Solutions

Data structures 2

Home exercises

1. build the search tree for:
   `?- member(c,[a,c,b]).`
   `?- plus1(Y,X,s(s(s(s(s(0)))))).` and
   `?- reverse([a,b,c],X).`
2. Write the PROLOG programs `times`, `power`, `factorial`, `minimum` using the definitions given for natural numbers.
3. Write the PROLOG programs `suffix`,`subset`,`intersection` using lists to represent sets.
4. Write a PROLOG program for a depth-first visit of possibly cyclic graphs, represented through the relation `arc(X,Y)`
5. Write a PROLOG program implementing insertion sort on lists.

Exercise 2

PROLOG programs `times, power, factorial, minimum`:

```
times1(0,_X,0).
times1(s(X),Y,Z) :- times1(X,Y,XY), plus1(XY,Y,Z).

exp1(s(_N),0,0).
exp1(0,s(_X),s(0)).
exp1(s(N),X,Y) :- exp1(N,X,Z), times1(Z,X,Y).

fac(0,s(0)).
fac(s(N),F) :- fac(N,F1), times1(s(N),F1,F).

minimum(N1,N2,N1) :- lesseq1(N1,N2).
minimum(N1,N2,N2) :- lesseq1(N2,N1).
```

Prolog: Exercise 3

PROLOG programs `suffix`, `subset`, `intersection` using lists to represent sets.

```
suffix(Xs,Xs).
suffix(Xs,[_Y|Ys]) :- suffix(Xs,Ys).
subset([],Xs).
subset([Y|Ys],Xs) :-
    member1(Y,Xs),subset(Ys,Xs).
intersection([],Xs,[]).
intersection([Y|Ys],Xs,[Y|Z]) :-
    member1(Y,Xs),intersection(Ys,Xs,Z).
intersection([_Y|Ys],Xs,Z) :-
    intersection(Ys,Xs,Z).
```

Exercise 4: return the path

Write a PROLOG program that returns the path between two nodes of a graph, represented through the relation `arc(X,Y)`, through a depth-first visit.

```
connected(Node,Node,[]).
connected(X,Node2,[X|L]) :- arc(X,NodeInt),
                            connected(NodeInt,Node2,L).
```

Exercise 4: visit of a cyclic structure

Write a PROLOG program for a depth-first visit of possibly cyclic graphs, represented through the relation `arc(X,Y)`

```
nonmember(X,[Y|Ys]) :- X \== Y, nonmember(X,Ys).
nonmember(_X,[]).

connected(X,Y) :- connectedcyclic(X,Y,[X]).

connectedcyclic(X,X,_Visited).
connectedcyclic(X,Y,Visited) :- arc(X,N),
        nonmember(N,Visited),
        connectedcyclic(N,Y,[N|Visited]).
```

Exercise 5: sorting lists

```
sort1(Xs,Ys) :- permutation(Xs,Ys), ordered(Ys).

permutation(Xs,[Z|Zs]) :- select(Z,Xs,Ys),
                          permutation(Ys,Zs).
permutation([],[]).

ordered([]).
ordered([_X]).
ordered([X,Y|Ys]) :- X =< Y, ordered([Y|Ys]).

select(X,[X|Xs],Xs).
select(X,[Y|Ys],[Y,Zs]):-select(X,Ys,Zs).
```

Exercise 5: insertion sort

A list is sorted if:

- it is empty.
- it has at least one element which must be inserted **in order** in the rest of the list suitably sorted.

```
sort([],[]).
sort([X|Xs], Ys):-sort(Xs, Zs), insert(X, Zs, Ys).
insert(X, [], [X]).
insert(X, [Y|Ys], [Y|Zs]):- X>Y, insert(X, Ys, Zs).
insert(X, [Y|Ys], [X,Y|Ys]):- X=<Y.
```

Prolog: pair-exchange sort

```
sort1(Xs,Xs) :- ordered(Xs).
sort1(Xs,Ys) :- append1(As,[X,Y|Bs],Xs), X > Y,
                append1(As,[Y,X|Bs],Xs1),
                sort1(Xs1,Ys).
```