```
parent(alice, bob).
parent(alice, rose).
parent(alice, jenny).
parent(alexander, charlie).
parent(alexander, tom).
parent(ray, bob).
parent(ray, caroline).
parent(susan, charlie).

male(alexander).
male(bob).
male(charlie).
male(ray).
male(tom).

female(alice).
female(susan).
female(rose).
female(jenny).
female(caroline).

/* 1. Using the structures parent(X, Y), male(X), and female(X), write a
structure that defines mother(X, Y) */
mother(X,Y) :- parent(X,Y),
               female(X).

/* 2. Using the structures parent(X, Y), male(X), and female(X), write a
structure that defines sister(X, Y) */
sister(X,Y) :- parent(M,X),
               parent(M,Y),
               female(X).

/* 3. Write a Prolog program that finds the maximum of a list of numbers
*/
max_list([H|T],M):-
max_list(T,H,M).
max_list([],C,C) :-
        write('List empty').
max_list([H|T],C,M):-
        C2 is max(C,H),
        max_list(T,C2,M).

/* 4. Write a Prolog program that succeeds if the intersection of two
given list parameters is empty */
intersection([], _, []).
intersection([Head|L1tail], L2, L3) :-
        memberchk(Head, L2),
        !,
        L3 = [Head|L3tail],
        intersection(L1tail, L2, L3tail).
intersection([_|L1tail], L2, L3) :-
        intersection(L1tail, L2, L3).
```

```
/* 5. Write a Prolog program that returns a list containing the union of
the elements of two given lists */
union([], L, L).
union([Head|L1tail], L2, L3) :-
        memberchk(Head, L2),
        !,
        union(L1tail, L2, L3).
union([Head|L1tail], L2, [Head|L3tail]) :-
        union(L1tail, L2, L3tail).

/* 6. Write a Prolog program that returns the final element of a given
list */
final(X,[X]).
final(X,[Y|Z]):- final(X,Z).


/* 7. Write a Prolog program that implements quicksort */
pivot(_, [], [], []).
pivot(Pivot, [Head|Tail], [Head|LessOrEqualThan], GreaterThan) :- Pivot
>= Head, pivot(Pivot, Tail, LessOrEqualThan, GreaterThan).
pivot(Pivot, [Head|Tail], LessOrEqualThan, [Head|GreaterThan]) :-
pivot(Pivot, Tail, LessOrEqualThan, GreaterThan).

quicksort([], []).
quicksort([Head|Tail], Sorted) :- pivot(Head, Tail, List1, List2),
quicksort(List1, SortedList1), quicksort(List2, SortedList2),
append(SortedList1, [Head|SortedList2], Sorted).
```