CS2LAP: Logic and Prolog 2000/2001	**Handed out: 02/11/00**
Coursework 3	**Due date: 15/11/00**

Important note. Answers to questions 3, 4 and 5 have to be submitted electronically. Put all programs into a single file called `cw3.pl` located in a directory called `cs2lap` under your home directory. The file will be collected automatically at midnight on the due date. If you have any doubts whether you understand and are able to follow these instructions, make sure you ask well *before* the deadline.

Question 1. Consider a language with the predicates $\mathit{friend}(x, y)$ (meaning x is a friend of y), $\mathit{child}(x)$, $\mathit{animal}(x)$, and $\mathit{dog}(x)$ (meaning x is a child, animal, or dog, respectively), and the constant *Fido*.

(a) Write the following English sentences in first-order predicate logic.
  (i) Every dog is an animal.
  (ii) There is a child such that Fido is his/her friend.
  (iii) If nobody is both a child and a dog then if Fido is a dog he cannot be a child.

(b) Translate the following formulas into English sentences.
  (i) $\forall x.\exists y.\mathit{friend}(x, y)$
  (ii) $\exists x.(\mathit{dog}(x) \land \forall y.(\mathit{friend}(x, y) \Rightarrow \mathit{child}(y)))$

Marks: 25 (5 each)

Question 2. Give goal-directed proofs for the following arguments.

(a) $\neg P \;\vdash\; (Q \lor P) \Rightarrow Q$
(b) $Q,\ (P \Rightarrow Q) \Rightarrow (Q \Rightarrow P) \;\vdash\; P$

Marks: 30 (15 each)

Question 3. **[Electronic submission]** Write a Prolog predicate `mean/2` to compute the arithmetic mean of a given list of numbers. Example:

```
?- mean( [1, 2, 3, 4], X).
X = 2.5
Yes
```

Marks: 15

Question 4. **[Electronic submission]** Define operators in Prolog for the connectives of propositional logic. Call your operators `neg`, `and`, `or`, and `implies`.

Think about what precedences and associativity patterns are appropriate. In particular, your declarations should reflect the precedence hierarchy of the connectives as they are defined in propositional logic. Define all binary logical operators as being left-associative. Your definitions should allow for double negation without parentheses (see examples).

Hint: You can easily test whether your operator declarations work as intended. Recall that Prolog when printing out answers omits all redundant parentheses. That means when you ask Prolog to match a variable with a formula, whose structure you have indicated using parentheses, the redundant ones should disappear in the output. Parentheses that are necessary, however, will be shown. Examples:

```
?- Formula = a implies ((b and c) and d).
Formula = a implies b and c and d
Yes

?- AnotherFormula = (neg (neg a)) or b.
AnotherFormula = neg neg a or b
Yes

?- ThirdFormula = (a or b) and c.
ThirdFormula = (a or b)and c
Yes
```

Marks: 15

Question 5. [**Electronic submission**] Write a Prolog predicate `dnf/1` to test whether a given formula is in disjunctive normal form (using the operators you've defined for the previous question). Examples:

```
?- dnf( (p and q and neg r) or (neg p)).
Yes

?- dnf( p and neg q or neg p and q or p and q).
Yes

?- dnf( r).
Yes

?- dnf( (p or q) and neg a).
No

?- dnf( (p and q and r) or (neg neg q)).
No
```

Hint: Propositional atoms correspond to atoms in Prolog. You can test whether a given term is a valid Prolog atom by using the built-in predicate `atom/1`.

Marks: 15