**Some simple Prolog Examples**

---------------------------

1. Here are some simple clauses.

likes(mary,food).

likes(mary,wine).

likes(john,wine).

likes(john,mary).

The following queries yield the specified answers.

| ?- likes(mary,food).

yes.

| ?- likes(john,wine).

yes.

| ?- likes(john,food).

no.

How do you add the following facts?

1. John likes anything that Mary likes

2. John likes anyone who likes wine

3. John likes anyone who likes themselves

==========================================================================

2. Slightly more complicated family tree.

James I

|

|

+----------------+-----------------+

| |

Charles I Elizabeth

| |

| |

+----------+------------+ |

| | | |

Catherine Charles II James II Sophia

|

|

|

George I

Here are the resultant clauses:

-------------------------------

male(james1).

male(charles1).

male(charles2).

male(james2).

male(george1).

female(catherine).

female(elizabeth).

female(sophia).

parent(charles1, james1).

parent(elizabeth, james1).

parent(charles2, charles1).

parent(catherine, charles1).

parent(james2, charles1).

parent(sophia, elizabeth).

parent(george1, sophia).

Here is how you would formulate the following queries:

Was George I the parent of Charles I?

Query: parent(charles1, george1).

Who was Charles I's parent?

Query: parent(charles1,X).

Who were the children of Charles I?

Query: parent(X,charles1).

Now try expressing the following rules:

M is the mother of X if she is a parent of X and is female

F is the father of X if he is a parent of X and is male

X is a sibling of Y if they both have the same parent.

Furthermore add rules defining:

"sister", "brother",

"aunt", "uncle",

"grandparent", "cousin"

====================================================================

3. Recursion: Towers of Hanoi

The 3-disk setup is like this:

| | |

xxx | |

xxxxx | |

xxxxxxx | |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Here's a sample:

% move(N,X,Y,Z) - move N disks from peg X to peg Y, with peg Z being the

% auxilliary peg

%

% Strategy:

% Base Case: One disc - To transfer a stack consisting of 1 disc from

% peg X to peg Y, simply move that disc from X to Y

% Recursive Case: To transfer n discs from X to Y, do the following:

Transfer the first n-1 discs to some other peg X

Move the last disc on X to Y

Transfer the n-1 discs from X to peg Y

move(1,X,Y,\_) :-

write('Move top disk from '),

write(X),

write(' to '),

write(Y),

nl.

move(N,X,Y,Z) :-

N>1,

M is N-1,

move(M,X,Z,Y),

move(1,X,Y,\_),

move(M,Z,Y,X).

- note the use of "anonymous" variables \_

Here is what happens when Prolog solves the case N=3.

?- move(3,left,right,center).

Move top disk from left to right

Move top disk from left to center

Move top disk from right to center

Move top disk from left to right

Move top disk from center to left

Move top disk from center to right

Move top disk from left to right

yes

====================================================================

4. An example using lists:

(a) length of a list

size([],0).

size([H|T],N) :- size(T,N1), N is N1+1.

% or size([\_|T],N) :- size(T,N1), N is N1+1.

| ?- size([1,2,3,4],N).

N = 4

yes

| ?- size([bill,ted,ming,pascal,nat,ron],N).

N = 6

yes

| ?- size([a, [b, c, d], e, [f | g], h], N).

N = 5

yes

(b) summing elements of a list of numbers

sumlist([],0).

ssumlist([H|T],N) :- sumlist(T,N1), N is N1+H.

(c) list membership

member(X,[X|\_]).

member(X,[\_|T]) :- member(X,T).

(d) reversing a list

reverse(List, Reversed) :-

reverse(List, [], Reversed).

reverse([], Reversed, Reversed).

reverse([Head|Tail], SoFar, Reversed) :-

reverse(Tail, [Head|SoFar], Reversed).

| ?- myreverse([a,b,c,d],X).

X = [d,c,b,a]; <- note semicolon (more solns?)

no

| ?- myreverse([a,b,c,d],[d,b,c,a]).

no

| ?- myreverse([a,b,c,d],[d,c,b,a]).

yes

- note difference between reverse/2 and reverse/3

- reverse/3 probably should be called reverseHelper or

something else for clarity

=========================================================