NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

SC3000 – Artificial Intelligence

Assignment 2: Learning to Use Prolog as a Logic Programming Tool

Lab Group	SCSB
Members	Siah Yee Long (U2220887B)
	Teo Xuan Ming (U2220719A)

Table of Contents

Note:

- Words highlighted in blue and are lowercased are constants.
- Words highlighted in orange and are PascalCased are Variables.

Exercise 1: The Smart Phone Rivalry

sumsum, a competitor of appy, developed some nice smart phone technology called galactica-s3, all of which was stolen by stevey, who is a boss of appy. It is unethical for a boss to steal business from rival companies. A competitor is a rival. Smart phone technology is business.

1.1.

Translate the natural language statements above describing the dealing within the Smart Phone industry into First Order Logic (FOL).

Statement:
sumsum, a competitor of appy
FOL:

$$competitor(Sumsum, Appy)$$

Statement:
(sumsum) developed some nice smart phone technology called galactica-s3
FOL:

$$develop(Sumsum, GalacticaS3)$$

Statement:
all of which (galactica-s3) was stolen by stevey
FOL:

$$steal(Stevey, GalacticaS3)$$

Statement:
who (stevey) is a boss of appy
FOL:

$$boss(Appy, Stevey)$$

Statement:
It is unethical for a boss to steal business from rival companies
FOL:

$$\begin{gathered}\forall x, \forall companyx, \forall companyy, \forall technologyy \\ boss(companyx,) \land steal(x, technologyy) \\ \land develop(companyy, technologyy) \land business(technologyy) \\ \land rival(companyx, companyy) \\ \Longrightarrow unethical(x)\end{gathered}$$

Statement:
A competitor is a rival
FOL:

$$\forall x, \forall y, competitor(x, y) \implies rival(x, y)$$

Statement:
Smart phone technology is business
FOL:

$$business(GalacticaS3)$$

1.2.

Write these FOL statements as Prolog clauses.

```
/* Exercise 1 */

% Facts
competitor(sumsum, appy). % sumsum is a competitor of appy
competitor(appy, sumsum). % appy is also a competitor of sumsum
develop(sumsum, galactica-s3). % sumsum produced some nice smart phone tech
steal(stevey,galactica-s3). % stevey steals galactica-s3
boss(appy, stevey). % the boss of appy is stevey

% Rules
rival(X,Y) :- competitor(X,Y). % competitor relationship same as rivalry

unethical(Person) :- % a boss is unethical if he steals business from rivals
    boss(Boss_Company,Person), % if the person in question is a boss of a company
    rival(Boss_Company, Rival_Company), % if boss's company has a rival company
    develop(Rival_Company, Technology), % if a technology belongs to the rival
company
    business(Technology), % if the technology is a business
    steal(Person, Technology). % that the person has stolen

business(galactica-s3).
```

1.3.

Using Prolog, prove that Stevey is unethical. Show a trace of your proof.

```
?- ['exercise1.pl'].
true.
?- trace,  unethical(stevey).
   Call: (13) unethical(stevey) ? creep
   Call: (14) boss(_7954, stevey) ? creep
   Exit: (14) boss(appy, stevey) ? creep
   Call: (14) rival(appy, _9760) ? creep
   Call: (15) competitor(appy, _9760) ? creep
   Exit: (15) competitor(appy, sumsum) ? creep
   Exit: (14) rival(appy, sumsum) ? creep
   Call: (14) develop(sumsum, _13370) ? creep
   Exit: (14) develop(sumsum, galactica-s3) ? creep
   Call: (14) business(galactica-s3) ? creep
   Exit: (14) business(galactica-s3) ? creep
   Call: (14) steal(stevey, galactica-s3) ? creep
   Exit: (14) steal(stevey, galactica-s3) ? creep
   Exit: (13) unethical(stevey) ? creep
true.
```

Exercise 2: The Royal Family

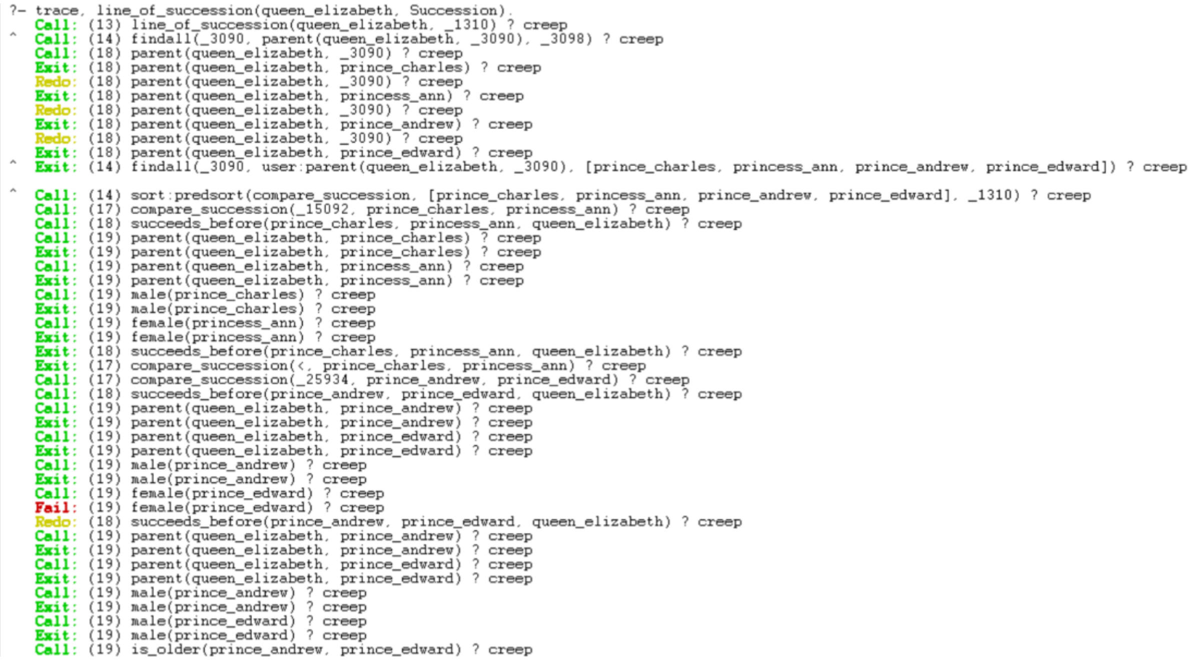
The old Royal succession rule states that the throne is passed down along the male line according to the order of birth before the consideration along the female line – similarly according to the order of birth. queen elizabeth, the monarch of United Kingdom, has four offsprings; namely:- prince charles, princess ann, prince andrew and prince edward – listed in the order of birth.

2.1.

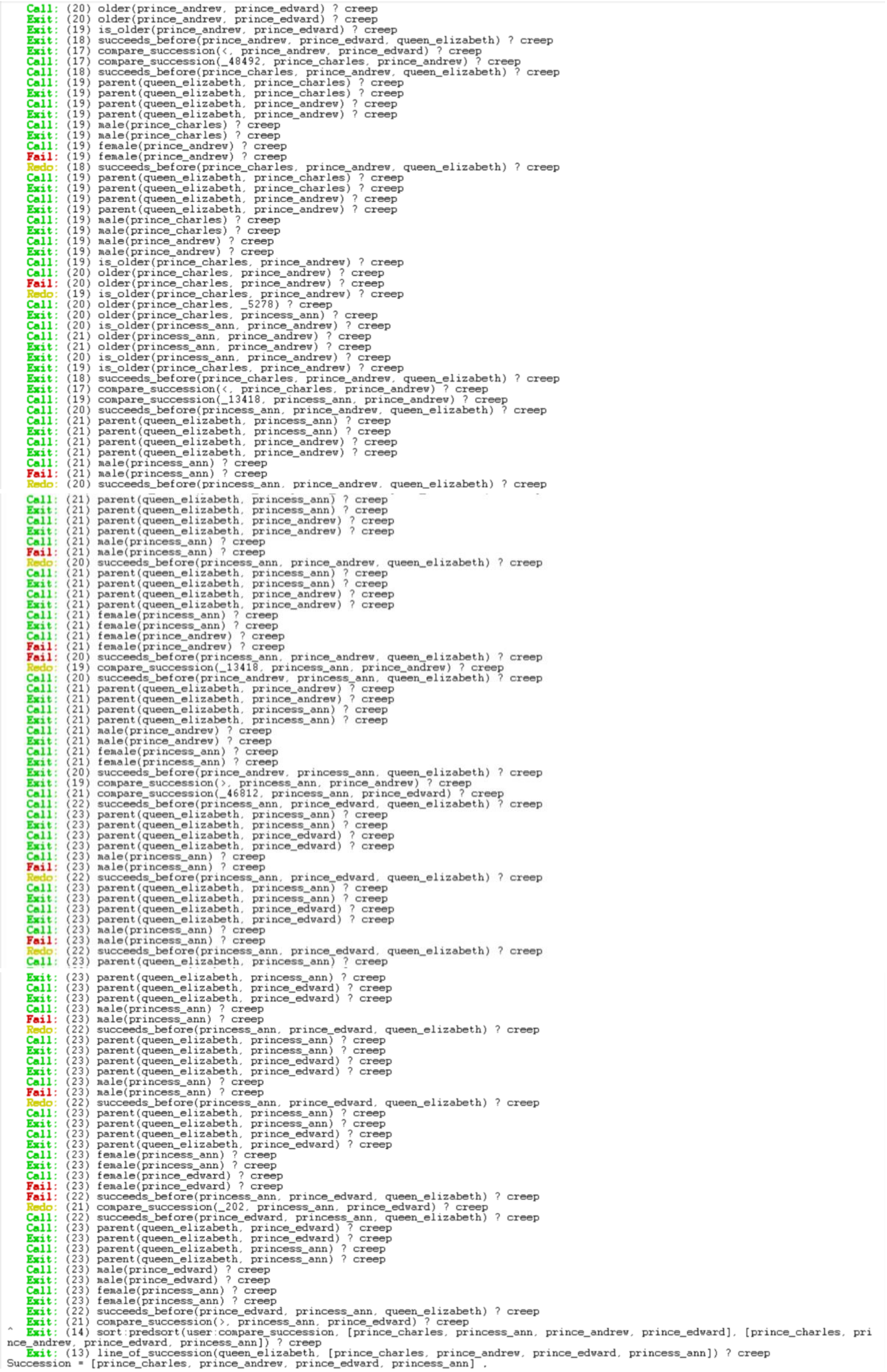
Code

```
/* Exercise 2.1 prolog code snippet */

% Facts gender: male(Person). female(Person).
female(queen_elizabeth).
male(prince_charles).
female(princess_ann).
male(prince_andrew).
male(prince_edward).

% Parent-child relation: parent(Parent, Child).
parent(queen_elizabeth, prince_charles).
parent(queen_elizabeth, princess_ann).
parent(queen_elizabeth, prince_andrew).
parent(queen_elizabeth, prince_edward).

% Define order of birth : older(OlderSibling, YoungerSibling).
older(prince_charles, princess_ann).
older(princess_ann, prince_andrew).
older(prince_andrew, prince_edward).

% Transitive rule to determine if someone is older than another
% is_older(PersonA, PersonB) means PersonA is older than PersonB.
is_older(X, Y) :- older(X, Y). % Base case: Directly older
is_older(X, Y) :- older(X, Z), is_older(Z, Y). % Recursive case

% Rule defining the old succession preference:
% succeeds_before(Heir1, Heir2, Queen) means Heir1 comes before Heir2
% Case 1: A male heir comes before a female heir.
succeeds_before(Heir1, Heir2, Monarch) :-
       parent(Monarch, Heir1),
       parent(Monarch, Heir2),
       male(Heir1),
       female(Heir2).

% Case 2: Both are male; the older one comes first.
succeeds_before(Heir1, Heir2, Monarch) :-
       parent(Monarch, Heir1),
       parent(Monarch, Heir2),
       male(Heir1),
       male(Heir2),
       is_older(Heir1, Heir2).

% Case 3: Both are female; the older one comes first.
succeeds_before(Heir1, Heir2, Monarch) :-
       parent(Monarch, Heir1),
       parent(Monarch, Heir2),
       female(Heir1),
       female(Heir2),
       is_older(Heir1, Heir2).

% Helper predicates for sorting based on the succession rule.
% Order is '<' if Person1 precedes Person2, '>' if Person2 precedes Person1.
compare_succession(<, P1, P2) :-
       succeeds_before(P1, P2, queen_elizabeth). % P1 comes before P2
compare_succession(>, P1, P2) :-
       succeeds_before(P2, P1, queen_elizabeth). % P2 comes before P1

% Rule to determine the line of succession for a given monarch.
line_of_succession(Queen, SuccessionList) :-

       findall(Child, parent(Queen, Child), Children),
       predsort(compare_succession, Children, SuccessionList).
```

Trace



```
?- trace, line_of_succession(queen_elizabeth, Succession).
   Call: (13) line_of_succession(queen_elizabeth, _1310) ? creep
^  Call: (14) findall(_3090, parent(queen_elizabeth, _3090), _3098) ? creep
   Call: (18) parent(queen_elizabeth, _3090) ? creep
   Exit: (18) parent(queen_elizabeth, prince_charles) ? creep
   Redo: (18) parent(queen_elizabeth, _3090) ? creep
   Exit: (18) parent(queen_elizabeth, princess_ann) ? creep
   Redo: (18) parent(queen_elizabeth, _3090) ? creep
   Exit: (18) parent(queen_elizabeth, prince_andrew) ? creep
   Redo: (18) parent(queen_elizabeth, _3090) ? creep
   Exit: (18) parent(queen_elizabeth, prince_edward) ? creep
^  Exit: (14) findall(_3090, user:parent(queen_elizabeth, _3090), [prince_charles, princess_ann, prince_andrew, prince_edward]) ? creep

^  Call: (14) sort:predsort(compare_succession, [prince_charles, princess_ann, prince_andrew, prince_edward], _1310) ? creep
   Call: (17) compare_succession(_15092, prince_charles, princess_ann) ? creep
   Call: (18) succeeds_before(prince_charles, princess_ann, queen_elizabeth) ? creep
   Call: (19) parent(queen_elizabeth, prince_charles) ? creep
   Exit: (19) parent(queen_elizabeth, prince_charles) ? creep
   Call: (19) parent(queen_elizabeth, princess_ann) ? creep
   Exit: (19) parent(queen_elizabeth, princess_ann) ? creep
   Call: (19) male(prince_charles) ? creep
   Exit: (19) male(prince_charles) ? creep
   Call: (19) female(princess_ann) ? creep
   Exit: (19) female(princess_ann) ? creep
   Exit: (18) succeeds_before(prince_charles, princess_ann, queen_elizabeth) ? creep
   Exit: (17) compare_succession(<, prince_charles, princess_ann) ? creep
   Call: (17) compare_succession(_25934, prince_andrew, prince_edward) ? creep
   Call: (18) succeeds_before(prince_andrew, prince_edward, queen_elizabeth) ? creep
   Call: (19) parent(queen_elizabeth, prince_andrew) ? creep
   Exit: (19) parent(queen_elizabeth, prince_andrew) ? creep
   Call: (19) parent(queen_elizabeth, prince_edward) ? creep
   Exit: (19) parent(queen_elizabeth, prince_edward) ? creep
   Call: (19) male(prince_andrew) ? creep
   Exit: (19) male(prince_andrew) ? creep
   Call: (19) female(prince_edward) ? creep
   Fail: (19) female(prince_edward) ? creep
   Redo: (18) succeeds_before(prince_andrew, prince_edward, queen_elizabeth) ? creep
   Call: (19) parent(queen_elizabeth, prince_andrew) ? creep
   Exit: (19) parent(queen_elizabeth, prince_andrew) ? creep
   Call: (19) parent(queen_elizabeth, prince_edward) ? creep
   Exit: (19) parent(queen_elizabeth, prince_edward) ? creep
   Call: (19) male(prince_andrew) ? creep
   Exit: (19) male(prince_andrew) ? creep
   Call: (19) male(prince_edward) ? creep
   Exit: (19) male(prince_edward) ? creep
   Call: (19) is_older(prince_andrew, prince_edward) ? creep
```



```
    Call: (20) older(prince_andrew, prince_edward) ? creep
    Exit: (20) older(prince_andrew, prince_edward) ? creep
    Exit: (19) is_older(prince_andrew, prince_edward) ? creep
    Exit: (18) succeeds_before(prince_andrew, prince_edward, queen_elizabeth) ? creep
    Exit: (17) compare_succession(<, prince_andrew, prince_edward) ? creep
    Call: (17) compare_succession(_48492, prince_charles, prince_andrew) ? creep
    Call: (18) succeeds_before(prince_charles, prince_andrew, queen_elizabeth) ? creep
    Call: (19) parent(queen_elizabeth, prince_charles) ? creep
    Exit: (19) parent(queen_elizabeth, prince_charles) ? creep
    Call: (19) parent(queen_elizabeth, prince_andrew) ? creep
    Exit: (19) parent(queen_elizabeth, prince_andrew) ? creep
    Call: (19) male(prince_charles) ? creep
    Exit: (19) male(prince_charles) ? creep
    Call: (19) female(prince_andrew) ? creep
    Fail: (19) female(prince_andrew) ? creep
    Redo: (18) succeeds_before(prince_charles, prince_andrew, queen_elizabeth) ? creep
    Call: (19) parent(queen_elizabeth, prince_charles) ? creep
    Exit: (19) parent(queen_elizabeth, prince_charles) ? creep
    Call: (19) parent(queen_elizabeth, prince_andrew) ? creep
    Exit: (19) parent(queen_elizabeth, prince_andrew) ? creep
    Call: (19) male(prince_charles) ? creep
    Exit: (19) male(prince_charles) ? creep
    Call: (19) male(prince_andrew) ? creep
    Exit: (19) male(prince_andrew) ? creep
    Call: (19) is_older(prince_charles, prince_andrew) ? creep
    Call: (20) older(prince_charles, prince_andrew) ? creep
    Fail: (20) older(prince_charles, prince_andrew) ? creep
    Redo: (19) is_older(prince_charles, prince_andrew) ? creep
    Call: (20) older(prince_charles, _5278) ? creep
    Exit: (20) older(prince_charles, princess_ann) ? creep
    Call: (20) is_older(princess_ann, prince_andrew) ? creep
    Call: (21) older(princess_ann, prince_andrew) ? creep
    Exit: (21) older(princess_ann, prince_andrew) ? creep
    Exit: (20) is_older(princess_ann, prince_andrew) ? creep
    Exit: (19) is_older(prince_charles, prince_andrew) ? creep
    Exit: (18) succeeds_before(prince_charles, prince_andrew, queen_elizabeth) ? creep
    Exit: (17) compare_succession(<, prince_charles, prince_andrew) ? creep
    Call: (19) compare_succession(_13418, princess_ann, prince_andrew) ? creep
    Call: (20) succeeds_before(princess_ann, prince_andrew, queen_elizabeth) ? creep
    Call: (21) parent(queen_elizabeth, princess_ann) ? creep
    Exit: (21) parent(queen_elizabeth, princess_ann) ? creep
    Call: (21) parent(queen_elizabeth, prince_andrew) ? creep
    Exit: (21) parent(queen_elizabeth, prince_andrew) ? creep
    Call: (21) male(princess_ann) ? creep
    Fail: (21) male(princess_ann) ? creep
    Redo: (20) succeeds_before(princess_ann, prince_andrew, queen_elizabeth) ? creep
    Call: (21) parent(queen_elizabeth, princess_ann) ? creep
    Exit: (21) parent(queen_elizabeth, princess_ann) ? creep
    Call: (21) parent(queen_elizabeth, prince_andrew) ? creep
    Exit: (21) parent(queen_elizabeth, prince_andrew) ? creep
    Call: (21) male(princess_ann) ? creep
    Fail: (21) male(princess_ann) ? creep
    Redo: (20) succeeds_before(princess_ann, prince_andrew, queen_elizabeth) ? creep
    Call: (21) parent(queen_elizabeth, princess_ann) ? creep
    Exit: (21) parent(queen_elizabeth, princess_ann) ? creep
    Call: (21) parent(queen_elizabeth, prince_andrew) ? creep
    Exit: (21) parent(queen_elizabeth, prince_andrew) ? creep
    Call: (21) female(princess_ann) ? creep
    Exit: (21) female(princess_ann) ? creep
    Call: (21) female(prince_andrew) ? creep
    Fail: (21) female(prince_andrew) ? creep
    Fail: (20) succeeds_before(princess_ann, prince_andrew, queen_elizabeth) ? creep
    Redo: (19) compare_succession(_13418, princess_ann, prince_andrew) ? creep
    Call: (20) succeeds_before(prince_andrew, princess_ann, queen_elizabeth) ? creep
    Call: (21) parent(queen_elizabeth, prince_andrew) ? creep
    Exit: (21) parent(queen_elizabeth, prince_andrew) ? creep
    Call: (21) parent(queen_elizabeth, princess_ann) ? creep
    Exit: (21) parent(queen_elizabeth, princess_ann) ? creep
    Call: (21) male(prince_andrew) ? creep
    Exit: (21) male(prince_andrew) ? creep
    Call: (21) female(princess_ann) ? creep
    Exit: (21) female(princess_ann) ? creep
    Exit: (20) succeeds_before(prince_andrew, princess_ann, queen_elizabeth) ? creep
    Exit: (19) compare_succession(>, princess_ann, prince_andrew) ? creep
    Call: (21) compare_succession(_46812, princess_ann, prince_edward) ? creep
    Call: (22) succeeds_before(princess_ann, prince_edward, queen_elizabeth) ? creep
    Call: (23) parent(queen_elizabeth, princess_ann) ? creep
    Exit: (23) parent(queen_elizabeth, princess_ann) ? creep
    Call: (23) parent(queen_elizabeth, prince_edward) ? creep
    Exit: (23) parent(queen_elizabeth, prince_edward) ? creep
    Call: (23) male(princess_ann) ? creep
    Fail: (23) male(princess_ann) ? creep
    Redo: (22) succeeds_before(princess_ann, prince_edward, queen_elizabeth) ? creep
    Call: (23) parent(queen_elizabeth, princess_ann) ? creep
    Exit: (23) parent(queen_elizabeth, princess_ann) ? creep
    Call: (23) parent(queen_elizabeth, prince_edward) ? creep
    Exit: (23) parent(queen_elizabeth, prince_edward) ? creep
    Call: (23) male(princess_ann) ? creep
    Fail: (23) male(princess_ann) ? creep
    Redo: (22) succeeds_before(princess_ann, prince_edward, queen_elizabeth) ? creep
    Call: (23) parent(queen_elizabeth, princess_ann) ? creep
    Exit: (23) parent(queen_elizabeth, princess_ann) ? creep
    Call: (23) parent(queen_elizabeth, prince_edward) ? creep
    Exit: (23) parent(queen_elizabeth, prince_edward) ? creep
    Call: (23) male(princess_ann) ? creep
    Fail: (23) male(princess_ann) ? creep
    Redo: (22) succeeds_before(princess_ann, prince_edward, queen_elizabeth) ? creep
    Call: (23) parent(queen_elizabeth, princess_ann) ? creep
    Exit: (23) parent(queen_elizabeth, princess_ann) ? creep
    Call: (23) parent(queen_elizabeth, prince_edward) ? creep
    Exit: (23) parent(queen_elizabeth, prince_edward) ? creep
    Call: (23) male(princess_ann) ? creep
    Fail: (23) male(princess_ann) ? creep
    Redo: (22) succeeds_before(princess_ann, prince_edward, queen_elizabeth) ? creep
    Call: (23) parent(queen_elizabeth, princess_ann) ? creep
    Exit: (23) parent(queen_elizabeth, princess_ann) ? creep
    Call: (23) parent(queen_elizabeth, prince_edward) ? creep
    Exit: (23) parent(queen_elizabeth, prince_edward) ? creep
    Call: (23) female(princess_ann) ? creep
    Exit: (23) female(princess_ann) ? creep
    Call: (23) female(prince_edward) ? creep
    Fail: (23) female(prince_edward) ? creep
    Fail: (22) succeeds_before(princess_ann, prince_edward, queen_elizabeth) ? creep
    Redo: (21) compare_succession(_202, princess_ann, prince_edward) ? creep
    Call: (22) succeeds_before(prince_edward, princess_ann, queen_elizabeth) ? creep
    Call: (23) parent(queen_elizabeth, prince_edward) ? creep
    Exit: (23) parent(queen_elizabeth, prince_edward) ? creep
    Call: (23) parent(queen_elizabeth, princess_ann) ? creep
    Exit: (23) parent(queen_elizabeth, princess_ann) ? creep
    Call: (23) male(prince_edward) ? creep
    Exit: (23) male(prince_edward) ? creep
    Call: (23) female(princess_ann) ? creep
    Exit: (23) female(princess_ann) ? creep
    Exit: (22) succeeds_before(prince_edward, princess_ann, queen_elizabeth) ? creep
    Exit: (21) compare_succession(>, princess_ann, prince_edward) ? creep
^   Exit: (14) sort:predsort(user:compare_succession, [prince_charles, princess_ann, prince_andrew, prince_edward], [prince_charles, pri
nce_andrew, prince_edward, princess_ann]) ? creep
    Exit: (13) line_of_succession(queen_elizabeth, [prince_charles, prince_andrew, prince_edward, princess_ann]) ? creep
Succession = [prince_charles, prince_andrew, prince_edward, princess_ann] .
```

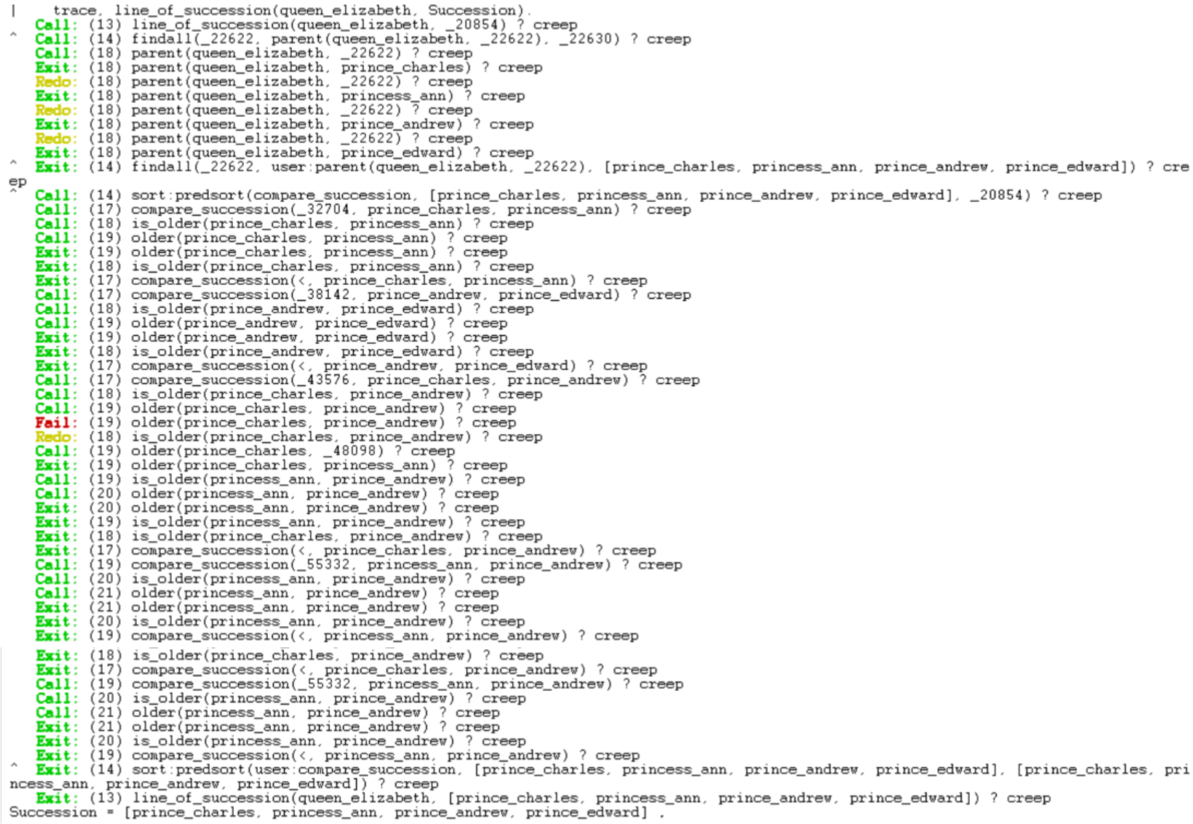
2.2.

As gender is no longer necessary in determining the next successor, it simplifies the code further. We don't need to define the genders, and the code for the cases regarding the gender is removed as there is no need to now move female candidates to the back of the succession list.

Code

```
/* Exercise 2.2 */
% Facts gender: not necessary as gender is not considered in monarch.
/*female(queen_elizabeth).
male(prince_charles).
female(princess_ann).
male(prince_andrew).
male(prince_edward).*/

% Parent-child relation: parent(Parent, Child).
parent(queen_elizabeth, prince_charles).
parent(queen_elizabeth, princess_ann).
parent(queen_elizabeth, prince_andrew).
parent(queen_elizabeth, prince_edward).

% Define order of birth : older(OlderSibling, YoungerSibling).
older(prince_charles, princess_ann).
older(princess_ann, prince_andrew).
older(prince_andrew, prince_edward).


% Transitive rule to determine if someone is older than another
% is_older(PersonA, PersonB) means PersonA is older than PersonB.
is_older(X, Y) :- older(X, Y). % Base case: Directly older
is_older(X, Y) :- older(X, Z), is_older(Z, Y). % Recursive case

% Removed all cases related to gender for new succession rule.
% Predicate is updated to not need Monarch as there is no need to
% search if P1, P2 are children of the Monarch again
compare_succession(<, P1, P2) :-
    is_older(P1, P2). % P1 comes before P2
compare_succession(>, P1, P2) :-
    is_older(P2, P1). % P2 comes before P1


% Rule to determine the line of succession for a given monarch.
line_of_succession(Queen, SuccessionList) :-
    findall(Child, parent(Queen, Child), Children),
    predsort(compare_succession, Children, SuccessionList).
```

Trace



```
|    trace, line_of_succession(queen_elizabeth, Succession).
   Call: (13) line_of_succession(queen_elizabeth, _20854) ? creep
^  Call: (14) findall(_22622, parent(queen_elizabeth, _22622), _22630) ? creep
   Call: (18) parent(queen_elizabeth, _22622) ? creep
   Exit: (18) parent(queen_elizabeth, prince_charles) ? creep
   Redo: (18) parent(queen_elizabeth, _22622) ? creep
   Exit: (18) parent(queen_elizabeth, princess_ann) ? creep
   Redo: (18) parent(queen_elizabeth, _22622) ? creep
   Exit: (18) parent(queen_elizabeth, prince_andrew) ? creep
   Redo: (18) parent(queen_elizabeth, _22622) ? creep
   Exit: (18) parent(queen_elizabeth, prince_edward) ? creep
^  Exit: (14) findall(_22622, user:parent(queen_elizabeth, _22622), [prince_charles, princess_ann, prince_andrew, prince_edward]) ? cre
ep
^  Call: (14) sort:predsort(compare_succession, [prince_charles, princess_ann, prince_andrew, prince_edward], _20854) ? creep
   Call: (17) compare_succession(_32704, prince_charles, princess_ann) ? creep
   Call: (18) is_older(prince_charles, princess_ann) ? creep
   Call: (19) older(prince_charles, princess_ann) ? creep
   Exit: (19) older(prince_charles, princess_ann) ? creep
   Exit: (18) is_older(prince_charles, princess_ann) ? creep
   Exit: (17) compare_succession(<, prince_charles, princess_ann) ? creep
   Call: (17) compare_succession(_38142, prince_andrew, prince_edward) ? creep
   Call: (18) is_older(prince_andrew, prince_edward) ? creep
   Call: (19) older(prince_andrew, prince_edward) ? creep
   Exit: (19) older(prince_andrew, prince_edward) ? creep
   Exit: (18) is_older(prince_andrew, prince_edward) ? creep
   Exit: (17) compare_succession(<, prince_andrew, prince_edward) ? creep
   Call: (17) compare_succession(_43576, prince_charles, prince_andrew) ? creep
   Call: (18) is_older(prince_charles, prince_andrew) ? creep
   Call: (19) older(prince_charles, prince_andrew) ? creep
   Fail: (19) older(prince_charles, prince_andrew) ? creep
   Redo: (18) is_older(prince_charles, prince_andrew) ? creep
   Call: (19) older(prince_charles, _48098) ? creep
   Exit: (19) older(prince_charles, princess_ann) ? creep
   Call: (19) is_older(princess_ann, prince_andrew) ? creep
   Call: (20) older(princess_ann, prince_andrew) ? creep
   Exit: (20) older(princess_ann, prince_andrew) ? creep
   Exit: (19) is_older(princess_ann, prince_andrew) ? creep
   Exit: (18) is_older(prince_charles, prince_andrew) ? creep
   Exit: (17) compare_succession(<, prince_charles, prince_andrew) ? creep
   Call: (19) compare_succession(_55332, princess_ann, prince_andrew) ? creep
   Call: (20) is_older(princess_ann, prince_andrew) ? creep
   Call: (21) older(princess_ann, prince_andrew) ? creep
   Exit: (21) older(princess_ann, prince_andrew) ? creep
   Exit: (20) is_older(princess_ann, prince_andrew) ? creep
   Exit: (19) compare_succession(<, princess_ann, prince_andrew) ? creep
   Exit: (18) is_older(prince_charles, prince_andrew) ? creep
   Exit: (17) compare_succession(<, prince_charles, prince_andrew) ? creep
   Call: (19) compare_succession(_55332, princess_ann, prince_andrew) ? creep
   Call: (20) is_older(princess_ann, prince_andrew) ? creep
   Call: (21) older(princess_ann, prince_andrew) ? creep
   Exit: (21) older(princess_ann, prince_andrew) ? creep
   Exit: (20) is_older(princess_ann, prince_andrew) ? creep
   Exit: (19) compare_succession(<, princess_ann, prince_andrew) ? creep
^  Exit: (14) sort:predsort(user:compare_succession, [prince_charles, princess_ann, prince_andrew, prince_edward], [prince_charles, pri
ncess_ann, prince_andrew, prince_edward]) ? creep
   Exit: (13) line_of_succession(queen_elizabeth, [prince_charles, princess_ann, prince_andrew, prince_edward]) ? creep
Succession = [prince_charles, princess_ann, prince_andrew, prince_edward] .
```