Sample Problems from Past Finals

Instructions:

- Please write your answers clearly to avoid losing points. For the coding questions, the algorithms and steps are the most important. We will not reduce your points because of the small syntax errors. You are also encouraged to add comments to clarify your code.
- Hint is provided for some questions, but your solutions don't have to be related to the hint. For some problems, there are more than one way to solve them.
- If you run out of space, you always can use the back of the paper to write your answers.
- Good luck!!!

Your Name: _____

1. Select *all* the correct answers for the following questions.

   (a) (4 pt) What is/are the decision(s) relevant to designing a language with references? **ABCD**

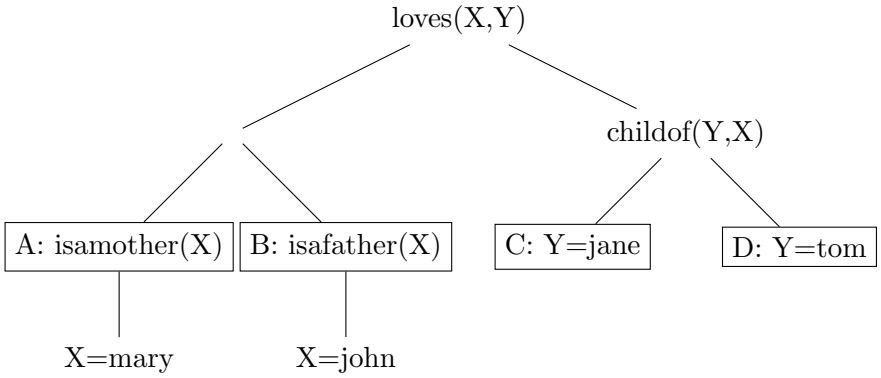
   i. manual memory management
   ii. typed heap
   iii. implicit reference
   iv. reference arithmetic

   (b) (4 pt) Which of the following understandings about type is/are correct: **ABCD**

   i. Types classify values of a program
   ii. Types specify range of values and operations on the value
   iii. Every program construct has a type in a strongly typed programming language
   iv. Type checking helps avoid execution errors

   (c) (4 pt) Consider the logic program and its resolving graph below. Where will we perform backtracking? Select the A, B, C and/or D node(s) from the resolving graph. **ABCD**

```
isamother(mary).
isafather(john)
childof(jane, mary).
childof(tom, mary).
childof(jane, john).
childof(tom, john).
loves(X, Y) :- isamother(X), childof(Y, X).
loves(X, Y) :- isafather(X), childof(Y, X).
```



2. (15 pt) The goal of this problem is to help you understand the Church encoding and how lambda calculus can be used to express computing.

   (a) Given *Pred*: $(\lambda(n)(\lambda(f)(\lambda(x)(((n(\lambda(g)(\lambda(h)h(gf))))(\lambda(u)x))(\lambda(u)u)))))$, define subtraction of integers and explain why your lambda expressions compute subtraction.

   (b) Define the logic Boolean operations of *xor a b* using *true*, *false* and *ite* given in the lecture.

   Sol

   (a) $SUB = (\lambda(m)(\lambda(n)(\lambda(f)(\lambda(x)(m\ Pred)((n\ f)x)))))$
   The inner most $((nf)x)$ represents a natural number n, i.e. f is applied n times to x. Using the provided $Pred$ one time will "unapply" one f, i.e. $Pred(f(f(x))) = f(x)$. By applying to $Pred$ m times, we unapply f m times. Thus the result will the $n - m$.

   (b) XOR:

```
(ite a
     (ite b true false)
     (ite b false true))
```

3. (5 pt) Given *false*: $(\lambda(x)(\lambda(y)\, y))$
   ite: $(\lambda(c)(\lambda(t)(\lambda(e)((c\, t)\, e))))$
   Prove or disprove the following:
   $((ite\ false)\ s1)\ s2) = s2$

4. (5 pt) Write a Prolog program: `subset(A, B)` Return *true* if A is a subset of B, else return *false*. (Hint: you can directly use the member function here)

   Sol

```
subset([], []).
subset([], [_|_]).
subset([H|T], Set) :-
    member(H, Set),
    subset(T, Set).
```

5. (5 pt)Write a short prolog program that returns

   - true if element A is in the tail of list B.
   - false if element A is the Head of B.

   Sol

```
    member(X,[X|_]).
    member(Y,[_|Tail]) :- member(Y,Tail).
    question1(A,[B|Tail]) :- (A \== B),member(A,Tail).
```

6. (15 pts) Write a Prolog program that reverses a list and any nested lists. Eg. [1,2,[2,4],5] = [5,[4,2],2,1].

   Sol

```
reverse([],Z,Z).
reverse([H|T],Z,Acc):- reverse(T,Z,[H|Acc]).
reverseAll(List,Result):- reverseAllHelper(List,Result,[]).
reverseAllHelper([],Acc,Acc).
reverseAllHelper([[A|B]|T],Result,Acc):-
  reverse([A|B],X,[]), reverseAllHelper(T,Result,[X|Acc]).
reverseAllHelper([H|T],Result,Acc):- reverseAllHelper(T,Result,[H|Acc]).
```

Extra Credit

(a) (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).


%% For the question
zebra_owner(Owner) :-
    houses(Hs),
    member(h(Owner,zebra,_,_,_), Hs).

water_drinker(Drinker) :-
    houses(Hs),
    member(h(Drinker,_,_,water,_), Hs).

%% Queries
%% ?- zebra_owner(Owner).
%% ?- water_drinker(Drinker).
```