Tutorial 4 Solutions

COMP 2411, Session 1, 2004

Q1:

```
:- writeln('\nThe goal "solve" solves the band puzzle.\n'),
   writeln('Three musicians of a multinational band take turns'),
   writeln('playing solo in a piece of music; each plays only once.'),
   writeln('The pianist plays first.'),
   writeln('John plays saxophone and plays before the Australian.'),
   writeln('Mark comes from the United States and plays before the violinist.'),
   writeln('One soloist comes from Japan and one is Sam.'),
   writeln('Who comes from which country, plays what instrument,'),
   writeln('and in what order?\n').

solve :-
        discover([[FirstSoloist, FirstCountry, piano],
                  [SecondSoloist, SecondCountry, SecondIntrument],
                  [ThirdSoloist, ThirdCountry, ThirdIntrument]]),
        writef('\nThe first player is %w; he comes from %w and plays the piano.\n',
               [FirstSoloist, FirstCountry]),
        writef('The second player is %w; he comes from %w and plays the %w.\n',
               [SecondSoloist, SecondCountry, SecondIntrument]),
        writef('The third player is %w; he comes from %w and plays the %w.\n',
               [ThirdSoloist, ThirdCountry, ThirdIntrument]).

discover(Solution) :-
        member(['John', _, saxophone], Solution),
        plays_before(['John', _, _], [_, 'Australia', _], Solution),
        member(['Mark', 'United Sates', _], Solution),
        plays_before(['Mark', _, _], [_, _, violin], Solution),
        member([_, 'Japan', _], Solution),
        member(['Sam', _, _], Solution).


plays_before(First, Second, [First, Second, _]).
plays_before(First, Third, [First, _, Third]).
plays_before(Second, Third, [_, Second, Third]).
```

Q2:

```
selection_sort([], []).
selection_sort(List, [H| T]) :-
        least(H, List, Rest),
        selection_sort(Rest, T).

least(X, List, Rest) :-
        select(X, List, Rest),
        smaller(X, Rest).

smaller(_, []).
smaller(X, [H| T]) :-
        X =< H,
        smaller(X, T).
```

Q3:

```
gcd(X, X, X).
gcd(X, Y, D) :-
        X < Y,
        Y1 is Y - X,
        gcd(X, Y1, D).
gcd(X, Y, D) :-
        Y < X,
        gcd(Y, X, D).
```

Q3: `L = 1+ (1+ (1+ (1+0)))`

Q4:

```
:- writeln('\nThe goal "hanoi(N)" solves the towers of Hanoi problem.\n'),
   writeln('N disks, with N>=0, have to be moved from peg a to peg b,'),
   writeln('using auxiliary peg c.\n').

hanoi(0) :-
        write('\nDo nothing').
hanoi(1) :-
        write('\nMove the disk from peg a to peg b').
hanoi(N) :-
        N > 1,
        solve_hanoi(N, a, b, c, Moves),
        write_solution(Moves).

solve_hanoi(1, A, B, _, [(A,B)]).
solve_hanoi(N, A, B, C, Moves) :-
        N > 1,
        M is N - 1,
        solve_hanoi(M, A, C, B, Moves1),
        solve_hanoi(M, C, B, A, Moves2),
        append(Moves1, [(A, B)| Moves2], Moves).

write_solution([]).
write_solution([(A, B)| Moves]) :-
        writef('\nMove one disk from peg %w to peg %w',[A, B]),
        write_solution(Moves).
```