Homework: Logic Programming

Learning Objectives:

1. Problem solving using logic programming paradigm
2. Prolog programming

Instructions:

- Total points 48 pt
- Early deadline: Dec 4 (Wed) 2019 at 11:59 PM; Regular deadline: Dec 6 (Fri) 2019 at 11:59 PM (or till TAs start grading the homework)
- Download and install Swi-prolog `http://www.swi-prolog.org/`
- Please zip .pl files and output files for all the solutions and submit it to Canvas.

Questions:

1. (3 pt) Understand the following Prolog program:

   Given: $mystery([\,], L2, L2)$.
   $mystery([H|Tail], L2, [R|RTail]) :-$
   $H = R$,
   $mystery(Tail, L2, RTail)$.

   What would Z be in $mystery([1,4,6],[3,6],Z)$.

   Sol: Z = [1, 4, 6, 3, 6].

2. (10 pt) Prolog programming:

   - (4 pt) Compute the nth number in Fibonacci Sequence.
   - (6 pt) Reverses a list and any nested lists. For example: [1,2,[2,4],5] = [5,[4,2],2,1].

   Sol:

   - Fibonacci

```
fibonaaci(0, 1).
fibonaaci(1, 1).
fibonaaci(N, Result):-
N1 is N - 1,
N2 is N - 2,
fibonaaci(N1, Result1),
fibonaaci(N2, Result2),
Result is Result1 + Result2.
```

- Reverse list

```
accRev([],A,A).
accRev([ [ H1 | T1 ] | T], A, R) :-
reverse([H1 | T1], R1),
accRev(T, [R1 | A], R).
accRev([H | T], A, R) :- accRev(T, [H | A], R).
reverse(L, R):-  accRev(L, [], R).
```

3. (15 pt) Write a Prolog program for parsing:

   (a) (8 pt) Consider the grammar we worked in HW1 below. Write a Prolog program that parses strings using this grammar. Your program can be used to check if a given sentence can be generated by the grammar. An example interpreter session is provided below.

   Grammar:

   - terminals: $x, y, z, >, <, 0, 1, +, -, =$, if, then, else
   - non-terminals: S, F, B, T, E, N
   - start symbol: S
   - production rules:
     $S \rightarrow F|T\ N\ T$
     $F \rightarrow$if B then S|if B then S else S
     $B \rightarrow T\ E\ T$
     $T \rightarrow x|y|z|1|0$
     $E \rightarrow > \mid <$
     $N \rightarrow +|-|=$

   Example:

```
| ?- sentence([if, x, > , 0, then, [x, =, 1]]).
| true.
| ?- sentence([if, x, > , 0, then, [x, =, 1], else, [x, =, 0]]).
| true.
```

   (b) (5 pt) Write the query to generate all possible sentences that can be derived from the grammar. Show the screenshot of 3 sentences.

   (c) (2 pt) Does the order of the sub-goals in your rules make a difference?

Sol:

(a)

```
sentence([]).
sentence([A,B,C]):- tntexpr(A,B,C).
sentence([A,B,C,D,E,F]):-
```

```
4  ifterm(A), bcond(B,C,D), thenterm(E), sentence(F).
5  sentence([A,B,C,D,E,F,G,H]):-
6  ifterm(A), bcond(B,C,D), thenterm(E), sentence(F), elseterm(G), sentence(H).
7
8  tntexpr(A,B,C):- t(A), n(B), t(C).
9  bcond(A,B,C):- t(A), e(B), t(C).
10
11 t(x).
12 t(y).
13 t(z).
14 t(1).
15 t(0).
16 n(+).
17 n(-).
18 n(=).
19 e(>).
20 e(<).
21 ifterm(if).
22 thenterm(then).
23 elseterm(else).
```

(b) Yes. for this particular implementation type in `sentence([A]).` into prolog continue to ask for more solutions until no more backtracking can be done. It is not optimized so that there are no repeat answers.

(c) No.

4. (20 pt) Write a prolog program to solve a constraint satisfaction puzzle: There are five houses, each of a different color and inhabited by men of different nationalities, with different pets, drinks, and cigarettes. Given the facts to the following, who drinks water and who owns the zebra?

   - the englishman lives in the red house
   - the spaniard owns the dog.
   - coffee is drunk in the green house
   - the ukrainian drinks tea.
   - the green house is immediately to the right of the ivory house.
   - the old gold smoker owns snails.
   - kools are being smoked in the yellow house.
   - milk is drunk in the middle house.
   - the norwegian lives in the first house on the left.
   - the camel smoker lives next to the fox owner.
   - kools are smoked in the house next to the house where the horse is kept.
   - the lucky strike smoker drinks orange juice.
   - the japanese smokes parlaiments.
   - the norwegian lives next to the blue house.

   Sol:

   Some description of the code:

- `Hs`: a list consists of houses, length is 5
- `h`: house. The signature is `h(Nationality, Pet, Smoke, Drink, Color)`
- To enforce a rule (such as english man live in red house), the member library function is used. E.g. `member(h(englishman,_,_,_,red), Hs)` enforces the house containing "englishman live in red house" is inside the list Hs.
- *right of* is defined as `rightof(A, B, Ls) :- append(_, [A,B|_], Ls).`, meaning in the list, B is to the right of A. *next to* is similar.
- To enforce the specific location, e.g. "milk is drunk in the middle house.", we use `Hs=[_,_,h(_,_,_,milk,_),_,_]`, meaning the list is consist of 5 elements, but the middle one is the house with milk.

```
houses(Hs) :-
% each house in the list Hs of houses is represented as:
%      h(Nationality, Pet, Smoke, Drink, Color)
length(Hs, 5),
%% 1. the englishman lives in the red house
member(h(englishman,_,_,_,red), Hs),
%% 2. the spaniard owns the dog.
member(h(spaniard,dog,_,_,_), Hs),
%% 3. coffee is drunk in the green house
member(h(_,_,_,coffee,green), Hs),
%% 4. the ukrainian drinks tea.
member(h(ukrainian,_,_,tea,_), Hs),
%% 5. the green house is immediately to the right of the ivory house.
rightof(h(_,_,_,_,ivory),h(_,_,_,_,green), Hs),
%% 6. the old gold smoker owns snails.
member(h(_,snails,oldgold,_,_), Hs),
%% 7. kools are being smoked in the yellow house.
member(h(_,_,kools,_,yellow), Hs),
%% 8. milk is drunk in the middle house.
Hs = [_,_,h(_,_,_,milk,_),_,_],
%% 9. the norwegian lives in the first house on the left.
Hs = [h(norwegian,_,_,_,_)|_],
%% 10. the camel smoker lives next to the fox owner.
nextto(h(_,_,camel,_,_), h(_,fox,_,_,_), Hs),
%% 11. kools are smoked in the house next to the house where the horse is kept.
nextto(h(_,_,kools,_,_), h(_,horse,_,_,_), Hs),
%% 12. the lucky strike smoker drinks orange juice.
member(h(_,_,luckystrike,orangejuice,_), Hs),
%% 13. the japanese smokes parlaiments.
member(h(japanese,_,parlaiments,_,_), Hs),
%% 14. the norwegian lives next to the blue house.
nextto(h(norwegian,_,_,_,_), h(_,_,_,_,blue), Hs),
% one of them drinks water
member(h(_,_,_,water,_), Hs),
% one of them owns a zebra
member(h(_,zebra,_,_,_), Hs).

nextto(A, B, Ls) :- append(_, [A,B|_], Ls).
nextto(A, B, Ls) :- append(_, [B,A|_], Ls).
rightof(A, B, Ls) :- append(_, [A,B|_], Ls).


```

```
43 %% For the question
44 zebra_owner(Owner) :-
45 houses(Hs),
46 member(h(Owner,zebra,_,_,_), Hs).
47
48 water_drinker(Drinker) :-
49 houses(Hs),
50 member(h(Drinker,_,_,water,_), Hs).
51
52 %% Queries
53 %% ?- zebra_owner(X).
54 %% ?- water_drinker(X).
```