CS2LAP: Logic and Prolog 2000/2001	**Handed out: 07/12/00**
Coursework 5	**Due date: 15/12/00**

Important notes. This coursework is compulsory for people currently registered as failing coursework (see course web site). For everyone else it won't be marked. Answers to questions 6 and 7 have to be submitted electronically. Put all programs into a single file called `cw5.pl` located in a directory called `cs2lap` under your home directory. The deadline for submission is **4pm** on Friday the 15/12/00. This is also when your Prolog program will be collected automatically.

Question 1. Show that $A \Rightarrow \neg\neg A$ is a tautology by giving a natural deduction proof. Use *only* the rules from page 5 of the logic lecture notes (the *Proof Methods Summary*).

Marks: 10

Question 2. Define a model satisfying the sentence $\forall x.\exists y.(A(x,y) \wedge \neg A(h(x),h(y)))$.

Hint: Recall that a *model* is a pair $\mathcal{M} = \langle \mathcal{D}, \mathcal{I} \rangle$, where $\mathcal{D}$ is the *domain* (a non-empty set of objects) and the $\mathcal{I}$ is the *interpretation function*. $\mathcal{I}$ is mapping each constant symbol c to some object $\mathcal{I}(c) \in \mathcal{D}$, each n-place function symbol f to some n-ary function $\mathcal{I}(f) : \mathcal{D}^n \to \mathcal{D}$, and each n-ary predicate symbol P to some n-ary relation $\mathcal{I}(P) \subseteq \mathcal{D}^n$. (Note that in the textbook instead of $\mathcal{I}$ three separate mappings π_{cons}, π_{func}, and π_{pred} are used.) To answer a question of this kind you have to define a suitable domain $\mathcal{D}$ and a suitable interpretation $\mathcal{I}$ such that the above sentence is true in the model $\mathcal{M} = \langle \mathcal{D}, \mathcal{I} \rangle$.

Marks: 15

Question 3.

(a) Show that $(\forall x.\neg P(x)) \Rightarrow (\neg\forall x.P(x))$ is a theorem of first-order logic by giving a goal-directed proof.

(b) Show that $(\neg\forall x.P(x)) \Rightarrow (\forall x.\neg P(x))$ isn't a theorem of first-order logic by giving a counter-model.

Marks: 20 (10 each)

Question 4. Translate the following Prolog program (which includes a query) into a set of first-order sentences.

```
mortal( X) :- man( X).
man( socrates).
:- mortal( soctrates).
```

Marks: 10

Question 5. Give Prolog's (first) reply for each of the following queries and briefly explain what is happening.

(a) `?- append( [elephant | X], X, [elephant]).`

(b) `?- Y + 17 = 3 * 4 + 17, \+ (Y > 15).`

(c) `?- Z * 17 = 3 + 4 * 17.`

Hint: To answer a question of this kind you have to say whether Prolog replies with `Yes` or `No`. In the former case you also need to give the instantiation for each of the variables involved (if any).

Marks: 15 (5 each)

Question 6. [**Electronic submission**] Tables in a database can be encoded as Prolog facts. In the following sample program the predicate `student/3` relates student IDs, surnames, and forenames, and the predicate `marks/2` relates student IDs with a list of marks.

```
student( i0666, 'Sadness', 'Suzy').    marks( i0815, [12, 0, 9, 15]).
student( i0815, 'Slough', 'Bob').      marks( i1234, [72, 80, 49, 95]).
student( i4711, 'Smart', 'Felicity').  marks( i4711, [100, 96, 90, 100]).
student( i5050, 'Public', 'John Q.').  marks( i5050, [52, 49, 50, 53]).
```

Write a predicate `table/0` to print out a table of students and their average marks (rounded to the next integer). If marks for a student are not in the database this should be indicated in the table. Example:

```
?- table.
Name                 | Mark
===========================
Suzy Sadness         |  n/a
Bob Slough           |    9
Felicity Smart       |   97
John Q. Public       |   51

No
```

Full marks will be given if your program brings all the information on the screen in a readable form, but if you manage to format the output in a nice way that's even better. Obviously, your program should not only work with the sample database but with any database of this kind.

Hints: You might wish to use the predicate `mean/2` defined in coursework 3. Use backtracking to get all the student entries in turn. Recall that you can force backtracking with the `fail`-predicate.

Marks: 15

Question 7. [**Electronic submission**] Implement a Prolog predicate `nelems/3` which, given a natural number `N` in the first argument and a list `L` in the second argument, will generate all possibilities of choosing `N` elements from `L` in turn. Each solution should be a list (of `N` elements) bound to the variable provided as the third argument of `nelems/3`. No solution should come up twice. Examples:

```
?- nelems( 2, [1, 2, 3, 4], Choices).
Choices = [1, 2] ;
Choices = [1, 3] ;
Choices = [1, 4] ;
Choices = [2, 3] ;
Choices = [2, 4] ;
Choices = [3, 4] ;
No

?- nelems( 5, [dog, zebra, elephant], X).
No
```

Marks: 15