Logic Programming

Example exam

1. a) Please define a predicate `member(X,L)`, that is fulfilled if `X` appears in the list `L`.

   b) Please extend predicate `member(X,L,)`, so that it is fulfilled, if `X` appears int `L` at an arbitrary level. Examples:

   ```
   ?- member(4,[1,2,[3,4,[5]],[6,7]]).
   True
   ?- member(0,[1,2,[3,4,[5]],[6,7]]).
   False
   ```

2. a) Please define a predicate `suffix(L1,L2)`, that is fulfilled if list `L1` equals the end of `L2`. Examples:

   ```
   ?- suffix([1,2,3],[1,2,3,4,5,6]).
   False
   ?- suffix([1,2,3],[3,4,5,1,2,3]).
   True
   ```

   b) Please define a predicate `palindrom(L)`, that is fulfilled if list `L` contains a palindrom. Examples: ?- palindrom([a,b,a,a]). False ?- palindrom([a,b,a,b,a]). True

3. a) Please define a predicate `split(X,L,L1,L2)`, that is fulfilled if list `L1` contains the elements of `L` smaller than `X` and `L2` the ones greater than `X`. Example:

   ```
   ?- split(5,[2,7,4,8,-1,5],L1,L2).
   L1 = [2,4,-1,5],
   L2 = [7,8]
   ```

   b) Please define a predicate `split(P,L,L1,L2)`, that is fulfilled if list `L1` contains the elements of `L` fulfilling predicate `P` and `L2` the ones not filfilling `P`. Example:

   ```
   odd(X) :- 1 is X mod 2.

   ?- split(odd,[2,7,4,8,-1,5],L1,L2).
   L1 = [7,-1,5],
   L2 = [2,4,8]
   ```

4. Binary trees `T` can be represented by terms: `nil` is the empty tree and `node(X,L,R)` is the tree with root `X`, left subtree `L`, and right subtree `R`. Please define the follwing predicates for binary trees.

   a) `search(T,X)`, that is fulfilled if `X` appears in `T`.

   b) `prod(T,P)`, that is fulfilled if `P` is the product of all elements in `T`.

c) `postorder(T,L)`, that is fulfilled if list `L` contains the element of `T` in post order.

5. For the following programs and queries please give the (complete) SLD-tree. What answers gives Prolog?

a)
```
p(X,Y) :- q(X,Y).
p(X,Y) :- r(X,Y).
q(X,Y) :- s(X), !, t(X).
r(c,d).
s(a).
s(b).
t(a).
t(b).

?- p(X,Y).
```

b)
```
erasePairs([],[]).
erasePairs([X,X|L1],L2) :- erasePairs(L1,L2).
erasePairs([X|L1],[X|L2]) :- erasePairs(L1,L2).

?-erasePairs([1,2,2],L).
```