Text_4),write(Text_4),nl,!)).
```

```

do(Student_No, show_accepted) :-
    check_zero_count(Student_No),!.
do(Student_No, show_accepted) :-
    !,nl,offered_already(Student_No,Course_Dept,Course_ID),
    write(Course_Dept),write(' '),write(Course_ID),nl,fail.

do(Student_No, remove) :-
    check_zero_count(Student_No),!.

do(Student_No, remove) :-
    !,nl,text(9,Text_1),write(Text_1),read(Dept),
    text(10,Text_2),write(Text_2),read(ID),
    ((offered_already(Student_No,Dept,ID),
    retract(offered_already(Student_No,Dept,ID)),
    offered_count(Student_No,Count),
    retract(offered_count(Student_No,Count)),
    plus(Count,-1,New_Count),
    assert(offered_count(Student_No,New_Count))),
    nl,text(13,Text_3),write(Text_3),nl,!);
    (nl,text(14,Text_4),write(Text_4),nl,!)).

do(Student_No, why) :-
    !,nl,text(9,Text_1),write(Text_1),read(Dept),
    text(10,Text_2),write(Text_2),read(ID),
    ((not(course(Dept,ID,_)),nl,text(30,Text_99),write(Text_99));
    (
    ((check(Student_No,Dept,ID),nl,text(15,Text_3),write(Text_3));
    (
    (
    (not(offered_already(Student_No,Dept,ID)));
    (nl,text(16,Text_4),write(Text_4))
    ),
    (
    (student(Student_No,Student_Dept,_,_,_),
    check_course_type(Student_No,Dept,ID,Student_Dept));
    (nl,text(17,Text_5),write(Text_5))
    ),
    (
    (hour_or_day(Student_No,Dept,ID));
    (nl,text(18,Text_6),write(Text_6))
    ),
    (
    (student(Student_No,_,Student_Type,_,_),
    education(ID,Student_Type));
    (nl,text(19,Text_7),write(Text_7))
    ),
    (
    (prerequisites(Student_No,Dept,ID));
    (nl,text(20,Text_8),write(Text_8))
    ),
    (
    (corequisties(Student_No,Dept,ID));
    (nl,text(21,Text_9),write(Text_9))
    ),
    (
    (not_taken_before(Student_No,Dept,ID));
    (nl,text(22,Text_10),write(Text_10))
    )
    )
    )
    ),!.

do(Student_No, offer) :-
    check_count(Student_No),!.
do(Student_No, offer) :-
    !,do_offering(Student_No),!.

do(_,X) :-
    nl,text(23,Text_1),write(Text_1),write(X),write(' ' ?..'),nl,fail.

/* below:
   input : constant X
   below is satisfied when X is less than below_ID value */

```

```

below(X) :-
    below_ID(ID),X<ID.

/* over:
   input : constant X
   over is satisfied when X is greater than over_ID value */
over(X) :-
    over_ID(ID),X>ID.

/* hour_or_day:
   input : constant Student_No
           constant Course_Dept
           constant Course_ID
   hour_or_day succeeds if the given course does not
   collide with another one in the current course set */
hour_or_day(Student_No,Course_Dept,Course_ID) :-
    schedule(Course_Dept,Course_ID,Hour1,Hour2,Hour3,Day1,Day2,Day3),
    forall((offered_already(Student_No,Dept,ID),
            schedule(Dept,ID,H1,H2,H3,D1,D2,D3)),
           ((D1\=Day1;H1\=Hour1),
            (D2\=Day2;H2\=Hour2),
            (D3\=Day3;H3\=Hour3))).

/* education:
   input : constant Course_ID
           constant Student_Type
   education checks whether the student is taking the right
   courses according to his/her degree */
education(Course_ID,Student_Type) :-
    Student_Type = undergraduate,not(over(Course_ID)).

education(Course_ID,Student_Type) :-
    Student_Type = graduate,not(below(Course_ID)).

/* not_taken_before:
   input : constant Student_No
           constant Course_Dept
           constant Course_ID
   not_taken_before check whether that course is taken before
   or not - failed courses are determined as not taken before*/
not_taken_before(Student_No,Course_Dept,Course_ID) :-
    background(Student_No,Course_Dept,Course_ID,Grade),
    lowest_grade(LG),Grade = LG.

not_taken_before(Student_No,Course_Dept,Course_ID) :-
    background(Student_No,Course_Dept,Course_ID,Grade),
    acceptable_grade(AG),Grade = AG.

not_taken_before(Student_No,Course_Dept,Course_ID) :-
    not(background(Student_No,Course_Dept,Course_ID,_)),
    student(Student_No,_,_,GPA),
    min_normal_GPA(NGPA),GPA >= NGPA.

/* prerequisites:
   input : constant Student_No
           constant Course_Dept
           constant Course_ID
   prerequisites succeeds if the prerequisites are all done */
prerequisites(Student_No,Course_Dept,Course_ID) :-
    forall(prerequisite(Course_Dept,Course_ID,Pre_Dept,Pre_ID),
           not(not_taken_before(Student_No,Pre_Dept,Pre_ID))).

/* corequisties:
   input : constant Student_No
           constant Course_Dept
           constant Course_ID
   corequisties succeeds if the corequisties are all done */
corequisties(Student_No,Course_Dept,Course_ID) :-
    forall(corequist(Course_Dept,Course_ID,Pre_Dept,Pre_ID),
           (not(not_taken_before(Student_No,Pre_Dept,Pre_ID));
            offered_already(Student_No,Pre_Dept,Pre_ID))).

```

```

/* check:
    input : constant Student_No
           constant Course_Dept
           constant Course_ID
    check succeeds if a given course is suitable for a given
    student */
check(Student_No,Course_Dept,Course_ID) :-
    not(offered_already(Student_No,Course_Dept,Course_ID)),
    student(Student_No,Student_Dept,Student_Type,_,_),
    check_course_type(Student_No,Course_Dept,Course_ID,Student_Dept),
    hour_or_day(Student_No,Course_Dept,Course_ID),
    education(Course_ID,Student_Type),
    prerequisites(Student_No,Course_Dept,Course_ID),
    corequisties(Student_No,Course_Dept,Course_ID),
    not_taken_before(Student_No,Course_Dept,Course_ID).

/* do_offering:
    input : Student_No
    do_offering does the course offering process */
do_offering(Student_No) :-
    student(Student_No,Student_Dept,_,Student_Semester,_),
    course(Course_Dept,Course_ID,Course_Semester),
    Course_Semester =< Student_Semester,
    check(Student_No,Course_Dept,Course_ID),
    nl,text(24,Text_1),write(Text_1),write(Course_Dept),write(' '),
    write(Course_ID),nl,text(25,Text_2),write(Text_2),nl,read(Answer),
    yes(Answer),!,
    assert(offered_already(Student_No,Course_Dept,Course_ID)),
    offered_count(Student_No,Count),
    retract(offered_count(Student_No,Count)),
    succ(Count,New_Count),assert(offered_count(Student_No,New_Count)),
    (
    ((hss(Course_Dept,Course_ID,Student_Dept),
    hss_count(Student_No,HSSCount),
    retract(hss_count(Student_No,HSSCount)),
    succ(HSSCount,New_HSSCount),
    assert(hss_count(Student_No,New_HSSCount))));
    (cc(Course_Dept,Course_ID,Student_Dept),
    cc_count(Student_No,CCCount),
    retract(cc_count(Student_No,CCCount)),
    succ(CCCount,New_CCCount),
    assert(cc_count(Student_No,New_CCCount)),
    ((Student_Dept \= Course_Dept,
    retract(cc_outdoor_count(Student_No,CCOutCount)),
    succ(CCOutCount,New_CCOutCount),
    assert(cc_outdoor_count(Student_No,New_CCOutCount)));(true)));
    (true)).

/* check_count:
    input : Student_No
    check_count handles the overloading process */
check_count(Student_No) :-
    offered_count(Student_No,Count),overloaded(Count),
    nl,text(26,Text_1),write(Text_1),nl,!.

check_count(Student_No) :-
    offered_count(Student_No,Count),not_overloaded(Count),
    nl,text(27,Text_1),write(Text_1),nl,read(Answer),yes(Answer),
    student(Student_No,_,_,Student_Semester,Student_GPA),
    ((overloading_gpa(GPA),overloading_semester(Semester),
    Student_GPA >= GPA,
    Student_Semester >= Semester,
    do_offering(Student_No),!);
    (nl,text(28,Text_2),write(Text_2),nl,!)).

/* check_zero_count:
    input : Student_No
    check_zero_count succeeds if no course is accepted yet */
check_zero_count(Student_No) :-
    offered_count(Student_No,0),nl,
    text(29,Text_1),write(Text_1),nl.

```

```
/* check_course_type:
   input : constant Student_No
           constant Course_Dept
           constant Course_ID
           constant Student_Dept
   check_course_type succeeds if a given course is consistent
   with the department of the student */
check_course_type(Student_No,Course_Dept,Course_ID,Student_Dept) :-
    hss(Course_Dept,Course_ID,Student_Dept),
    hss_count(Student_No,Count1),
    max_hss(Count2),Count1 < Count2,! .

check_course_type(Student_No,Course_Dept,Course_ID,Student_Dept) :-
    cc(Course_Dept,Course_ID,Student_Dept),
    cc_count(Student_No,Count1),
    max_cc(Count2),Count1 < Count2,
    ((Course_Dept\=Student_Dept,
     cc_outdoor_count(Student_No,Count3),
     max_outdoor_cc(Count4),Count3 < Count4,!);(true,!)).

check_course_type(_,Course_Dept,Course_ID,Student_Dept) :-
    offered_only(Course_Dept,Course_ID,Student_Dept),! .
```