Eric Smrkovsky
10/25/19
CSci 117
Lab 10

Part One

```
<Sentence>      ::=   <NounPhrase> <VerbPhrase>
<NounPhrase>    ::=   <Determiner> <Noun> <RelClause> | <Name>
<VerbPhrase>    ::=   <TransVerb><NounPhrase> | <NounPhrase><TransVerb> | <IntransVerb>
<RelClause>     ::=   who <VerbPhrase> | ε

<Determiner>    ::=   every | a
<Noun>          ::=   man | woman | child
<Name>          ::=   john | mary | eric | aiden
<TransVerb>     ::=   loves | knows
<IntransVerb>   ::=   lives | runs
```

```
det(S,P1,P2,[every|X],X,all(S,imply(P1,P2))).
det(S,P1,P2,[a|X],X,exists(S,and(P1,P2))).

noun(N,[man|X],X,man(N)).
noun(N,[woman|X],X,woman(N)).
noun(N,[child|X],X,child(N)).

name([john|X],X,john).
name([mary|X],X,mary).
name([eric|X],X,eric).
name([aiden|X],X,aiden).

trVerb(S,O,[loves|X],X,loves(S,O)).
trVerb(S,O,[knows|X],X,knows(S,O)).

intrVerb(S,[lives|X],X,lives(S)).
intrVerb(S,[runs|X],X,runs(S)).

sen(X0,X,P) :- nounPh(N,P1,X0,X1,P),
               verbPh(N,X1,X,P1).

nounPh(N,P1,X0,X,P) :- det(N,P2,P1,X0,X1,P),
                       noun(N,X1,X2,P3),
                       relCl(N,P3,X2,X,P2).
nounPh(N,P1,X0,X,P1) :- (name(X0,X,N)).

verbPh(S,X0,X,NP) :- trVerb(S,O,X0,X1,P1),
                     nounPh(O,P1,X1,X,NP).
verbPh(S,X0,X,TV) :- nounPh(O,P,X0,X1,TV),
                     trVerb(O,S,X1,X,P).
verbPh(S,X0,X,IV) :- intrVerb(S,X0,X,IV).
```

Eric Smrkovsky
10/25/19
CSci 117
Lab 10

```
relCl(S,P1,[who|X1],X,and(P1,P2)) :- verbPh(S,X1,X,P2).
relCl(_,P1,X,X,P1).
```

Below is an example of the TransVerb swap.

```
| ?- sen([every,child,who,knows,aiden,runs],[],X).
X = all(A,imply(and(child(A),knows(A,aiden)),runs(A)))

| ?- sen([every,child,who,aiden,knows,runs],[],X).
X = all(A,imply(and(child(A),knows(aiden,A)),runs(A)))
```

Below are four more examples of at least length 10.

```
| ?- sen([every,man,who,john,loves,loves,a,woman,who,runs],[],X).
X = all(A,imply(and(man(A),loves(john,A)),exists(B,and(and(woman(B),runs(B)),loves(A,B)))))

| ?- sen([every,woman,who,eric,knows,loves,a,woman,who,lives],[],X).
X =
all(A,imply(and(woman(A),knows(eric,A)),exists(B,and(and(woman(B),lives(B)),loves(A,B)))))

| ?- sen([a,man,who,runs,every,child,who,loves,aiden,knows],[],X).
X = exists(A,and(and(man(A),runs(A)),all(B,imply(and(child(B),loves(B,aiden)),knows(B,A)))))

| ?- sen([a,woman,who,lives,every,man,who,loves,a,child,knows],[],X).
X =
exists(A,and(and(woman(A),lives(A)),all(B,imply(and(man(B),exists(C,and(child(C),loves(B,C)
))),knows(B,A))))) ?
```

Eric Smrkovsky
10/25/19
CSci 117
Lab 10

Part 2.1

```
% Render the ship term as a nice table.
:- use_rendering(table,
        [header(s('Ship', 'Leaves at', 'Carries', 'Chimney', 'Goes to'))]).

goes_PortSaid(Goes) :-
        ships(S),
        member(s(Goes,_,_,_,portSaid), S).

carries_tea(Carries) :-
        ships(S),
        member(s(Carries,_,tea,_,_), S).

ships(S) :-
  length(S,5),
  member(s(greek,6,coffee,_,_),S),                    % 1
  S = [_,_,s(_,_,_,black,_),_,_],                     % 2
  member(s(english,9,_,_,_),S),                       % 3
  left(s(french,_,_,blue,_),s(_,_,coffee,_,_),S),     % 4
  left(s(_,_,cocoa,_,_),s(_,_,_,_,marseille),S),      % 5
  member(s(brazilian,_,_,_,manila),S),                % 6
  next(s(_,_,rice,_,_),s(_,_,_,green,_),S),           % 7
  member(s(_,5,_,_,genoa),S),                         % 8
  left(s(_,_,_,_,marseille),s(spanish,7,_,_,_),S),    % 9
  member(s(_,_,_,red,hamburg),S),                     % 10
  next(s(_,7,_,_,_),s(_,_,_,white,_),S),              % 11
  border(s(_,_,corn,_,_),S),                          % 12
  member(s(_,8,_,black,_),S),                         % 13
  next(s(_,_,corn,_,_),s(_,_,rice,_,_),S),            % 14
  member(s(_,6,_,_,hamburg),S),                       % 15
  member(s(_,_,_,_,portSaid),S),                      % 16 (given)
  member(s(_,_,tea,_,_),S).                           % 17 (given)

next(A, B, Ls) :- append(_, [A,B|_], Ls).     % could be as shown
next(A, B, Ls) :- append(_, [B,A|_], Ls).     % or could be opposite
left(A, B, Ls) :- append(_, [A,B|_], Ls).     % A to the left of B || B right of A
border(A, [A|_]).                             % could be on left border
border(A, Ls)  :- append(_, [A], Ls).         % could be on right border
```

Which ship goes to Port Said?
The Spanish one.

Which ship carries tea?
The French one.

Eric Smrkovsky
10/25/19
CSci 117
Lab 10

Part2.2 with extra credit.
Changes to program are in bold.

```
% N is size of rows/columns, T is tour taken (as list of positions m(P1,P2))
knightTour(N,M,T) :- N2 is N*M,  % N2 is the number of positions on the board (N*M)
  kT(N,M,N2,[m(0,0)],T).         % [m(0,0)] is starting position of the knight

% kT(N,M,N2,[m(P1,P2)|Pt],T)
% N is the row size
% M is the column size
% N2 is the number of positions on the board
% [m(P1,P2)|Pt] is the accumulator for the tour
%   m(P1,P2) is the current position of the knight
%   Pt (partial tour) is the list of previous positions of the knight
% T is the full tour

% Finished: Length of tour is equal to size of board and at a spot that is one valid
% move away from m(0,0).
% return accumulator as tour adding the final move back to m(0,0).
kT(_,_,N2,[H|T],[m(0,0),H|T]) :- (H==m(1,2);H==m(2,1)),length([H|T],N2).

kT(N,M,N2,[m(P1,P2)|Pt],T) :-
    moves(m(P1,P2),m(D1,D2)), % get next position from current position
    D1>=0,D2>=0,D1<N,D2<M,    % verify next position is within board dimensions
    \+ member(m(D1,D2),Pt),   % next position has not already been covered in tour
    kT(N,M,N2,[m(D1,D2),m(P1,P2)|Pt],T). % append next position to front of accumulator

% 8 possible moves for a knight
% P1,P2 is knight's position, D1,D2 is knight's destination after one move
% Iterated list solution
moves(m(X,Y), m(U,V)) :-
  member(m(A,B), [m(1,2),m(1,-2),m(-1,2),m(-1,-2),m(2,1),m(2,-1),m(-2,1),m(-2,-1)]),
  U is X + A,
  V is Y + B.

% "Number Hacking" solution
offsets(N1,N2) :-
    member(N,[1,2]),member(B1,[1,-1]),member(B2,[1,-1]),
    N1 is N*B1, N2 is (3-N)*B2.
% moves(m(P1,P2),m(D1,D2)) :-
%    offsets(A,B), D1 is P1+A, D2 is P2+B.
```