Prolog Lab 3

University of Wales, Cardiff

This exercise is not assessed, but it is in your interest to practise this, because SOON there might be one that is.

It was mentioned in the lectures that Prolog can be used as a database, where relations could be analysed using queries combined with AND / OR operations.
A database containing information about students, lecturers, courses and individual lectures contains relations of the type :
course(CourseNo,Ctitle, Clecturer)
lecture(LectureNo, CourseNo, Ltitle)
attended(StudentName, CourseNo, LectureNo)

The database is as follows :

```
course(122, c_programming, dave_marshall).
course(233, operating_systems, jeremy_smith).
course(325, expert_systems, andrew_jones).
course(422, neural_networks, antonia_jones).
course(256, databases, alex_grey).

lecture(1, 122, introduction).
lecture(2, 122, basic_syntax).
lecture(3, 122, memory_management).
lecture(4, 122, pointers).
lecture(1, 233, introduction).
lecture(2, 233, kernel).
lecture(1, 325, introduction).
lecture(2, 325, trees).
lecture(3, 325, predicate_calculus).
lecture(1, 422, introduction).
lecture(2, 422, backpropagation).
lecture(3, 422, gamma_test).
lecture(1, 256, introduction).
lecture(2, 256, oracle).
lecture(3, 256, processing_queries).

attended(willaim_parsons, 422,1).
attended(william_parsons, 422,2).
attended(william_parsons, 325,1).
attended(william_parsons, 325,2).
attended(william_parsons, 325,3).
attended(james_smith, 122, 1).
attended(james_smith, 122, 2).
attended(james_smith, 122, 3).
attended(james_smith, 122, 4).
attended(james_smith, 233, 2).
attended(helen_randall, 256,1).
attended(helen_randall, 256,2).
```

```
attended(helen_randall, 256,3).
attended(helen_randall, 233,1).
attended(helen_randall, 422,1).
```

Note : The above data is available as a *text* file at http://www.cs.cf.ac.uk/User/O.F.Rana/prolog/exercise3.txt or via the *course web page* at : http://www.cs.cf.ac.uk/User/O.F.Rana/prologpage.html

Write the following Prolog clauses :

1. **taught_by(StudentName, LecturerName)** - student *StudentName* was taught by lecturer *LecturerName*
2. **was_at(StudentName, Coursetitle, Lecturetitle)** - student *StudentName* was at lecture titled *Lecturetitle* on course titled *Coursetitle*
3. **pursued_course(StudentName, CourseNo)** - student named *StudentName* pursued course *CourseNo*
4. Add a simple user interface to the three clauses above. Can you combine the three into one dialog which uses check boxes or radio buttons. Write a clause to incorporate the user interface features.
5. Add another clause which suggests that the student was a poor course attender - i.e. he/she pursued a given course, but there is some lecture within that course he/she did not attend **poor_attender(StudentName)** - student named *StudentName* is a poor attender
6. Change your clauses to be able to add (append) and delete (retract) information dynamically from the database.
7. Amend the program so that each person (students and lecturers alike) is given a unique number, and a **name_of** clause associates the person's name and number

Omer Rana
February 1997.