SC3000: Artificial Intelligence

SDAB

TANG YEQING	U2320655F

Exercise 1: **The Smart Phone Rivalry** (15 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 of appy. It is unethical for a boss to steal business from rival companies. A competitor is a rival. Smartphone technology is a business.

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

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

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

Anser:

1.
Relation:

Competitor(x, y): x is y's competitor
Rival(x,y): x is y's rival
Boss(x,y): x is y's boss
Developed(x,y): x develops y
Stole(x,y): x stoles y
Business(x): x is business
Technology(x): x is technology
SmartPhoneTech(x): x is Smart phone technology
Unethical(x): x is unethical

Object:

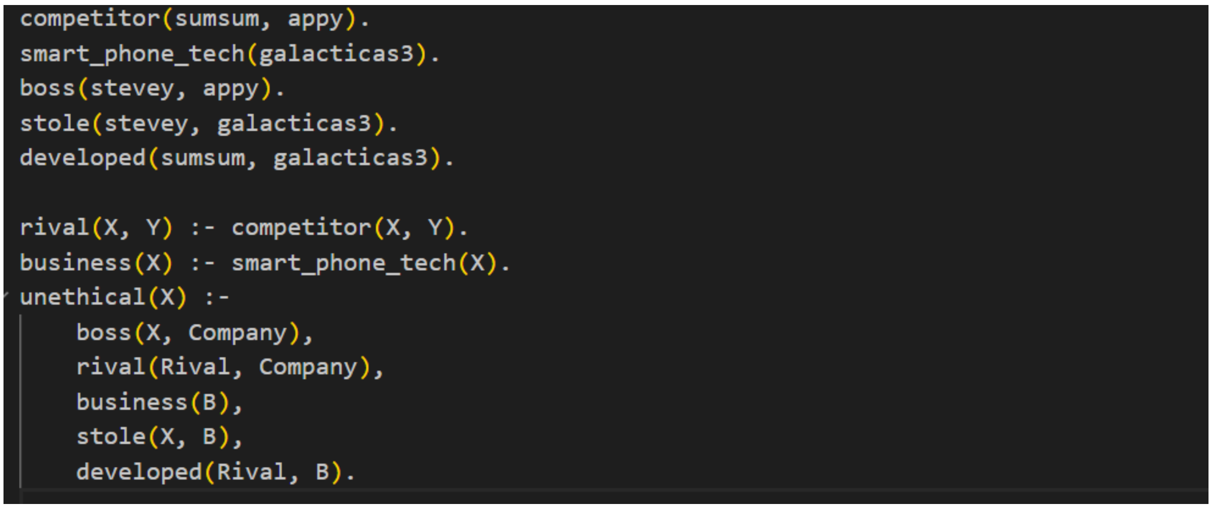
sumsum
galacticas3
appy
stevey

FOL statesments:

Competitor(sumsum,appy)
SmartPhoneTech(galacticas3)
Boss(stevey, appy)
Stole(stevey, galacticas3)

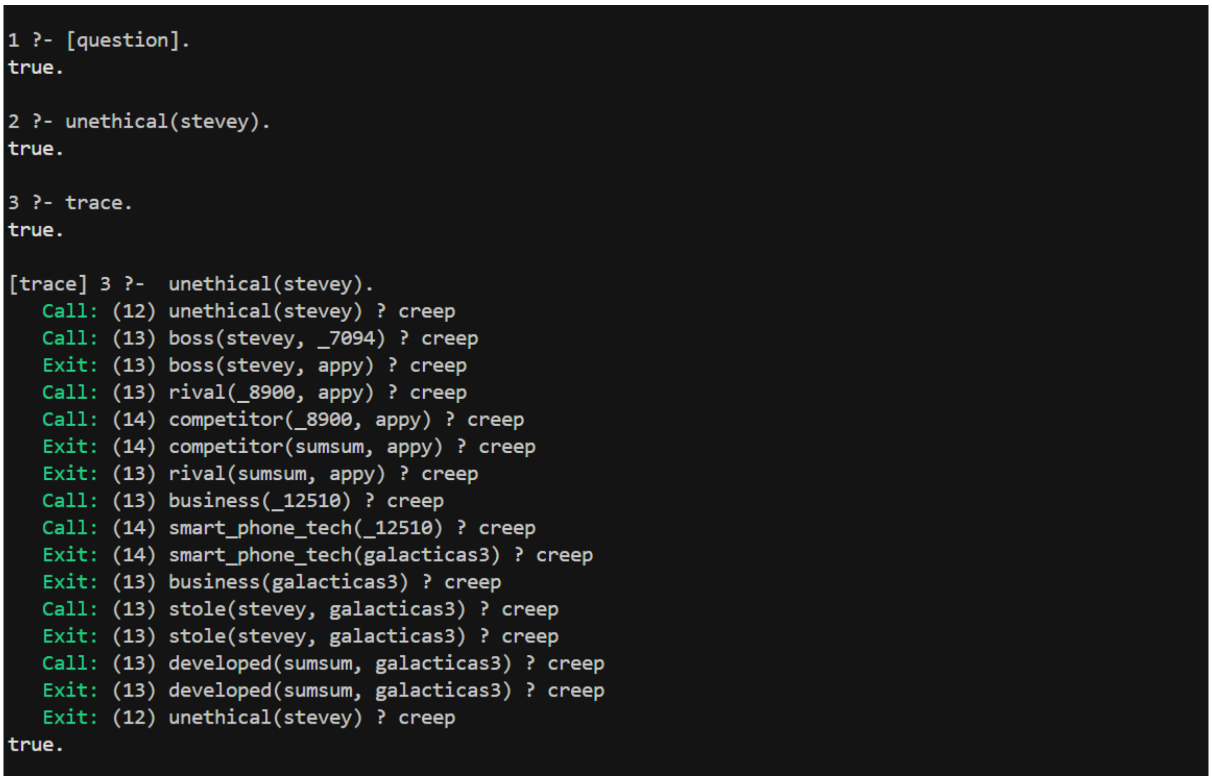
$\forall x\ \forall y\ (Competitor(x, y) \rightarrow Rival(x, y))$ #A competitor is a rival.
$\forall x\ (SmartPhoneTech(x) \rightarrow Business(x))$ #Smartphone technology is a business.
$\exists x\ \exists y\ \exists z\ \exists b\ (Boss(x, y) \land Rival(z, y) \land Business(b) \land Stole(x, b) \land Developed(z, b) \rightarrow Unethical(x))$ # It is unethical for a boss to steal business from rival companies

2.



```
competitor(sumsum, appy).
smart_phone_tech(galacticas3).
boss(stevey, appy).
stole(stevey, galacticas3).
developed(sumsum, galacticas3).

rival(X, Y) :- competitor(X, Y).
business(X) :- smart_phone_tech(X).
unethical(X) :-
    boss(X, Company),
    rival(Rival, Company),
    business(B),
    stole(X, B),
    developed(Rival, B).
```

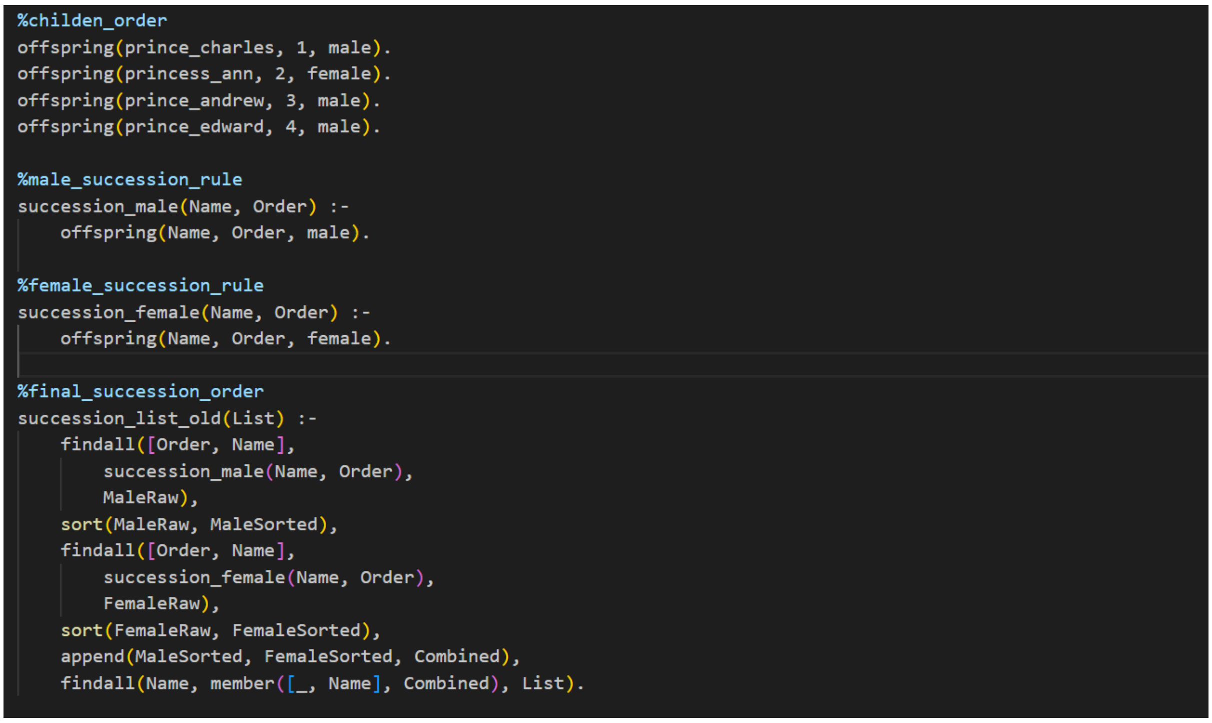
3.



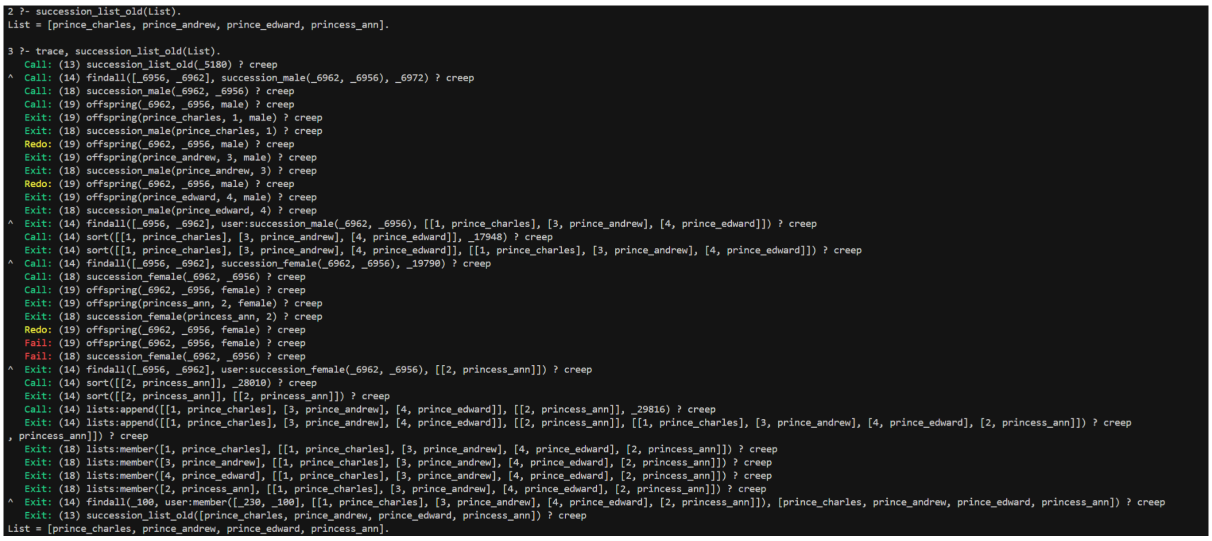
```
1 ?- [question].
true.

2 ?- unethical(stevey).
true.

3 ?- trace.
true.

[trace] 3 ?-  unethical(stevey).
   Call: (12) unethical(stevey) ? creep
   Call: (13) boss(stevey, _7094) ? creep
   Exit: (13) boss(stevey, appy) ? creep
   Call: (13) rival(_8900, appy) ? creep
   Call: (14) competitor(_8900, appy) ? creep
   Exit: (14) competitor(sumsum, appy) ? creep
   Exit: (13) rival(sumsum, appy) ? creep
   Call: (13) business(_12510) ? creep
   Call: (14) smart_phone_tech(_12510) ? creep
   Exit: (14) smart_phone_tech(galacticas3) ? creep
   Exit: (13) business(galacticas3) ? creep
   Call: (13) stole(stevey, galacticas3) ? creep
   Exit: (13) stole(stevey, galacticas3) ? creep
   Call: (13) developed(sumsum, galacticas3) ? creep
   Exit: (13) developed(sumsum, galacticas3) ? creep
   Exit: (12) unethical(stevey) ? creep
true.
```

2.**Exercise 2: The Royal Family** (10 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.

1. 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)

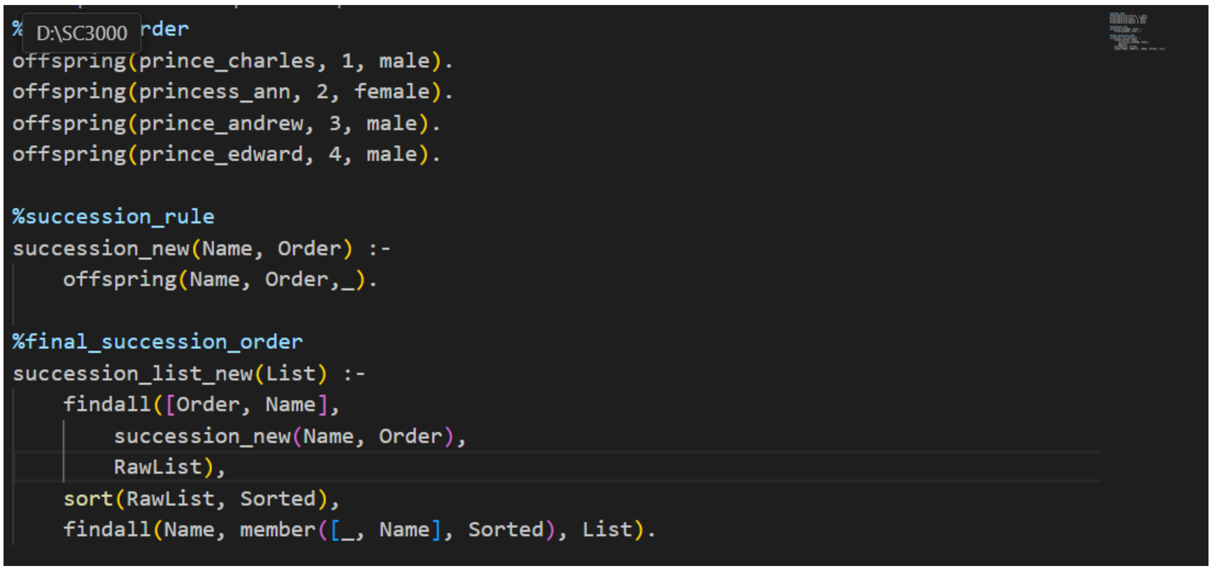


```prolog
%childen_order
offspring(prince_charles, 1, male).
offspring(princess_ann, 2, female).
offspring(prince_andrew, 3, male).
offspring(prince_edward, 4, male).

%male_succession_rule
succession_male(Name, Order) :-
    offspring(Name, Order, male).

%female_succession_rule
succession_female(Name, Order) :-
    offspring(Name, Order, female).

%final_succession_order
succession_list_old(List) :-
    findall([Order, Name],
        succession_male(Name, Order),
        MaleRaw),
    sort(MaleRaw, MaleSorted),
    findall([Order, Name],
        succession_female(Name, Order),
        FemaleRaw),
    sort(FemaleRaw, FemaleSorted),
    append(MaleSorted, FemaleSorted, Combined),
    findall(Name, member([_, Name], Combined), List).
```

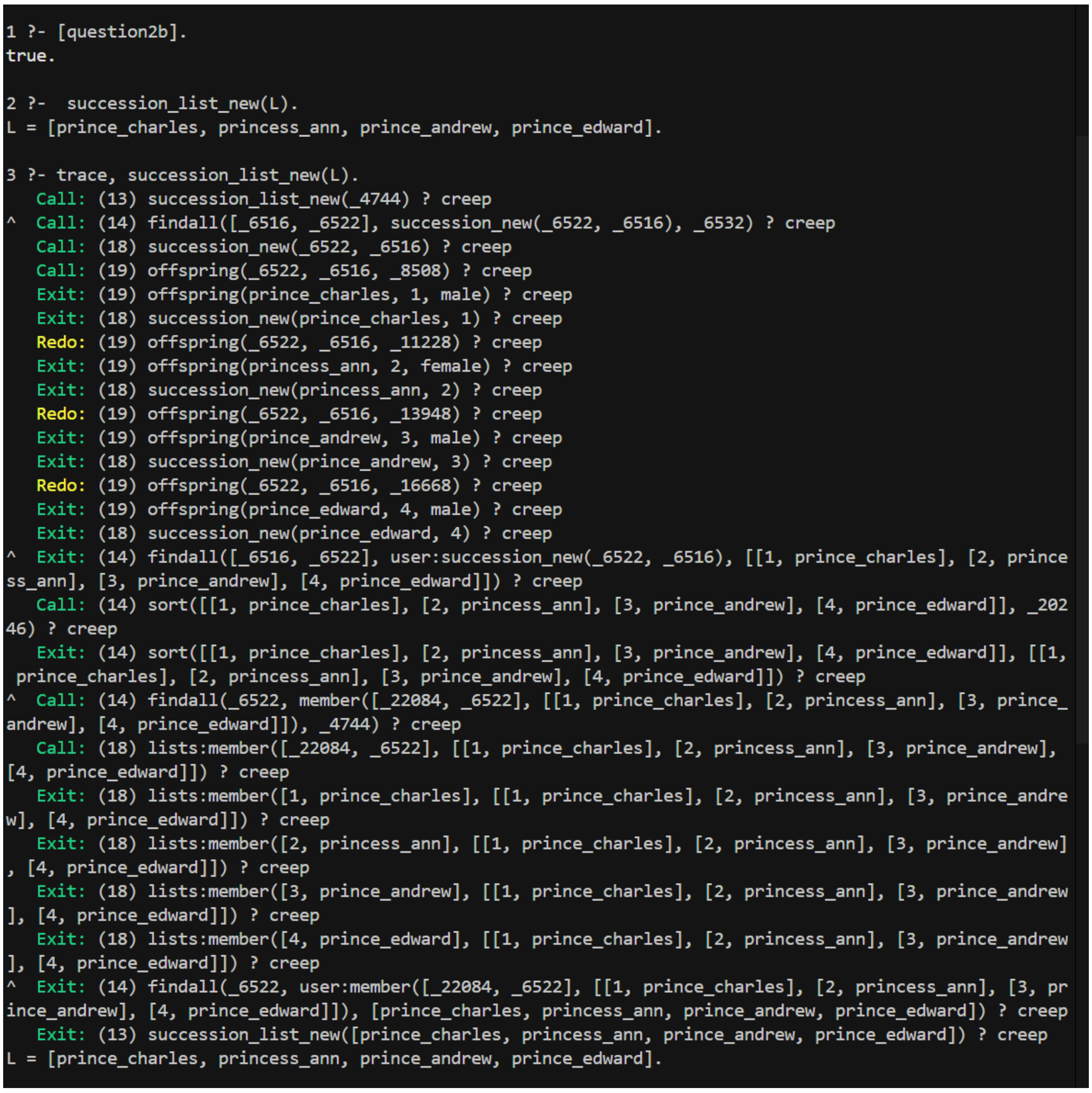


```
2 ?- succession_list_old(List).
List = [prince_charles, prince_andrew, prince_edward, princess_ann].

3 ?- trace, succession_list_old(List).
   Call: (13) succession_list_old(_5180) ? creep
^  Call: (14) findall([_6956, _6962], succession_male(_6962, _6956), _6972) ? creep
   Call: (18) succession_male(_6962, _6956) ? creep
   Call: (19) offspring(_6962, _6956, male) ? creep
   Exit: (19) offspring(prince_charles, 1, male) ? creep
   Exit: (18) succession_male(prince_charles, 1) ? creep
   Redo: (19) offspring(_6962, _6956, male) ? creep
   Exit: (19) offspring(prince_andrew, 3, male) ? creep
   Exit: (18) succession_male(prince_andrew, 3) ? creep
   Redo: (19) offspring(_6962, _6956, male) ? creep
   Exit: (19) offspring(prince_edward, 4, male) ? creep
   Exit: (18) succession_male(prince_edward, 4) ? creep
^  Exit: (14) findall([_6956, _6962], user:succession_male(_6962, _6956), [[1, prince_charles], [3, prince_andrew], [4, prince_edward]]) ? creep
   Call: (14) sort([[1, prince_charles], [3, prince_andrew], [4, prince_edward]], _17948) ? creep
   Exit: (14) sort([[1, prince_charles], [3, prince_andrew], [4, prince_edward]], [[1, prince_charles], [3, prince_andrew], [4, prince_edward]]) ? creep
^  Call: (14) findall([_6956, _6962], succession_female(_6962, _6956), _19790) ? creep
   Call: (18) succession_female(_6962, _6956) ? creep
   Call: (19) offspring(_6962, _6956, female) ? creep
   Exit: (19) offspring(princess_ann, 2, female) ? creep
   Exit: (18) succession_female(princess_ann, 2) ? creep
   Redo: (19) offspring(_6962, _6956, female) ? creep
   Fail: (19) offspring(_6962, _6956, female) ? creep
   Fail: (18) succession_female(_6962, _6956) ? creep
^  Exit: (14) findall([_6956, _6962], user:succession_female(_6962, _6956), [[2, princess_ann]]) ? creep
   Call: (14) sort([[2, princess_ann]], _28010) ? creep
   Exit: (14) sort([[2, princess_ann]], [[2, princess_ann]]) ? creep
   Call: (14) lists:append([[1, prince_charles], [3, prince_andrew], [4, prince_edward]], [[2, princess_ann]], _29816) ? creep
   Exit: (14) lists:append([[1, prince_charles], [3, prince_andrew], [4, prince_edward]], [[2, princess_ann]], [[1, prince_charles], [3, prince_andrew], [4, prince_edward], [2, princess_ann]]) ? creep
, princess_ann]]) ? creep
   Exit: (18) lists:member([1, prince_charles], [[1, prince_charles], [3, prince_andrew], [4, prince_edward], [2, princess_ann]]) ? creep
   Exit: (18) lists:member([3, prince_andrew], [[1, prince_charles], [3, prince_andrew], [4, prince_edward], [2, princess_ann]]) ? creep
   Exit: (18) lists:member([4, prince_edward], [[1, prince_charles], [3, prince_andrew], [4, prince_edward], [2, princess_ann]]) ? creep
   Exit: (18) lists:member([2, princess_ann], [[1, prince_charles], [3, prince_andrew], [4, prince_edward], [2, princess_ann]]) ? creep
^  Exit: (14) findall(_100, user:member([_230, _100], [[1, prince_charles], [3, prince_andrew], [4, prince_edward], [2, princess_ann]]), [prince_charles, prince_andrew, prince_edward, princess_ann]) ? creep
   Exit: (13) succession_list_old([prince_charles, prince_andrew, prince_edward, princess_ann]) ? creep
List = [prince_charles, prince_andrew, prince_edward, princess_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)



```
% D:\SC3000 rder
offspring(prince_charles, 1, male).
offspring(princess_ann, 2, female).
offspring(prince_andrew, 3, male).
offspring(prince_edward, 4, male).

%succession_rule
succession_new(Name, Order) :-
    offspring(Name, Order,_).

%final_succession_order
succession_list_new(List) :-
    findall([Order, Name],
        succession_new(Name, Order),
        RawList),
    sort(RawList, Sorted),
    findall(Name, member([_, Name], Sorted), List).
```



```
1 ?- [question2b].
true.

2 ?-  succession_list_new(L).
L = [prince_charles, princess_ann, prince_andrew, prince_edward].

3 ?- trace, succession_list_new(L).
   Call: (13) succession_list_new(_4744) ? creep
^  Call: (14) findall([_6516, _6522], succession_new(_6522, _6516), _6532) ? creep
   Call: (18) succession_new(_6522, _6516) ? creep
   Call: (19) offspring(_6522, _6516, _8508) ? creep
   Exit: (19) offspring(prince_charles, 1, male) ? creep
   Exit: (18) succession_new(prince_charles, 1) ? creep
   Redo: (19) offspring(_6522, _6516, _11228) ? creep
   Exit: (19) offspring(princess_ann, 2, female) ? creep
   Exit: (18) succession_new(princess_ann, 2) ? creep
   Redo: (19) offspring(_6522, _6516, _13948) ? creep
   Exit: (19) offspring(prince_andrew, 3, male) ? creep
   Exit: (18) succession_new(prince_andrew, 3) ? creep
   Redo: (19) offspring(_6522, _6516, _16668) ? creep
   Exit: (19) offspring(prince_edward, 4, male) ? creep
   Exit: (18) succession_new(prince_edward, 4) ? creep
^  Exit: (14) findall([_6516, _6522], user:succession_new(_6522, _6516), [[1, prince_charles], [2, prince
ss_ann], [3, prince_andrew], [4, prince_edward]]) ? creep
   Call: (14) sort([[1, prince_charles], [2, princess_ann], [3, prince_andrew], [4, prince_edward]], _202
46) ? creep
   Exit: (14) sort([[1, prince_charles], [2, princess_ann], [3, prince_andrew], [4, prince_edward]], [[1,
 prince_charles], [2, princess_ann], [3, prince_andrew], [4, prince_edward]]) ? creep
^  Call: (14) findall(_6522, member([_22084, _6522], [[1, prince_charles], [2, princess_ann], [3, prince_
andrew], [4, prince_edward]]), _4744) ? creep
   Call: (18) lists:member([_22084, _6522], [[1, prince_charles], [2, princess_ann], [3, prince_andrew],
[4, prince_edward]]) ? creep
   Exit: (18) lists:member([1, prince_charles], [[1, prince_charles], [2, princess_ann], [3, prince_andre
w], [4, prince_edward]]) ? creep
   Exit: (18) lists:member([2, princess_ann], [[1, prince_charles], [2, princess_ann], [3, prince_andrew]
, [4, prince_edward]]) ? creep
   Exit: (18) lists:member([3, prince_andrew], [[1, prince_charles], [2, princess_ann], [3, prince_andrew
], [4, prince_edward]]) ? creep
   Exit: (18) lists:member([4, prince_edward], [[1, prince_charles], [2, princess_ann], [3, prince_andrew
], [4, prince_edward]]) ? creep
^  Exit: (14) findall(_6522, user:member([_22084, _6522], [[1, prince_charles], [2, princess_ann], [3, pr
ince_andrew], [4, prince_edward]]), [prince_charles, princess_ann, prince_andrew, prince_edward]) ? creep
   Exit: (13) succession_list_new([prince_charles, princess_ann, prince_andrew, prince_edward]) ? creep
L = [prince_charles, princess_ann, prince_andrew, prince_edward].
```