Week beginning 13.10.85

Department of Artificial Intelligence
University of Edinburgh
Artificial Intelligence 2

PROLOG PRACTICAL 1

To be handed in Tuesday, week 3 (21.10.85)

1. Write the following program:

```
sentence:-
        noun(N1), write(' '), write(N1),
        verb(V1), write(' '), write(V1),
        noun(N2), write(' '), write(N2).

noun(fred).
noun(beer).
noun(doris).
noun(gin).

verb(likes).
verb(drinks).
```

and name it, for example, sent1.

Get into Prolog by typing the command

```
prolog
```

Consult the file sent1 (or whatever you have called it)

```
?- consult(sent1).
```

List it. Predict what the result will be if you type the following goal:

```
?- sentence.
```

fred likes fred/beer

2. Using an editor, modify the program to the following:

```
sentence(X):-
        noun(N1),
        verb(V1),
        noun(N2),
        make([N1,V1,N2],X).

make(Anything,Anything).

noun(fred).
noun(beer).
noun(doris).
noun(gin).

verb(likes).
verb(drinks).
```

What is the outcome of the goal:

```
?- sentence(S).
```


Explain how this solution is achieved. With the same goal, *redo* to find all solutions.

Draw an AND/OR tree to represent the execution of this goal, showing all solutions and their ordering.

3. The last subgoal of the sentence predicate ('make([N1,V1,N2],X)') is actually redundant in this example. Instead of unifying X and [N1,V1,N2] in the make predicate it can be done in the head of the sentence predicate itself:

```
sentence([N1,V1,N2]):-
        noun(N1),
        verb(V1),
        noun(N2).

noun(fred).   cat
noun(beer).
noun(doris).
noun(gin).

verb(likes).
verb(drinks).
```

det(the) det(a) adj(green) adj(red)

Edit and run this version of the program.

4. In these examples, lists of items are printed. Write a higher level predicate called go that uses the sentence predicate and another predicate called prlist. The prlist predicate should write out all items in the list that is unified with the variable S in the solution of the goal sentence(S).

Modify your version of prlist to deal with embedded lists, e.g.

```
prlist([a,b,[c,d],e,[f,[g]],h]).
```

should produce

```
a b c d e f g
```

5. If we had sentences of constructions other than simple 'noun verb noun', e.g. 'det noun verb det adj noun', we could cope if we wrote a new predicate for every construction. Add two such constructions of your own choice.

In general, this is not practical. Think about how else this might be done.

```
sentence([D1,N1,V1,D2,ADJ,N2]):-
        det(D1),
        noun(N1),
        verb(V1),
        det(D2),
        adj(ADJ),
        noun(N2)
```