Final Exam
Course: COSC 4P79 Expert Systems
Date: Monday April 16, 2001
Time: 0900-1200
Instructor: Brian Ross

Number of pages: 9
Number of questions: 7
Total marks: 94
students: 4
#hours: 3

NAME (print): _____

STUDENT NUMBER: _____

All questions are to be answered on the examination paper.
Use the backs of pages if necessary.
Please keep written answers brief and to the point.
No aids are permitted, other than a calculator without a memory bank, and a 3" by 5" index card with handwritten notes.
Use or possession of unauthorized materials will automatically result in the award of a zero grade for this examination.
A grade of 40% is required on this exam to pass the course.

Question	Total	Mark
1	16	
2	12	
3	18	
4	12	
5	12	
6	12	
7	12	
Total:	94	

Question 1 [16 marks] Define the following terms as used in the course:

a) Knowledge engineer

b) MYCIN

c) ID3

d) frame

e) blackboard

f) probe

g) meta-interpreter

h) conflict set

Question 2 [12 marks]
(a) Discuss the basic architecture of an expert system. Discuss the function and relevance of all the components. Use a diagram.
(b) Explain the importance of separating the shell from the knowledge base, both physically and conceptually.

Question 3 [18 marks]
(a) Define the terms "retrospective case study" and "observational case study". Compare and contrast these two concepts as they relate to knowledge acquisition.
(b) During the lecture on system design and expert systems, the concept of "design cycle" pertinent to expert system development was discussed. Identify and discuss the essential ideas behind this design cycle process
(c) Identify and discuss the main differences between expert system development and conventional software development.

Question 4 [12 marks]
(a) Define, compare and contrast forward chaining and backward chaining inference schemes.
(b) Discuss reasons why Prolog is a good language for expert system shell development.

Question 5 [12 marks] Consider the following decision table used to classify dinosaurs:

WALKS	EATS	SIZE	FEATURES	SPECIES
bipedal	carnivorous	small	retracting_claw	Utahraptor
quadripedal	herbivorous	huge	none	Diplodocus
bipedal	carnivorous	large	none	Tyrannosaurus
bipedal	herbivorous	large	duck-bill	Hadrosaur
quadripedal	herbivorous	large	horn	Triceratops
bipedal	carnivorous	small	flies	Pteradon

(a) Using the ID3 algorithm, create a decision tree for the table. Show all your work, and draw your final tree.
(b) Convert the tree in (b) into production rules.
(c) Explain why clashing examples are a problem for ID3. Give an example of an example that would clash with the above table.

Question 6 [12 marks]
Consider the following MYCIN rules used to identify fish:

1: IF salt water
AND sharp teeth
THEN shark CF 90

2: IF fresh water
AND sharp teeth
THEN pike CF 80

3: IF salt water
AND no teeth
THEN shark CF 15

4: IF fresh water
AND no teeth
THEN sucker CF 20

Facts from user:

salt water CF 90.	no teeth CF 10.
sharp teeth CF 50.	fresh water CF 40.

(a) Presuming a cutoff d=20, fire all the rules in the order given using MYCIN's uncertainty model. Combine the CF's of multiply determined goals appropriately. Show your work.
(b) Repeat (a), but with d=0. Initialize the working storage beforehand (ie. don't combine with results from (a)).
(c) Explain the effect of raising or lowering cutoff d, with respect to the quality of results obtained in general.

Question 7 [12 marks]

Recall the generic Prolog meta-interpreter discussed in the course:

```
prove(true) :- !.
prove(not(A)) :- !, \+ prove(A).
prove((A, B)) :- !, prove(A), prove(B).
prove(A) :- clause(A, B), prove(B).
```

(a) Convert the meta-interpreter so that (i) it creates a conflict set of clauses which can unify with a given goal; (ii) it selects one such clause from the conflict set at random; and (iii) once this random clause is selected, the remainder of the conflict set is disgarded. In other words, deterministic execution arises with no backtracking behavior. You can assume the use of a pseudo-random number generator random(Low, High, N), where N is a random integer between Low and High inclusive.

(b) Convert the meta-interpreter to use clauses in the reverse order found in the program file, which is opposite to Prolog's natural search strategy. Backtracking should work as usual.

?- assert(Happy Summer CF 100).