Name: Kannan Shivani

Matric: U1822998H

Group: TSP6

Exercise 1: The Smart Phone Rivalry (15 marks)

Translate the natural language statements above describing the dealing within the Smart Phone industry in to First Order Logic (FOL). [5 marks]

sumsum, a competitor of appy, developed some nice smart phone technology called galacticas3, all of which was stolen by stevey, who is a boss. It is unethical for a boss to steal business from rival companies. A competitor of appy is a rival. Smart phone technology is business.

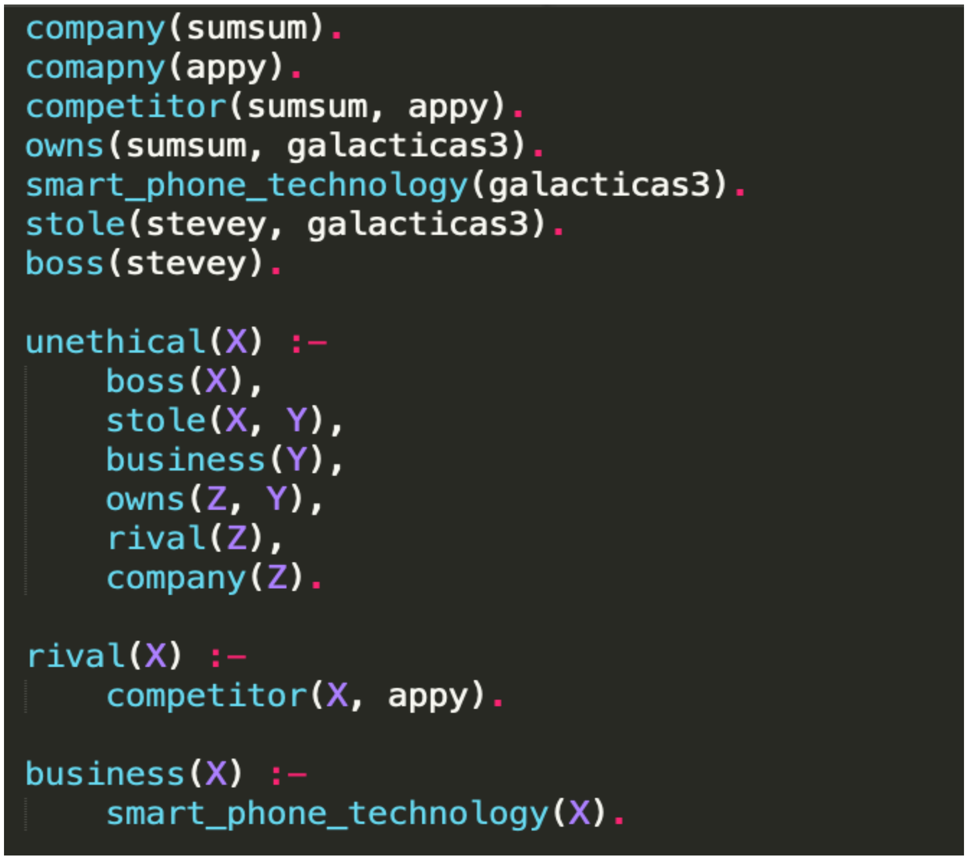
Predicate	Definition
company(x)	x is a company
competitor(x, y)	x is a competitor of y
owns(x, y)	x owns y
smart_phone_technology(x)	x is a smart phone technology
stole(x, y)	x stole y
boss(x)	x is a boss
unethical(x)	x is unethical
rival(x)	x is a rival
business(x)	x is a business

Converting the sentences to FOL statements

Sentence	FOL statement
"sumsum, a competitor of appy"	competitor(sumsum,appy)
"smart phone technology called galacticas3"	smart_phone_technology(galacticas3)
"(sumsum) developed galacticas3"	owns(sumsum,galacticas3)
"stevey, who is a boss"	boss(stevey)

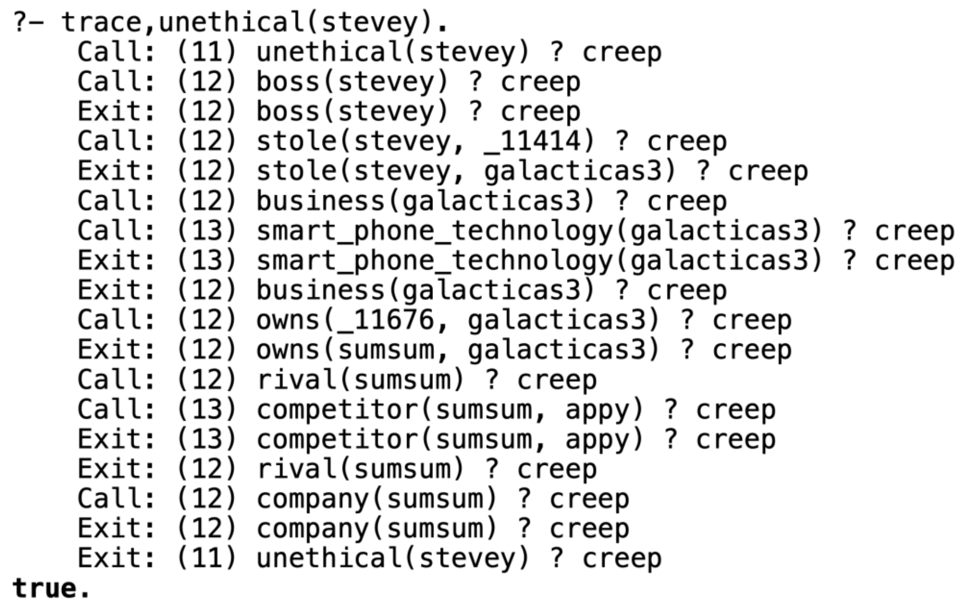
“all of which(galacticas3) was stolen by stevey”	stole(stevey,galacticas3)
“competitor of appy is a rival”	∀x, competitor(x,appy) -> rival(x)
“smart phone technology is a business”	∀x, smart_phone_technology(x) -> business(x)
“unethical for a boss to steal business from rival companies”	∀x, ∃y, ∃z, (boss(x) ∧ stole(x,y) ∧ business(y) ∧ owns(z,y) ∧ rival(z) ∧ company(z))-> unethical(x)

Write these FOL statements as Prolog clauses. (5 marks)



```
company(sumsum).
comapny(appy).
competitor(sumsum, appy).
owns(sumsum, galacticas3).
smart_phone_technology(galacticas3).
stole(stevey, galacticas3).
boss(stevey).

unethical(X) :-
    boss(X),
    stole(X, Y),
    business(Y),
    owns(Z, Y),
    rival(Z),
    company(Z).

rival(X) :-
    competitor(X, appy).

business(X) :-
    smart_phone_technology(X).
```

Using Prolog, prove that Stevey is unethical. Show a trace of your proof. (5 marks)



```
?- trace,unethical(stevey).
   Call: (11) unethical(stevey) ? creep
   Call: (12) boss(stevey) ? creep
   Exit: (12) boss(stevey) ? creep
   Call: (12) stole(stevey, _11414) ? creep
   Exit: (12) stole(stevey, galacticas3) ? creep
   Call: (12) business(galacticas3) ? creep
   Call: (13) smart_phone_technology(galacticas3) ? creep
   Exit: (13) smart_phone_technology(galacticas3) ? creep
   Exit: (12) business(galacticas3) ? creep
   Call: (12) owns(_11676, galacticas3) ? creep
   Exit: (12) owns(sumsum, galacticas3) ? creep
   Call: (12) rival(sumsum) ? creep
   Call: (13) competitor(sumsum, appy) ? creep
   Exit: (13) competitor(sumsum, appy) ? creep
   Exit: (12) rival(sumsum) ? creep
   Call: (12) company(sumsum) ? creep
   Exit: (12) company(sumsum) ? creep
   Exit: (11) unethical(stevey) ? creep
true.
```

<u>Exercise 2: The Royal Family (10 marks)</u>

Define their relations and rules in a Prolog rule base. Hence, define the old Royal succession rule. Using this old succession rule determine the line of succession based on the information given. Do a trace to show your results. [5 marks]

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.

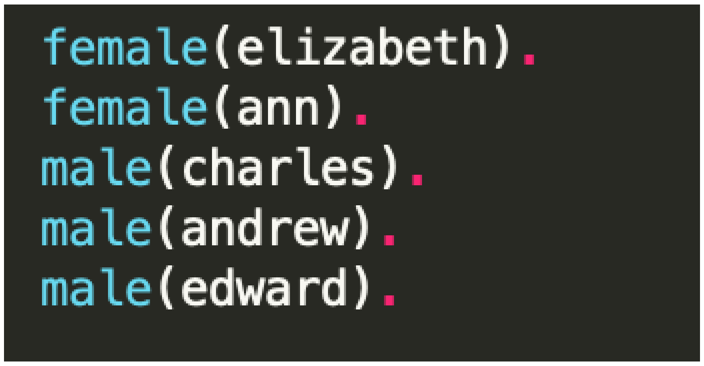
Define their relations and rules in a Prolog rule base. Hence, define the old Royal succession rule. Using this old succession rule determine the line of succession based on the information given. Do a trace to show your results. (5 marks)

Prolog Rule Base

Predicates Used	Definition
female(x)	x is female
male(x)	x is male
older(x,y)	x is a sibling of y and is born before y
is_older(x,y)	x is a sibling of y and is older than y
offspring_of(x,y)	x is the offspring of y

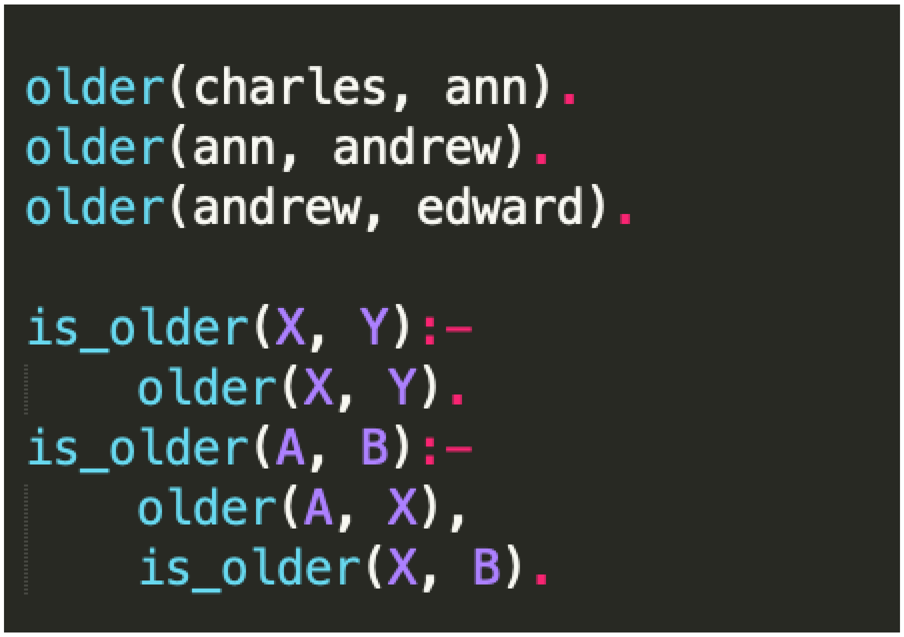
Prolog code based on each predicate

Gender



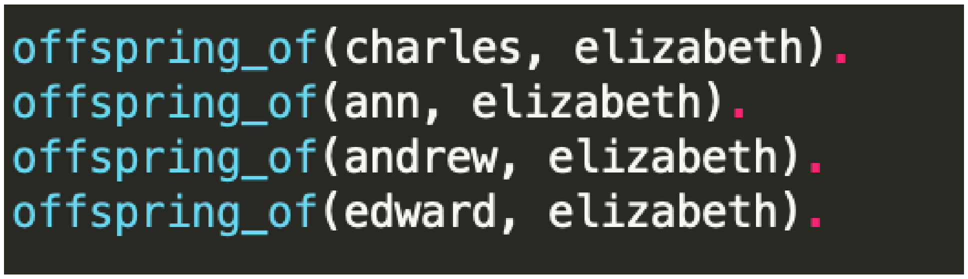
```
female(elizabeth).
female(ann).
male(charles).
male(andrew).
male(edward).
```

Order of Birth



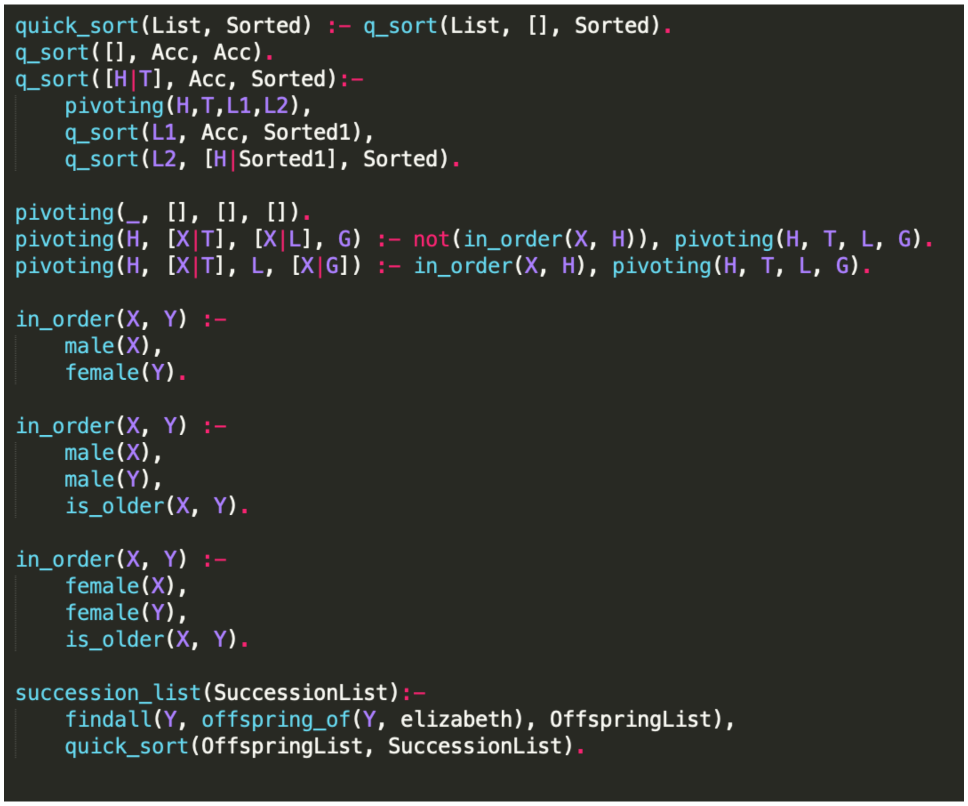
```
older(charles, ann).
older(ann, andrew).
older(andrew, edward).

is_older(X, Y):-
    older(X, Y).
is_older(A, B):-
    older(A, X),
    is_older(X, B).
```

Parent



```
offspring_of(charles, elizabeth).
offspring_of(ann, elizabeth).
offspring_of(andrew, elizabeth).
offspring_of(edward, elizabeth).
```

<u>Old Royal Succession Rule</u>



```
quick_sort(List, Sorted) :- q_sort(List, [], Sorted).
q_sort([], Acc, Acc).
q_sort([H|T], Acc, Sorted):-
    pivoting(H,T,L1,L2),
    q_sort(L1, Acc, Sorted1),
    q_sort(L2, [H|Sorted1], Sorted).

pivoting(_, [], [], []).
pivoting(H, [X|T], [X|L], G) :- not(in_order(X, H)), pivoting(H, T, L, G).
pivoting(H, [X|T], L, [X|G]) :- in_order(X, H), pivoting(H, T, L, G).

in_order(X, Y) :-
    male(X),
    female(Y).

in_order(X, Y) :-
    male(X),
    male(Y),
    is_older(X, Y).

in_order(X, Y) :-
    female(X),
    female(Y),
    is_older(X, Y).

succession_list(SuccessionList):-
    findall(Y, offspring_of(Y, elizabeth), OffspringList),
    quick_sort(OffspringList, SuccessionList).
```

<u>Determining the succession</u>

Succession_list = [charles, andrew, edward, ann]

1. Prince Charles
2. Prince Andrew
3. Prince Edward
4. Princess Ann

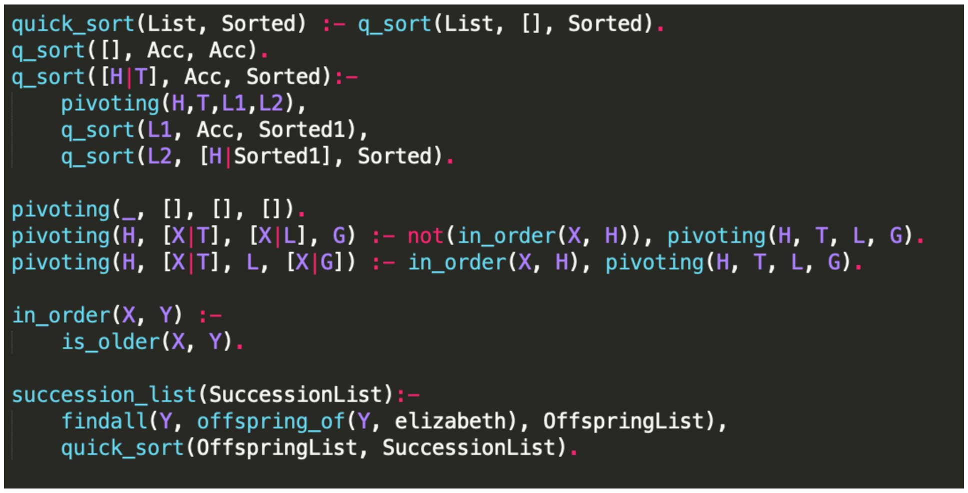
Prolog Trace

(Note: I took 3 screenshots and pasted them below, hence the spacing in between)

```
?- trace,succession_list(SuccessionList).
   Call: (11) succession_list(_11656) ? creep
^  Call: (12) findall(_12138, offspring_of(_12138, elizabeth), _12200) ? creep
   Call: (17) offspring_of(_12138, elizabeth) ? creep
   Exit: (17) offspring_of(charles, elizabeth) ? creep
   Redo: (17) offspring_of(_12138, elizabeth) ? creep
   Exit: (17) offspring_of(ann, elizabeth) ? creep
   Redo: (17) offspring_of(_12138, elizabeth) ? creep
   Exit: (17) offspring_of(andrew, elizabeth) ? creep
   Redo: (17) offspring_of(_12138, elizabeth) ? creep
   Exit: (17) offspring_of(edward, elizabeth) ? creep
^  Exit: (12) findall(_12138, user:offspring_of(_12138, elizabeth), [charles,
ann, andrew, edward]) ? creep
   Call: (12) quick_sort([charles, ann, andrew, edward], _11656) ? creep
   Call: (13) q_sort([charles, ann, andrew, edward], [], _11656) ? creep
   Call: (14) pivoting(charles, [ann, andrew, edward], _12786, _12788) ? creep
^  Call: (15) not(in_order(ann, charles)) ? creep
   Call: (16) in_order(ann, charles) ? creep
   Call: (17) male(ann) ? creep
   Fail: (17) male(ann) ? creep
   Redo: (16) in_order(ann, charles) ? creep
   Call: (17) male(ann) ? creep
   Fail: (17) male(ann) ? creep
   Redo: (16) in_order(ann, charles) ? creep
   Call: (17) female(ann) ? creep
   Exit: (17) female(ann) ? creep
   Call: (17) female(charles) ? creep
   Fail: (17) female(charles) ? creep
   Fail: (16) in_order(ann, charles) ? creep
^  Exit: (15) not(user:in_order(ann, charles)) ? creep
   Call: (15) pivoting(charles, [andrew, edward], _12776, _13466) ? creep
^  Call: (16) not(in_order(andrew, charles)) ? creep
   Call: (17) in_order(andrew, charles) ? creep
   Call: (18) male(andrew) ? creep
   Exit: (18) male(andrew) ? creep
   Call: (18) female(charles) ? creep
   Fail: (18) female(charles) ? creep
   Redo: (17) in_order(andrew, charles) ? creep
   Call: (18) male(andrew) ? creep
   Exit: (18) male(andrew) ? creep
   Call: (18) male(charles) ? creep
   Exit: (18) male(charles) ? creep
   Call: (18) is_older(andrew, charles) ? creep
   Call: (19) older(andrew, charles) ? creep
   Fail: (19) older(andrew, charles) ? creep
   Redo: (18) is_older(andrew, charles) ? creep
   Call: (19) older(andrew, _14184) ? creep
   Exit: (19) older(andrew, edward) ? creep
   Call: (19) is_older(edward, charles) ? creep
   Call: (20) older(edward, charles) ? creep
   Fail: (20) older(edward, charles) ? creep
   Redo: (19) is_older(edward, charles) ? creep
   Call: (20) older(edward, _14448) ? creep
   Fail: (20) older(edward, _14492) ? creep
   Fail: (19) is_older(edward, charles) ? creep
   Fail: (18) is_older(andrew, charles) ? creep
   Redo: (17) in_order(andrew, charles) ? creep
   Call: (18) female(andrew) ? creep
   Fail: (18) female(andrew) ? creep
   Fail: (17) in_order(andrew, charles) ? creep
^  Exit: (16) not(user:in_order(andrew, charles)) ? creep
   Call: (16) pivoting(charles, [edward], _13454, _14848) ? creep
^  Call: (17) not(in_order(edward, charles)) ? creep
   Call: (18) in_order(edward, charles) ? creep
   Call: (19) male(edward) ? creep
   Exit: (19) male(edward) ? creep
   Call: (19) female(charles) ? creep
   Fail: (19) female(charles) ? creep
   Redo: (18) in_order(edward, charles) ? creep
   Call: (19) male(edward) ? creep
   Exit: (19) male(edward) ? creep
   Call: (19) male(charles) ? creep
   Exit: (19) male(charles) ? creep
   Call: (19) is_older(edward, charles) ? creep
   Call: (20) older(edward, charles) ? creep
   Fail: (20) older(edward, charles) ? creep
   Redo: (19) is_older(edward, charles) ? creep
   Call: (20) older(edward, _15566) ? creep
   Fail: (20) older(edward, _15610) ? creep
   Fail: (19) is_older(edward, charles) ? creep
   Redo: (18) in_order(edward, charles) ? creep
   Call: (19) female(edward) ? creep
   Fail: (19) female(edward) ? creep
   Fail: (18) in_order(edward, charles) ? creep
^  Exit: (17) not(user:in_order(edward, charles)) ? creep
   Call: (17) pivoting(charles, [], _14836, _15922) ? creep
   Exit: (17) pivoting(charles, [], [], []) ? creep
   Exit: (16) pivoting(charles, [edward], [edward], []) ? creep
   Exit: (15) pivoting(charles, [andrew, edward], [andrew, edward], []) ? creep
   Exit: (14) pivoting(charles, [ann, andrew, edward], [ann, andrew, edward],
[]) ? creep
   Call: (14) q_sort([ann, andrew, edward], [], _16140) ? creep
   Call: (15) pivoting(ann, [andrew, edward], _16184, _16186) ? creep
^  Call: (16) not(in_order(andrew, ann)) ? creep
   Call: (17) in_order(andrew, ann) ? creep
   Call: (18) male(andrew) ? creep
   Exit: (18) male(andrew) ? creep
   Call: (18) female(ann) ? creep
   Exit: (18) female(ann) ? creep
   Exit: (17) in_order(andrew, ann) ? creep
^  Fail: (16) not(user:in_order(andrew, ann)) ? creep
   Redo: (15) pivoting(ann, [andrew, edward], _16598, _16600) ? creep
   Call: (16) in_order(andrew, ann) ? creep
   Call: (17) male(andrew) ? creep
   Exit: (17) male(andrew) ? creep
   Call: (17) female(ann) ? creep
   Exit: (17) female(ann) ? creep
   Exit: (16) in_order(andrew, ann) ? creep
   Call: (16) pivoting(ann, [edward], _16912, _16588) ? creep
^  Call: (17) not(in_order(edward, ann)) ? creep
   Call: (18) in_order(edward, ann) ? creep
   Call: (19) male(edward) ? creep
   Exit: (19) male(edward) ? creep
   Call: (19) female(ann) ? creep
   Exit: (19) female(ann) ? creep
   Exit: (18) in_order(edward, ann) ? creep
^  Fail: (17) not(user:in_order(edward, ann)) ? creep
   Redo: (16) pivoting(ann, [edward], _17326, _16588) ? creep
   Call: (17) in_order(edward, ann) ? creep
   Call: (18) male(edward) ? creep
   Exit: (18) male(edward) ? creep
   Call: (18) female(ann) ? creep
   Exit: (18) female(ann) ? creep
   Exit: (17) in_order(edward, ann) ? creep
   Call: (17) pivoting(ann, [], _17640, _17316) ? creep
   Exit: (17) pivoting(ann, [], [], []) ? creep
   Exit: (16) pivoting(ann, [edward], [], [edward]) ? creep
   Exit: (15) pivoting(ann, [andrew, edward], [], [andrew, edward]) ? creep
   Call: (15) q_sort([], [], _17816) ? creep
   Exit: (15) q_sort([], [], []) ? creep
   Call: (15) q_sort([andrew, edward], [ann], _17910) ? creep
   Call: (16) pivoting(andrew, [edward], _17954, _17956) ? creep
^  Call: (17) not(in_order(edward, andrew)) ? creep
   Call: (18) in_order(edward, andrew) ? creep
   Call: (19) male(edward) ? creep
   Exit: (19) male(edward) ? creep
   Call: (19) female(andrew) ? creep
   Fail: (19) female(andrew) ? creep
   Redo: (18) in_order(edward, andrew) ? creep
   Call: (19) male(edward) ? creep
   Exit: (19) male(edward) ? creep
   Call: (19) male(andrew) ? creep
   Exit: (19) male(andrew) ? creep
   Call: (19) is_older(edward, andrew) ? creep
   Call: (20) older(edward, andrew) ? creep
   Fail: (20) older(edward, andrew) ? creep
   Redo: (19) is_older(edward, andrew) ? creep
   Call: (20) older(edward, _18674) ? creep
   Fail: (20) older(edward, _18718) ? creep
   Fail: (19) is_older(edward, andrew) ? creep
   Redo: (18) in_order(edward, andrew) ? creep
   Call: (19) female(edward) ? creep
   Fail: (19) female(edward) ? creep
   Fail: (18) in_order(edward, andrew) ? creep
^  Exit: (17) not(user:in_order(edward, andrew)) ? creep
   Call: (17) pivoting(andrew, [], _17944, _19030) ? creep
   Exit: (17) pivoting(andrew, [], [], []) ? creep
   Exit: (16) pivoting(andrew, [edward], [edward], []) ? creep
   Call: (16) q_sort([edward], [ann], _19160) ? creep
   Call: (17) pivoting(edward, [], _19204, _19206) ? creep
   Exit: (17) pivoting(edward, [], [], []) ? creep
   Call: (17) q_sort([], [ann], _19292) ? creep
   Exit: (17) q_sort([], [ann], [ann]) ? creep
   Call: (17) q_sort([], [edward, ann], _19386) ? creep
   Exit: (17) q_sort([], [edward, ann], [edward, ann]) ? creep
   Exit: (16) q_sort([edward], [ann], [edward, ann]) ? creep
   Call: (16) q_sort([], [andrew, edward, ann], _19524) ? creep
   Exit: (16) q_sort([], [andrew, edward, ann], [andrew, edward, ann]) ? creep
   Exit: (15) q_sort([andrew, edward], [ann], [andrew, edward, ann]) ? creep
   Exit: (14) q_sort([ann, andrew, edward], [], [andrew, edward, ann]) ? creep
   Call: (14) q_sort([], [charles, andrew, edward, ann], _11656) ? creep
   Exit: (14) q_sort([], [charles, andrew, edward, ann], [charles, andrew,
edward, ann]) ? creep
   Exit: (13) q_sort([charles, ann, andrew, edward], [], [charles, andrew,
edward, ann]) ? creep
   Exit: (12) quick_sort([charles, ann, andrew, edward], [charles, andrew,
edward, ann]) ? creep
   Exit: (11) succession_list([charles, andrew, edward, ann]) ? creep
SuccessionList = [charles, andrew, edward, ann] .
```

2. Recently, the Royal succession rule has been modified. The throne is now passed down according to the order of birth irrespective of gender. Modify your rules and Prolog knowledge base to handle the new succession rule. Explain the necessary changes to the knowledge needed to represent the new information. Use this new succession rule to determine the new line of succession based on the same knowledge given. Show your results using a trace. (5 marks)

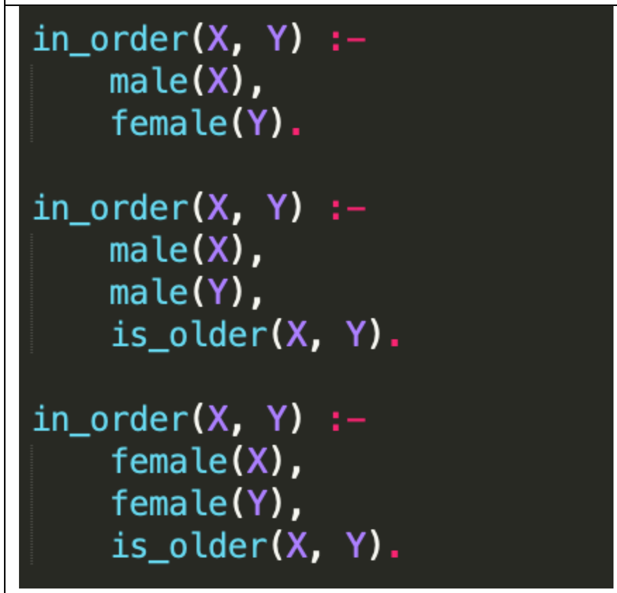
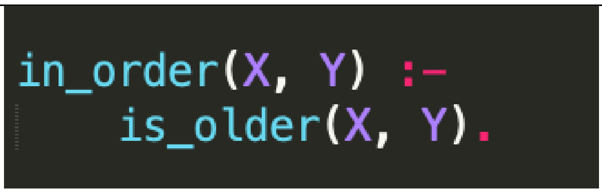
<u>Modified Royal Succession Rule</u>



```
quick_sort(List, Sorted) :- q_sort(List, [], Sorted).
q_sort([], Acc, Acc).
q_sort([H|T], Acc, Sorted):-
    pivoting(H,T,L1,L2),
    q_sort(L1, Acc, Sorted1),
    q_sort(L2, [H|Sorted1], Sorted).

pivoting(_, [], [], []).
pivoting(H, [X|T], [X|L], G) :- not(in_order(X, H)), pivoting(H, T, L, G).
pivoting(H, [X|T], L, [X|G]) :- in_order(X, H), pivoting(H, T, L, G).

in_order(X, Y) :-
    is_older(X, Y).

succession_list(SuccessionList):-
    findall(Y, offspring_of(Y, elizabeth), OffspringList),
    quick_sort(OffspringList, SuccessionList).
```

Explain the necessary changes to the knowledge needed to represent the new information.

As the Royal succession rule was modified, (the throne is passed down according to the order of birth irrespective of gender), the rules for the in_order predicate had to be modified to reflect this change. The in_order predicate is used to check if the succession list is in the correct order by checking adjacent items in the list. in_order(X, Y) is true when X is before Y in the line of succession. Before, the gender and order of birth of the offsprings were considered for determining whether in_order(X, Y) is true. After, only the order of birth was considered for determining whether in_order(X, Y) is true. The changes of the predicate are shown in the table below.

Before	After
 `in_order(X, Y) :- male(X), female(Y).` `in_order(X, Y) :- male(X), male(Y), is_older(X, Y).` `in_order(X, Y) :- female(X), female(Y), is_older(X, Y).`	 `in_order(X, Y) :- is_older(X, Y).`

Determining new line of succession

SuccessionList = [charles, ann, andrew, edward]

1. Prince Charles
2. Princess Ann
3. Prince Andrew
4. Prince Edward

Show your results using a trace.

(Note: I took 3 screenshots and pasted them below, hence the spacing in between)

```
?- trace,succession_list(SuccessionList).
   Call: (11) succession_list(_5900) ? creep
^  Call: (12) findall(_6374, offspring_of(_6374, elizabeth), _6436) ? creep
   Call: (17) offspring_of(_6374, elizabeth) ? creep
   Exit: (17) offspring_of(charles, elizabeth) ? creep
   Redo: (17) offspring_of(_6374, elizabeth) ? creep
   Exit: (17) offspring_of(ann, elizabeth) ? creep
   Redo: (17) offspring_of(_6374, elizabeth) ? creep
   Exit: (17) offspring_of(andrew, elizabeth) ? creep
   Redo: (17) offspring_of(_6374, elizabeth) ? creep
   Exit: (17) offspring_of(edward, elizabeth) ? creep
^  Exit: (12) findall(_6374, user:offspring_of(_6374, elizabeth), [charles, ann,
andrew, edward]) ? creep
   Call: (12) quick_sort([charles, ann, andrew, edward], _5900) ? creep
   Call: (13) q_sort([charles, ann, andrew, edward], [], _5900) ? creep
   Call: (14) pivoting(charles, [ann, andrew, edward], _7022, _7024) ? creep
^  Call: (15) not(in_order(ann, charles)) ? creep
   Call: (16) in_order(ann, charles) ? creep
   Call: (17) is_older(ann, charles) ? creep
   Call: (18) older(ann, charles) ? creep
   Fail: (18) older(ann, charles) ? creep
   Redo: (17) is_older(ann, charles) ? creep
   Call: (18) older(ann, _7346) ? creep
   Exit: (18) older(ann, andrew) ? creep
   Call: (18) is_older(andrew, charles) ? creep
   Call: (19) older(andrew, charles) ? creep
   Fail: (19) older(andrew, charles) ? creep
   Redo: (18) is_older(andrew, charles) ? creep
   Call: (19) older(andrew, _7610) ? creep
   Exit: (19) older(andrew, edward) ? creep
   Call: (19) is_older(edward, charles) ? creep
   Call: (20) older(edward, charles) ? creep
   Fail: (20) older(edward, charles) ? creep
   Redo: (19) is_older(edward, charles) ? creep
   Call: (20) older(edward, _7874) ? creep
   Fail: (20) older(edward, _7918) ? creep
   Fail: (19) is_older(edward, charles) ? creep
   Fail: (18) is_older(andrew, charles) ? creep
   Fail: (17) is_older(ann, charles) ? creep
   Fail: (16) in_order(ann, charles) ? creep
^  Exit: (15) not(user:in_order(ann, charles)) ? creep
   Call: (15) pivoting(charles, [andrew, edward], _7012, _8186) ? creep
^  Call: (16) not(in_order(andrew, charles)) ? creep
   Call: (17) in_order(andrew, charles) ? creep
   Call: (18) is_older(andrew, charles) ? creep
   Call: (19) older(andrew, charles) ? creep
   Fail: (19) older(andrew, charles) ? creep
   Redo: (18) is_older(andrew, charles) ? creep
   Call: (19) older(andrew, _8508) ? creep
   Exit: (19) older(andrew, edward) ? creep
   Call: (19) is_older(edward, charles) ? creep
   Call: (20) older(edward, charles) ? creep
   Fail: (20) older(edward, charles) ? creep
   Redo: (19) is_older(edward, charles) ? creep
   Call: (20) older(edward, _8772) ? creep
   Fail: (20) older(edward, _8816) ? creep
   Fail: (19) is_older(edward, charles) ? creep
   Fail: (18) is_older(andrew, charles) ? creep
   Fail: (17) in_order(andrew, charles) ? creep
^  Exit: (16) not(user:in_order(andrew, charles)) ? creep
   Call: (16) pivoting(charles, [edward], _8174, _9040) ? creep
^  Call: (17) not(in_order(edward, charles)) ? creep
   Call: (18) in_order(edward, charles) ? creep
   Call: (19) is_older(edward, charles) ? creep
   Call: (20) older(edward, charles) ? creep
   Fail: (20) older(edward, charles) ? creep
   Redo: (19) is_older(edward, charles) ? creep
   Call: (20) older(edward, _9362) ? creep
   Fail: (20) older(edward, _9406) ? creep
   Fail: (19) is_older(edward, charles) ? creep
   Fail: (18) in_order(edward, charles) ? creep
^  Exit: (17) not(user:in_order(edward, charles)) ? creep
   Call: (17) pivoting(charles, [], _9028, _9586) ? creep
   Exit: (17) pivoting(charles, [], [], []) ? creep
   Exit: (16) pivoting(charles, [edward], [edward], []) ? creep
   Exit: (15) pivoting(charles, [andrew, edward], [andrew, edward], []) ? creep
   Exit: (14) pivoting(charles, [ann, andrew, edward], [ann, andrew, edward],
[]) ? creep
   Call: (14) q_sort([ann, andrew, edward], [], _9804) ? creep
   Call: (15) pivoting(ann, [andrew, edward], _9848, _9850) ? creep
^  Call: (16) not(in_order(andrew, ann)) ? creep
   Call: (17) in_order(andrew, ann) ? creep
   Call: (18) is_older(andrew, ann) ? creep
   Call: (19) older(andrew, ann) ? creep
   Fail: (19) older(andrew, ann) ? creep
   Redo: (18) is_older(andrew, ann) ? creep
   Call: (19) older(andrew, _10172) ? creep
   Exit: (19) older(andrew, edward) ? creep
   Call: (19) is_older(edward, ann) ? creep
   Call: (20) older(edward, ann) ? creep
   Fail: (20) older(edward, ann) ? creep
   Redo: (19) is_older(edward, ann) ? creep
   Call: (20) older(edward, _10436) ? creep
   Fail: (20) older(edward, _10480) ? creep
   Fail: (19) is_older(edward, ann) ? creep
   Fail: (18) is_older(andrew, ann) ? creep
   Fail: (17) in_order(andrew, ann) ? creep
^  Exit: (16) not(user:in_order(andrew, ann)) ? creep
   Call: (16) pivoting(ann, [edward], _9838, _10704) ? creep
^  Call: (17) not(in_order(edward, ann)) ? creep
   Call: (18) in_order(edward, ann) ? creep
   Call: (19) is_older(edward, ann) ? creep
   Call: (20) older(edward, ann) ? creep
   Fail: (20) older(edward, ann) ? creep
   Redo: (19) is_older(edward, ann) ? creep
   Call: (20) older(edward, _11026) ? creep
   Fail: (20) older(edward, _11070) ? creep
   Fail: (19) is_older(edward, ann) ? creep
   Fail: (18) in_order(edward, ann) ? creep
^  Exit: (17) not(user:in_order(edward, ann)) ? creep
   Call: (17) pivoting(ann, [], _10692, _11250) ? creep
   Exit: (17) pivoting(ann, [], [], []) ? creep
   Exit: (16) pivoting(ann, [edward], [edward], []) ? creep
   Exit: (15) pivoting(ann, [andrew, edward], [andrew, edward], []) ? creep
   Call: (15) q_sort([andrew, edward], [], _11424) ? creep
   Call: (16) pivoting(andrew, [edward], _11468, _11470) ? creep
^  Call: (17) not(in_order(edward, andrew)) ? creep
   Call: (18) in_order(edward, andrew) ? creep
   Call: (19) is_older(edward, andrew) ? creep
   Call: (20) older(edward, andrew) ? creep
   Fail: (20) older(edward, andrew) ? creep
   Redo: (19) is_older(edward, andrew) ? creep
   Call: (20) older(edward, _11792) ? creep
   Fail: (20) older(edward, _11836) ? creep
   Fail: (19) is_older(edward, andrew) ? creep
   Fail: (18) in_order(edward, andrew) ? creep
^  Exit: (17) not(user:in_order(edward, andrew)) ? creep
   Call: (17) pivoting(andrew, [], _11458, _12016) ? creep
   Exit: (17) pivoting(andrew, [], [], []) ? creep
   Exit: (16) pivoting(andrew, [edward], [edward], []) ? creep
```

```
   Call: (16) q_sort([edward], [], _12146) ? creep
   Call: (17) pivoting(edward, [], _12190, _12192) ? creep
   Exit: (17) pivoting(edward, [], [], []) ? creep
   Call: (17) q_sort([], [], _12278) ? creep
   Exit: (17) q_sort([], [], []) ? creep
   Call: (17) q_sort([], [edward], _12372) ? creep
   Exit: (17) q_sort([], [edward], [edward]) ? creep
   Exit: (16) q_sort([edward], [], [edward]) ? creep
   Call: (16) q_sort([], [andrew, edward], _12510) ? creep
   Exit: (16) q_sort([], [andrew, edward], [andrew, edward]) ? creep
   Exit: (15) q_sort([andrew, edward], [], [andrew, edward]) ? creep
   Call: (15) q_sort([], [ann, andrew, edward], _12648) ? creep
   Exit: (15) q_sort([], [ann, andrew, edward], [ann, andrew, edward]) ? creep
   Exit: (14) q_sort([ann, andrew, edward], [], [ann, andrew, edward]) ? creep
   Call: (14) q_sort([], [charles, ann, andrew, edward], _5900) ? creep
   Exit: (14) q_sort([], [charles, ann, andrew, edward], [charles, ann, andrew,
edward]) ? creep
   Exit: (13) q_sort([charles, ann, andrew, edward], [], [charles, ann, andrew,
edward]) ? creep
   Exit: (12) quick_sort([charles, ann, andrew, edward], [charles, ann, andrew,
edward]) ? creep
   Exit: (11) succession_list([charles, ann, andrew, edward]) ? creep
SuccessionList = [charles, ann, andrew, edward] .
```