Question 1

```
/**
 * ancestralLine(L): L is a list of people making up a direct line through
 * parentage
 *
 * Example Queries:
 * ?- ancestralLine([frodo|T]).
 * T = [] ;
 * T = [drogo] ;
 * T = [drogo, fosco] ;
 * T = [drogo, fosco, largo] ;
 * T = [drogo, fosco, largo, balbo] ;
 * T = [drogo, fosco, largo, berylla] ;
 * T = [drogo, fosco, tanta] ;
 * T = [drogo, ruby] ;
 * T = [primula] ;
 * false.
 * ?- ancestralLine([bilbo|T]).
 * T = [] ;
 * T = [bungo] ;
 * T = [bungo, mungo] ;
 * T = [bungo, mungo, balbo] ;
 * T = [bungo, mungo, berylla] ;
 * T = [bungo, laura] ;
 * T = [belladonna] ;
 * false.
 */
ancestralLine([_]).
ancestralLine([Child | L]) :-
  parent(Parent, Child),
  ancestralLine([Parent | NewL]),
  L = [Parent | NewL].
```

Question 2

```
/**
 * friendGroup(N,G): G is a group of N people who are all friends with each
 * other.
 *
 * You can use the friend facts from SA #5 for testing
 *
 * Example Query:
 * ?- friendGroup(3,X).
 * X = [deshawn, anna, ali] ;
 * X = [deshawn, ali, anna] ;
 * X = [coco, elena, lucas] ;
 * X = [coco, lucas, elena] ...
 */
friendGroup(N, G) :-
  length(G, N),
  allFriends(G).

friends(X, Y) :-
  friend(X, Y);
  friend(Y, X).

isFriendsWithAll(_, []).
isFriendsWithAll(X, [P | L]) :-
  friends(X, P),
  isFriendsWithAll(X, L).

allFriends([_]).
allFriends([P | G]) :-
  isFriendsWithAll(P, G),
  allFriends(G).
```

Question 3

```
/**
 * everyOtherOne(X,Y): X and Y are lists and Y contains every other
 * element in X.
 *
 * Example Queries:
 * ?- everyOtherOne([1,2,3,4],X).
 * X = [1, 3].
 * ?- everyOtherOne([1,2,3,4,5],X).
 * X = [1, 3, 5]
 */
everyOtherOne([], []).
everyOtherOne([X], [X]).
everyOtherOne([X1, _ | TX], [Y1 | TY]) :-
  Y1 = X1,
  everyOtherOne(TX, TY).
```

Question 4

```
/**
 * removeDuplicates(X,Y): X and Y are lists and Y is X with all the
 * duplicates removed. Do not use sort.
 *
 * Example Query:
 * ?- removeDuplicates([1,0,2,0,3,3,6,0,4],X).
 * X = [1,2,3,6,0,4]
 */
removeDuplicates([],[]).
removeDuplicates([XH | XT], [YH | YT]) :-
  % Pop X until X's head is not a dup.
  (member(XH, XT), removeDuplicates(XT, [YH | YT])) ;
  % Assert heads are equal and proceed.
  (XH = YH, removeDuplicates(XT, YT)).
```