Knowledge-Based Systems
Exercise 2 (assessed)

Please hand in your solutions to this exercise at the revision lecture on Tuesday 7th May 2002 (This deadline has been slightly extended since we are still slightly behind due to the various lecture rearrangements that we have made and shall have to make). Remember to hand in evidence that these programs work, in addition to the program listings.

This exercise is worth *half* of the total continuous assessment marks available for CM0377.

Question 1

(NB this question is taken, with small modifications, from the 2000/2001 examination paper)

Consider the following relational clausal logic program *P*:

```
father_of(jim, john).
same_family(jim, john).
same_family(X, Y):- same_family(Y, X).
```

and the following clause *C:*

```
same_family(john, jim)
```

(i) Use proof by refutation to prove *C* from *P.*

(ii) What is the *Herbrand Universe* of *P?*

(iii) What is the *Herbrand Base* of *P?*

(iv) Write down all the ground instances of each of the clauses in *P.*

(v) Determine which *Herbrand Interpretations* of *P* are *models* of *P*.

HINT for part (v): there are certain elements of the Herbrand Base of *P* that must *obviously* be assigned to **true** in any model of *P.* Also, there are some *father_of* atoms where whether they are assigned to **true** or not in some interpretation makes no difference to whether the interpretation is a model of *P.* Determine these first before exploring the combinations of truth value assignments for the other elements of the Herbrand Base that, in combination with these obvious assignments, comprise models of *P.*

(vi) Hence show that:

- *C* is a logical consequence of *P,* and
- The clause:

  `same_family(jim, jim)`

  is *not* a logical consequence of *P.*

Question 2

NB the meta-interpreter used in this question is to be made available at:
http://www.cs.cf.ac.uk/User/Andrew.C.Jones/kbs0102/Lect9/interp.pl

(a) Modify the Prolog meta-interpreter given in lecture 9, slide 20, so that if there is no clause matching a given goal *G* the user is asked whether *G* is true. If (s)he replies 'yes' then the proof of *G* should succeed; else it should fail.

Devise a small animal recognition knowledge base containing rules such as:

```
tiger:-stripy, fierce.
albatros:-bird, flies_well.
bird:-has_feathers.
```

and show your meta-interpreter in action with your knowledge base. (NB you have effectively developed a primitive backward-chaining inference engine for an expert system.)

HINT: create another clause similar to the long, fourth clause already in the program, but which deals with the case in which `clause(Goal, Body)` does *not* succeed.

Use `read(Term)` to read a term from the keyboard and `write(Term)` to display a term on the screen; `nl` to commence a new line of output. Note that when typing in a term at the keyboard you will need to terminate it with a full stop.

(b) [Optional, but maximum mark for the entire exercise if you do not try this part is 85%] Extend this program so that if the user replies 'yes' regarding goal *G,* then a clause of the form:

```
positive(G)
```

is asserted; otherwise, a clause of the form:

```
negative(G)
```

is to be asserted. Your meta-interpreter should be modified in order to check these clauses first to see whether the answer to a question is already known. If it is already known then the user should not be asked the question again.

Andrew Jones
16th April 2002