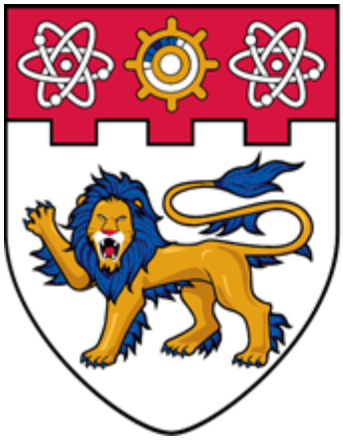


NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

CZ3005: Artificial Intelligence

LAB Assignment 2

By

Name: Datta Anusha

Matriculation Number: U1822948G

Lab Group: TS8

Date: 5th March 2021

School of Computer Science and Engineering (SCSE)

Exercise 1: The Smart Phone Rivalry

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

   sumsum, a competitor of appy [1], developed some nice smart phone technology called galacticas3 [2], all of which was stolen by stevey [3], who is a boss [4]. It is unethical for a boss to steal business from rival companies [5]. A competitor of appy is a rival [6]. Smart phone technology is business [7].

   Translating Natural Language Statements to First Order Logic (FOL)

   Key assumptions made:
   - Competitors are rivals.
   - Rivalry between companies is commutative (If company A is competitor of company B, company B is also competitor of company A).
   - Stevey is the boss of appy.
   - There may be businesses which are not smart phone technologies.

Statements	First Order Logic (FOL)
[1] sumsum, a competitor of appy	Company(sumsum) Company(appy) Competitors(sumsum, appy) ∀ a. ∀ b. (Competitors(a, b) ↔ Competitors(b, a))
[2] developed some nice smart phone technology called galacticas3	Develop(sumsum, galactica-s3) SmartPhoneTechnology(galactica-s3)
[3] all of which was stolen by stevey	Steal(stevey, galactica-s3)
[4] who is a boss	Boss(stevey, appy)

[5] It is unethical for a boss to steal business from rival companies	∀p. ∀q. ∀r. ∀s. (Boss(p, q) ∧ Steal(p, s) ∧ Business(s) ∧ Rival(q, r) ∧ Company(r) ∧ Develop(r, s) → Unethical(p))
[6] A competitor of appy is a rival	∀ x. (Competitors(x, appy) → Rival(x, appy))
[7] Smart phone technology is business	∀ x. (SmartPhoneTechnology(x) → Business(x))

2. Write these FOL statements as Prolog clauses.

```
/* FOL Statements as Prolog clauses */

/* facts */
company(sumsum).
company(appy).
smartphonetechnology(galactica-s3).

/* relationships */
competitors(sumsum, appy).
develop(sumsum, galactica-s3).
boss(stevey, appy).
steal(stevey, galactica-s3).

/* defining conjunctive/disjunctive sentences */
business(X):-
    smartphonetechnology(X).

competitors(X, Y):-
    competitors(Y, X).

rivals(X, Y):-
    competitors(X, Y).

unethical(A):-
    boss(A, B), steal(A, D), rivals(B, C), company(C), develop(C, D), business(D).
```

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

```
[trace]  ?- trace, unethical(stevey).
   Call: (11) unethical(stevey) ? creep
   Call: (12) boss(stevey, _5068) ? creep
   Exit: (12) boss(stevey, appy) ? creep
   Call: (12) steal(stevey, _5156) ? creep
   Exit: (12) steal(stevey, galactica-s3) ? creep
   Call: (12) rivals(appy, _5250) ? creep
   Call: (13) competitors(appy, _5294) ? creep
   Call: (14) competitors(_5336, appy) ? creep
   Exit: (14) competitors(sumsum, appy) ? creep
   Exit: (13) competitors(appy, sumsum) ? creep
   Exit: (12) rivals(appy, sumsum) ? creep
   Call: (12) company(sumsum) ? creep
   Exit: (12) company(sumsum) ? creep
   Call: (12) develop(sumsum, galactica-s3) ? creep
   Exit: (12) develop(sumsum, galactica-s3) ? creep
   Call: (12) business(galactica-s3) ? creep
   Call: (13) smartphonetechnology(galactica-s3) ? creep
   Exit: (13) smartphonetechnology(galactica-s3) ? creep
   Exit: (12) business(galactica-s3) ? creep
   Exit: (11) 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.

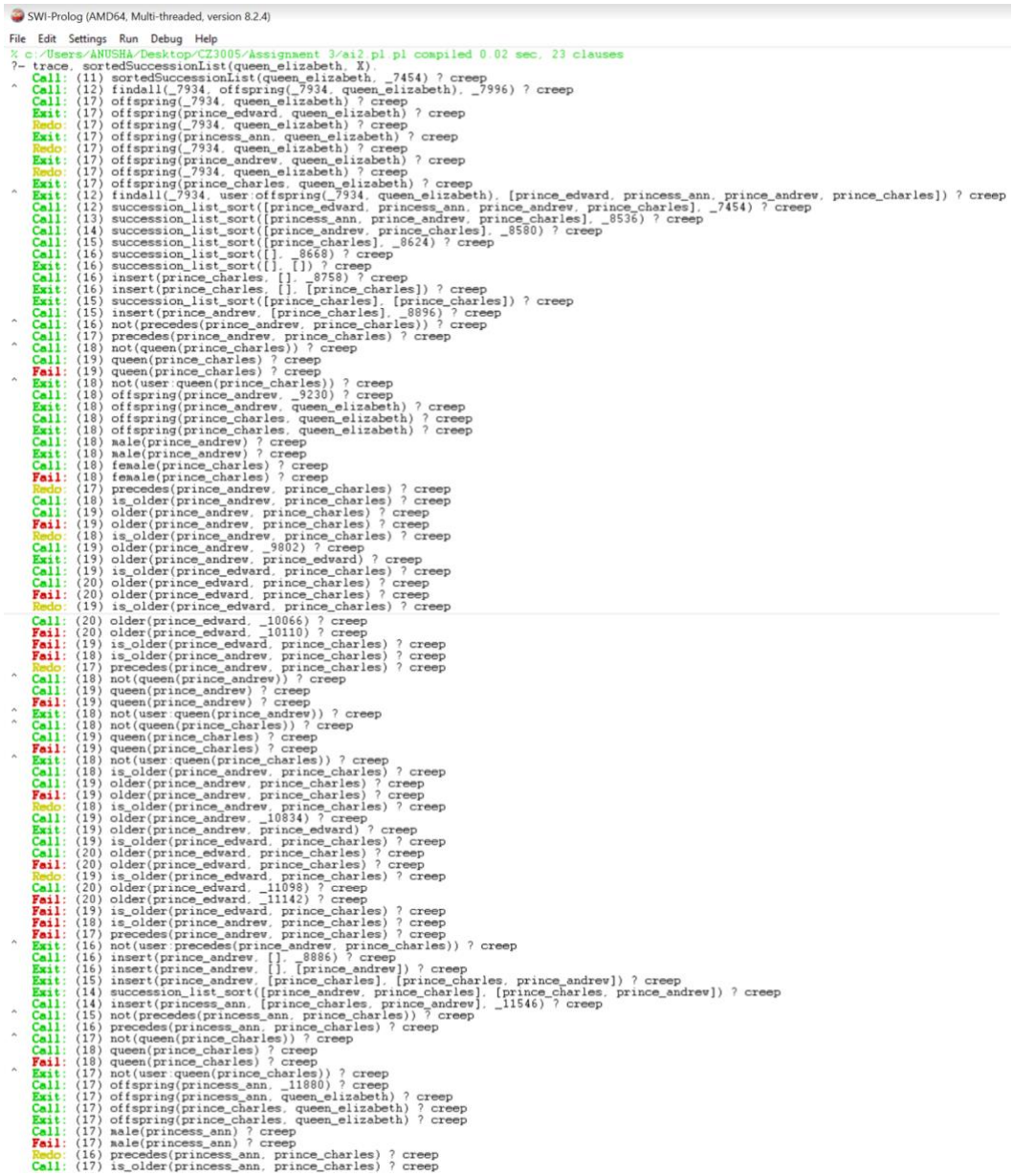
1. **Define their relations and rules in a Prolog rule base. Hence, define the old Royal succession rule. Using this old succession rule determine the line of succession based on the information given. Do a trace to show your results.**

   Succession Rules as Prolog Clauses
   The original throne succession rules are as follows:

```
/* Throne Succession Rules */

/* Rule 1: Male heir > Female heir */
precedes(X, Y):-
    not(queen(Y)),
    offspring(X, A), offspring(Y, A),
    male(X), female(Y).

/* Rule 2: Older male heir > Younger male heir */
precedes(X, Y):-
    is_older(X, Y),
    offspring(X, A), offspring(Y, A),
    male(X), male(Y).

/* Rule 3: Older female heir > Younger female heir */
precedes(X, Y):-
    not(queen(X)), not(queen(Y)),
    is_older(X, Y),
    offspring(X, A), offspring(Y, A),
    female(X), female(Y).
```

Line of Succession Trace

Final Result
X = [prince_charles, prince_andrew, prince_edward, princess_ann]



```
   Call: (18) older(princess_ann, prince_charles) ? creep
   Fail: (18) older(princess_ann, prince_charles) ? creep
   Redo: (17) is_older(princess_ann, prince_charles) ? creep
   Call: (18) older(princess_ann, _12364) ? creep
   Exit: (18) older(princess_ann, prince_andrew) ? creep
   Call: (18) is_older(prince_andrew, prince_charles) ? creep
   Call: (19) older(prince_andrew, prince_charles) ? creep
   Fail: (19) older(prince_andrew, prince_charles) ? creep
   Redo: (18) is_older(prince_andrew, prince_charles) ? creep
   Call: (19) older(prince_andrew, _12628) ? creep
   Exit: (19) older(prince_andrew, prince_edward) ? creep
   Call: (19) is_older(prince_edward, prince_charles) ? creep
   Call: (20) older(prince_edward, prince_charles) ? creep
   Fail: (20) older(prince_edward, prince_charles) ? creep
   Redo: (19) is_older(prince_edward, prince_charles) ? creep
   Call: (20) older(prince_edward, _12892) ? creep
   Fail: (20) older(prince_edward, _12936) ? creep
   Fail: (19) is_older(prince_edward, prince_charles) ? creep
   Fail: (18) is_older(prince_andrew, prince_charles) ? creep
   Fail: (17) is_older(princess_ann, prince_charles) ? creep
   Redo: (16) precedes(princess_ann, prince_charles) ? creep
^  Call: (17) not(queen(princess_ann)) ? creep
   Call: (18) queen(princess_ann) ? creep
   Fail: (18) queen(princess_ann) ? creep
^  Exit: (17) not(user:queen(princess_ann)) ? creep
^  Call: (17) not(queen(prince_charles)) ? creep
   Call: (18) queen(prince_charles) ? creep
   Fail: (18) queen(prince_charles) ? creep
^  Exit: (17) not(user:queen(prince_charles)) ? creep
   Call: (17) is_older(princess_ann, prince_charles) ? creep
   Call: (18) older(princess_ann, prince_charles) ? creep
   Fail: (18) older(princess_ann, prince_charles) ? creep
   Redo: (17) is_older(princess_ann, prince_charles) ? creep
   Call: (18) older(princess_ann, _13704) ? creep
   Exit: (18) older(princess_ann, prince_andrew) ? creep
   Call: (18) is_older(prince_andrew, prince_charles) ? creep
   Call: (19) older(prince_andrew, prince_charles) ? creep
   Fail: (19) older(prince_andrew, prince_charles) ? creep
   Redo: (18) is_older(prince_andrew, prince_charles) ? creep
   Call: (19) older(prince_andrew, _13968) ? creep
   Exit: (19) older(prince_andrew, prince_edward) ? creep
   Call: (19) is_older(prince_edward, prince_charles) ? creep
   Call: (20) older(prince_edward, prince_charles) ? creep
   Fail: (20) older(prince_edward, prince_charles) ? creep
   Redo: (19) is_older(prince_edward, prince_charles) ? creep
   Call: (20) older(prince_edward, _14232) ? creep
   Fail: (20) older(prince_edward, _14276) ? creep
   Fail: (19) is_older(prince_edward, prince_charles) ? creep
   Fail: (18) is_older(prince_andrew, prince_charles) ? creep
   Fail: (17) is_older(princess_ann, prince_charles) ? creep
   Fail: (16) precedes(princess_ann, prince_charles) ? creep
^  Exit: (15) not(user:precedes(princess_ann, prince_charles)) ? creep
   Call: (15) insert(princess_ann, [prince_andrew], _11536) ? creep
^  Call: (16) not(precedes(princess_ann, prince_andrew)) ? creep
^  Call: (17) precedes(princess_ann, prince_andrew) ? creep
^  Call: (18) not(queen(prince_andrew)) ? creep
   Call: (19) queen(prince_andrew) ? creep
   Fail: (19) queen(prince_andrew) ? creep
^  Exit: (18) not(user:queen(prince_andrew)) ? creep
   Call: (18) offspring(princess_ann, _14876) ? creep
   Exit: (18) offspring(princess_ann, queen_elizabeth) ? creep
   Call: (18) offspring(prince_andrew, queen_elizabeth) ? creep
   Exit: (18) offspring(prince_andrew, queen_elizabeth) ? creep
   Call: (18) male(princess_ann) ? creep
   Fail: (18) male(princess_ann) ? creep
   Redo: (17) precedes(princess_ann, prince_andrew) ? creep
   Call: (18) is_older(princess_ann, prince_andrew) ? creep
   Call: (19) older(princess_ann, prince_andrew) ? creep
   Exit: (19) older(princess_ann, prince_andrew) ? creep
   Exit: (18) is_older(princess_ann, prince_andrew) ? creep
   Call: (18) offspring(princess_ann, _15360) ? creep
   Exit: (18) offspring(princess_ann, queen_elizabeth) ? creep
   Call: (18) offspring(prince_andrew, queen_elizabeth) ? creep
   Exit: (18) offspring(prince_andrew, queen_elizabeth) ? creep
   Call: (18) male(princess_ann) ? creep
   Fail: (18) male(princess_ann) ? creep
   Redo: (18) is_older(princess_ann, prince_andrew) ? creep
   Call: (19) older(princess_ann, _15668) ? creep
   Exit: (19) older(princess_ann, prince_andrew) ? creep
   Call: (19) is_older(prince_andrew, prince_andrew) ? creep
   Call: (20) older(prince_andrew, prince_andrew) ? creep
   Fail: (20) older(prince_andrew, prince_andrew) ? creep
   Redo: (19) is_older(prince_andrew, prince_andrew) ? creep
   Call: (20) older(prince_andrew, _15932) ? creep
   Exit: (20) older(prince_andrew, prince_edward) ? creep
   Call: (20) is_older(prince_edward, prince_andrew) ? creep
   Call: (21) older(prince_edward, prince_andrew) ? creep
   Fail: (21) older(prince_edward, prince_andrew) ? creep
   Redo: (20) is_older(prince_edward, prince_andrew) ? creep
   Call: (21) older(prince_edward, _16196) ? creep
   Fail: (21) older(prince_edward, _16240) ? creep
   Fail: (20) is_older(prince_edward, prince_andrew) ? creep
   Fail: (19) is_older(prince_andrew, prince_andrew) ? creep
   Fail: (18) is_older(princess_ann, prince_andrew) ? creep
   Redo: (17) precedes(princess_ann, prince_andrew) ? creep
^  Call: (18) not(queen(princess_ann)) ? creep
   Call: (19) queen(princess_ann) ? creep
   Fail: (19) queen(princess_ann) ? creep
^  Exit: (18) not(user:queen(princess_ann)) ? creep
^  Call: (18) not(queen(prince_andrew)) ? creep
   Call: (19) queen(prince_andrew) ? creep
   Fail: (19) queen(prince_andrew) ? creep
^  Exit: (18) not(user:queen(prince_andrew)) ? creep
   Call: (18) is_older(princess_ann, prince_andrew) ? creep
   Call: (19) older(princess_ann, prince_andrew) ? creep
   Exit: (19) older(princess_ann, prince_andrew) ? creep
   Exit: (18) is_older(princess_ann, prince_andrew) ? creep
   Call: (18) offspring(princess_ann, _17008) ? creep
   Exit: (18) offspring(princess_ann, queen_elizabeth) ? creep
   Call: (18) offspring(prince_andrew, queen_elizabeth) ? creep
   Exit: (18) offspring(prince_andrew, queen_elizabeth) ? creep
   Call: (18) female(princess_ann) ? creep
   Exit: (18) female(princess_ann) ? creep
   Call: (18) female(prince_andrew) ? creep
   Fail: (18) female(prince_andrew) ? creep
   Redo: (18) is_older(princess_ann, prince_andrew) ? creep
   Call: (19) older(princess_ann, _17404) ? creep
   Exit: (19) older(princess_ann, prince_andrew) ? creep
   Call: (19) is_older(prince_andrew, prince_andrew) ? creep
   Call: (20) older(prince_andrew, prince_andrew) ? creep
   Fail: (20) older(prince_andrew, prince_andrew) ? creep
   Redo: (19) is_older(prince_andrew, prince_andrew) ? creep
   Call: (20) older(prince_andrew, _17668) ? creep
   Exit: (20) older(prince_andrew, prince_edward) ? creep
   Call: (20) is_older(prince_edward, prince_andrew) ? creep
   Call: (21) older(prince_edward, prince_andrew) ? creep
   Fail: (21) older(prince_edward, prince_andrew) ? creep
   Redo: (20) is_older(prince_edward, prince_andrew) ? creep
   Call: (21) older(prince_edward, _17932) ? creep
   Fail: (21) older(prince_edward, _17976) ? creep
   Fail: (20) is_older(prince_edward, prince_andrew) ? creep
   Fail: (19) is_older(prince_andrew, prince_andrew) ? creep
   Fail: (18) is_older(princess_ann, prince_andrew) ? creep
   Fail: (17) precedes(princess_ann, prince_andrew) ? creep
^  Exit: (16) not(user:precedes(princess_ann, prince_andrew)) ? creep
   Call: (16) insert(princess_ann, [], _14532) ? creep
   Exit: (16) insert(princess_ann, [], [princess_ann]) ? creep
   Exit: (15) insert(princess_ann, [prince_andrew], [prince_andrew, princess_ann]) ? creep
   Exit: (14) insert(princess_ann, [prince_charles, prince_andrew], [prince_charles, prince_andrew, princess_ann]) ? creep
   Exit: (13) succession_list_sort([princess_ann, prince_andrew, prince_charles], [prince_charles, prince_andrew, princess_ann]) ? creep
   Call: (13) insert(prince_edward, [prince_charles, prince_andrew, princess_ann], _7454) ? creep
^  Call: (14) not(precedes(prince_edward, prince_charles)) ? creep
^  Call: (15) precedes(prince_edward, prince_charles) ? creep
^  Call: (16) not(queen(prince_charles)) ? creep
```

```
   Fail: (17) queen(prince_charles) ? creep
^  Exit: (16) not(user:queen(prince_charles)) ? creep
   Call: (16) offspring(prince_edward, _18802) ? creep
   Exit: (16) offspring(prince_edward, queen_elizabeth) ? creep
   Call: (16) offspring(prince_charles, queen_elizabeth) ? creep
   Exit: (16) offspring(prince_charles, queen_elizabeth) ? creep
   Call: (16) male(prince_edward) ? creep
   Exit: (16) male(prince_edward) ? creep
   Call: (16) female(prince_charles) ? creep
   Fail: (16) female(prince_charles) ? creep
   Redo: (15) precedes(prince_edward, prince_charles) ? creep
   Call: (16) is_older(prince_edward, prince_charles) ? creep
   Call: (17) older(prince_edward, prince_charles) ? creep
   Fail: (17) older(prince_edward, prince_charles) ? creep
   Redo: (16) is_older(prince_edward, prince_charles) ? creep
   Call: (17) older(prince_edward, _19374) ? creep
   Fail: (17) older(prince_edward, _19418) ? creep
   Fail: (16) is_older(prince_edward, prince_charles) ? creep
   Redo: (15) precedes(prince_edward, prince_charles) ? creep
^  Call: (16) not(queen(prince_edward)) ? creep
   Call: (17) queen(prince_edward) ? creep
   Fail: (17) queen(prince_edward) ? creep
^  Exit: (16) not(user:queen(prince_edward)) ? creep
^  Call: (16) not(queen(prince_charles)) ? creep
   Call: (17) queen(prince_charles) ? creep
   Fail: (17) queen(prince_charles) ? creep
^  Exit: (16) not(user:queen(prince_charles)) ? creep
   Call: (16) is_older(prince_edward, prince_charles) ? creep
   Call: (17) older(prince_edward, prince_charles) ? creep
   Fail: (17) older(prince_edward, prince_charles) ? creep
   Redo: (16) is_older(prince_edward, prince_charles) ? creep
   Call: (17) older(prince_edward, _20098) ? creep
   Fail: (17) older(prince_edward, _20142) ? creep
   Fail: (16) is_older(prince_edward, prince_charles) ? creep
   Fail: (15) precedes(prince_edward, prince_charles) ? creep
^  Exit: (14) not(user:precedes(prince_edward, prince_charles)) ? creep
   Call: (14) insert(prince_edward, [prince_andrew, princess_ann], _18458) ? creep
^  Call: (15) not(precedes(prince_edward, prince_andrew)) ? creep
   Call: (16) precedes(prince_edward, prince_andrew) ? creep
^  Call: (17) not(queen(prince_andrew)) ? creep
   Call: (18) queen(prince_andrew) ? creep
   Fail: (18) queen(prince_andrew) ? creep
^  Exit: (17) not(user:queen(prince_andrew)) ? creep
   Call: (17) offspring(prince_edward, _20654) ? creep
   Exit: (17) offspring(prince_edward, queen_elizabeth) ? creep
   Call: (17) offspring(prince_andrew, queen_elizabeth) ? creep
   Exit: (17) offspring(prince_andrew, queen_elizabeth) ? creep
   Call: (17) male(prince_edward) ? creep
   Fail: (18) older(prince_edward, prince_andrew) ? creep
   Redo: (17) is_older(prince_edward, prince_andrew) ? creep
   Call: (18) older(prince_edward, _21226) ? creep
   Fail: (18) older(prince_edward, _21270) ? creep
   Fail: (17) is_older(prince_edward, prince_andrew) ? creep
   Redo: (16) precedes(prince_edward, prince_andrew) ? creep
^  Call: (17) not(queen(prince_edward)) ? creep
   Call: (18) queen(prince_edward) ? creep
   Fail: (18) queen(prince_edward) ? creep
^  Exit: (17) not(user:queen(prince_edward)) ? creep
^  Call: (17) not(queen(prince_andrew)) ? creep
   Call: (18) queen(prince_andrew) ? creep
   Fail: (18) queen(prince_andrew) ? creep
^  Exit: (17) not(user:queen(prince_andrew)) ? creep
   Call: (17) is_older(prince_edward, prince_andrew) ? creep
   Call: (18) older(prince_edward, prince_andrew) ? creep
   Fail: (18) older(prince_edward, prince_andrew) ? creep
   Redo: (17) is_older(prince_edward, prince_andrew) ? creep
   Call: (18) older(prince_edward, _21950) ? creep
   Fail: (18) older(prince_edward, _21994) ? creep
   Fail: (17) is_older(prince_edward, prince_andrew) ? creep
   Fail: (16) precedes(prince_edward, prince_andrew) ? creep
^  Exit: (15) not(user:precedes(prince_edward, prince_andrew)) ? creep
   Call: (15) insert(prince_edward, [princess_ann], _20310) ? creep
^  Call: (16) not(precedes(prince_edward, princess_ann)) ? creep
   Call: (17) precedes(prince_edward, princess_ann) ? creep
^  Call: (18) not(queen(princess_ann)) ? creep
   Call: (19) queen(princess_ann) ? creep
   Fail: (19) queen(princess_ann) ? creep
^  Exit: (18) not(user:queen(princess_ann)) ? creep
   Call: (18) offspring(prince_edward, _22506) ? creep
   Exit: (18) offspring(prince_edward, queen_elizabeth) ? creep
   Call: (18) offspring(princess_ann, queen_elizabeth) ? creep
   Exit: (18) offspring(princess_ann, queen_elizabeth) ? creep
   Call: (18) male(prince_edward) ? creep
   Exit: (18) male(prince_edward) ? creep
   Call: (18) female(princess_ann) ? creep
   Exit: (18) female(princess_ann) ? creep
   Exit: (17) precedes(prince_edward, princess_ann) ? creep
^  Fail: (16) not(user:precedes(prince_edward, princess_ann)) ? creep
   Redo: (15) insert(prince_edward, [princess_ann], _20310) ? creep
   Exit: (15) insert(prince_edward, [princess_ann], [prince_edward, princess_ann]) ? creep
   Exit: (14) insert(prince_edward, [prince_andrew, princess_ann], [prince_andrew, prince_edward, princess_ann]) ? creep
   Exit: (13) insert(prince_edward, [prince_charles, prince_andrew, princess_ann], [prince_charles, prince_andrew, prince_edward, princess_
   Exit: (12) succession_list_sort([prince_edward, princess_ann, prince_andrew, prince_charles], [prince_charles, prince_andrew, prince_edw
n]) ? creep
   Exit: (11) sortedSuccessionList(queen_elizabeth, [prince_charles, prince_andrew, prince_edward, princess_ann]) ? creep
X = [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.**

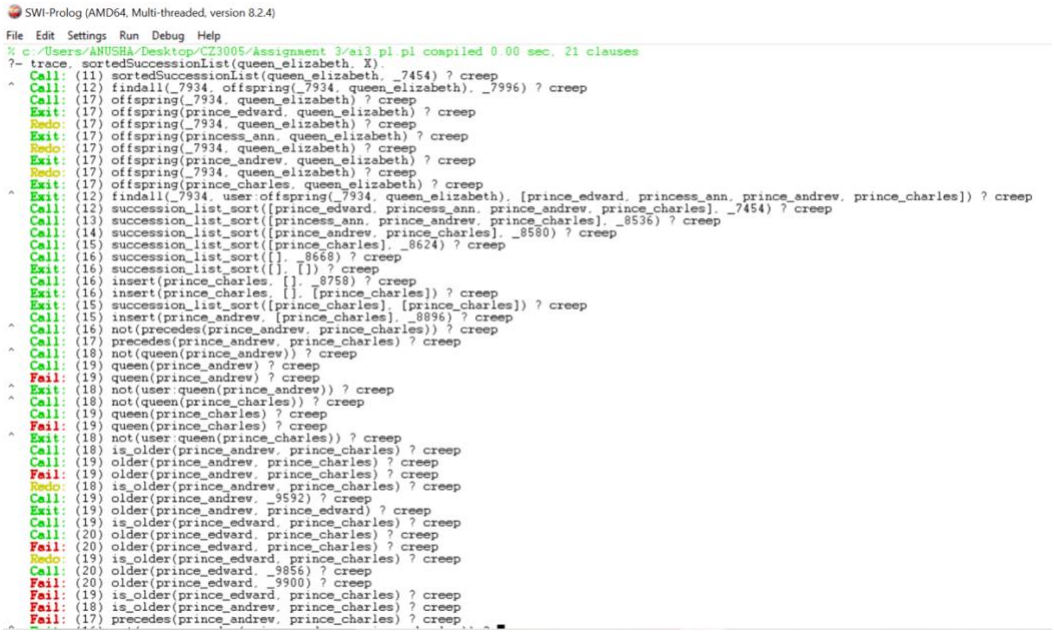
Succession Rules as Prolog clauses

As throne succession rules have been updated to disregard gender, the new succession rules shall be as follows:

```
/* Throne Succession Rule */

/* Older child > Younger Child */
precedes(X, Y):-
    not(queen(X)), not(queen(Y)),
    is_older(X, Y),
    offspring(X, A), offspring(Y, A).
```

Line of Succession Trace

Final Result
X = [prince_charles, princess_ann, prince_andrew, prince_edward]



```
^  Exit: (16) not(user:precedes(prince_andrew, prince_charles)) ? creep
   Call: (16) insert(prince_andrew, [], _8886) ? creep
   Exit: (16) insert(prince_andrew, [], [prince_andrew]) ? creep
   Exit: (15) insert(prince_andrew, [prince_charles], [prince_charles, prince_andrew]) ? creep
   Exit: (14) succession_list_sort([prince_andrew, prince_charles], [prince_charles, prince_andrew]) ? creep
   Call: (14) insert(princess_ann, [prince_charles, prince_andrew], _10304) ? creep
^  Call: (15) not(precedes(princess_ann, prince_charles)) ? creep
   Call: (16) precedes(princess_ann, prince_charles) ? creep
^  Call: (17) not(queen(princess_ann)) ? creep
   Call: (18) queen(princess_ann) ? creep
   Fail: (18) queen(princess_ann) ? creep
^  Exit: (17) not(user:queen(princess_ann)) ? creep
^  Call: (17) not(queen(prince_charles)) ? creep
   Call: (18) queen(prince_charles) ? creep
   Fail: (18) queen(prince_charles) ? creep
^  Exit: (17) not(user:queen(prince_charles)) ? creep
   Call: (17) is_older(princess_ann, prince_charles) ? creep
   Call: (18) older(princess_ann, prince_charles) ? creep
   Fail: (18) older(princess_ann, prince_charles) ? creep
   Redo: (17) is_older(princess_ann, prince_charles) ? creep
   Call: (18) older(princess_ann, _11000) ? creep
   Exit: (18) older(princess_ann, prince_andrew) ? creep
   Call: (18) is_older(prince_andrew, prince_charles) ? creep
   Call: (19) older(prince_andrew, prince_charles) ? creep
   Fail: (19) older(prince_andrew, prince_charles) ? creep
   Redo: (18) is_older(prince_andrew, prince_charles) ? creep
   Call: (19) older(prince_andrew, _11264) ? creep
   Exit: (19) older(prince_andrew, prince_edward) ? creep
   Call: (19) is_older(prince_edvard, prince_charles) ? creep
   Call: (20) older(prince_edward, prince_charles) ? creep
   Fail: (20) older(prince_edward, prince_charles) ? creep
   Redo: (19) is_older(prince_edvard, prince_charles) ? creep
   Call: (20) older(prince_edward, _11528) ? creep
   Fail: (20) older(prince_edward, _11572) ? creep
   Fail: (19) is_older(prince_edvard, prince_charles) ? creep
   Fail: (18) is_older(prince_andrew, prince_charles) ? creep
   Fail: (17) is_older(princess_ann, prince_charles) ? creep
   Fail: (16) precedes(princess_ann, prince_charles) ? creep
^  Exit: (15) not(user:precedes(princess_ann, prince_charles)) ? creep
   Call: (15) insert(princess_ann, [prince_andrew], _10294) ? creep
^  Call: (16) not(precedes(princess_ann, prince_andrew)) ? creep
   Call: (16) precedes(princess_ann, prince_andrew) ? creep
^  Call: (18) not(queen(princess_ann)) ? creep
   Call: (19) queen(princess_ann) ? creep
   Fail: (19) queen(princess_ann) ? creep
^  Exit: (18) not(user:queen(princess_ann)) ? creep
^  Call: (18) not(queen(prince_andrew)) ? creep
   Call: (19) queen(prince_andrew) ? creep
   Fail: (19) queen(prince_andrew) ? creep
^  Exit: (18) not(user:queen(prince_andrew)) ? creep
   Call: (18) is_older(princess_ann, prince_andrew) ? creep
   Call: (19) older(princess_ann, prince_andrew) ? creep
   Exit: (19) older(princess_ann, prince_andrew) ? creep
   Exit: (18) is_older(princess_ann, prince_andrew) ? creep
   Call: (18) offspring(princess_ann, _12534) ? creep
   Exit: (18) offspring(princess_ann, queen_elizabeth) ? creep
   Call: (18) offspring(prince_andrew, queen_elizabeth) ? creep
   Exit: (18) offspring(prince_andrew, queen_elizabeth) ? creep
   Exit: (17) precedes(princess_ann, prince_andrew) ? creep
^  Fail: (16) not(user:precedes(princess_ann, prince_andrew)) ? creep
   Redo: (15) insert(princess_ann, [prince_andrew], _10294) ? creep
   Exit: (15) insert(princess_ann, [prince_andrew], [princess_ann, prince_andrew]) ? creep
   Exit: (14) insert(princess_ann, [prince_charles, prince_andrew], [prince_charles, princess_ann, prince_andrew]) ? creep
   Exit: (13) succession_list_sort([princess_ann, prince_andrew, prince_charles], [prince_charles, princess_ann, prince_andrew]) ? creep
   Call: (13) insert(prince_edward, [prince_charles, princess_ann, prince_andrew], _7454) ? creep
^  Call: (14) not(precedes(prince_edward, prince_charles)) ? creep
   Call: (15) precedes(prince_edward, prince_charles) ? creep
^  Call: (16) not(queen(prince_edward)) ? creep
   Call: (17) queen(prince_edward) ? creep
   Fail: (17) queen(prince_edward) ? creep
^  Exit: (16) not(user:queen(prince_edward)) ? creep
^  Call: (16) not(queen(prince_charles)) ? creep
   Call: (17) queen(prince_charles) ? creep
   Fail: (17) queen(prince_charles) ? creep
^  Exit: (16) not(user:queen(prince_charles)) ? creep
   Call: (16) is_older(prince_edward, prince_charles) ? creep
   Call: (17) older(prince_edward, prince_charles) ? creep
   Fail: (17) older(prince_edward, prince_charles) ? creep
   Redo: (16) is_older(prince_edvard, prince_charles) ? creep
   Call: (17) older(prince_edward, _13678) ? creep
   Fail: (17) older(prince_edward, _13722) ? creep
   Fail: (16) is_older(prince_edvard, prince_charles) ? creep
   Fail: (15) precedes(prince_edvard, prince_charles) ? creep
^  Exit: (14) not(user:precedes(prince_edvard, prince_charles)) ? creep
   Call: (14) insert(prince_edvard, [princess_ann, prince_andrew], _12972) ? creep
^  Call: (15) not(precedes(prince_edvard, princess_ann)) ? creep
   Call: (16) precedes(prince_edvard, princess_ann) ? creep
^  Call: (17) not(queen(prince_edvard)) ? creep
   Call: (18) queen(prince_edvard) ? creep
   Fail: (18) queen(prince_edvard) ? creep
^  Exit: (17) not(user:queen(prince_edvard)) ? creep
^  Call: (17) not(queen(princess_ann)) ? creep
   Call: (18) queen(princess_ann) ? creep
   Fail: (18) queen(princess_ann) ? creep
^  Exit: (17) not(user:queen(princess_ann)) ? creep
   Call: (17) is_older(prince_edvard, princess_ann) ? creep
   Call: (18) older(prince_edvard, princess_ann) ? creep
   Fail: (18) older(prince_edvard, princess_ann) ? creep
   Redo: (17) is_older(prince_edvard, princess_ann) ? creep
   Call: (18) older(prince_edvard, _14596) ? creep
   Fail: (18) older(prince_edvard, _14640) ? creep
   Fail: (17) is_older(prince_edvard, princess_ann) ? creep
   Fail: (16) precedes(prince_edvard, princess_ann) ? creep
^  Exit: (15) not(user:precedes(prince_edvard, princess_ann)) ? creep
   Call: (15) insert(prince_edvard, [prince_andrew], _13890) ? creep
^  Call: (16) not(precedes(prince_edvard, prince_andrew)) ? creep
   Call: (17) precedes(prince_edvard, prince_andrew) ? creep
^  Call: (18) not(queen(prince_edvard)) ? creep
   Call: (19) queen(prince_edvard) ? creep
   Fail: (19) queen(prince_edvard) ? creep
^  Exit: (18) not(user:queen(prince_edvard)) ? creep
^  Call: (18) not(queen(prince_andrew)) ? creep
   Call: (19) queen(prince_andrew) ? creep
   Fail: (19) queen(prince_andrew) ? creep
^  Exit: (18) not(user:queen(prince_andrew)) ? creep
   Call: (18) is_older(prince_edvard, prince_andrew) ? creep
   Call: (19) older(prince_edvard, prince_andrew) ? creep
   Fail: (19) older(prince_edvard, prince_andrew) ? creep
   Redo: (18) is_older(prince_edvard, prince_andrew) ? creep
   Call: (19) older(prince_edvard, _15514) ? creep
   Fail: (19) older(prince_edvard, _15558) ? creep
   Fail: (18) is_older(prince_edvard, prince_andrew) ? creep
   Fail: (17) precedes(prince_edvard, prince_andrew) ? creep
^  Exit: (16) not(user:precedes(prince_edvard, prince_andrew)) ? creep
   Call: (16) insert(prince_edvard, [], _14808) ? creep
   Exit: (16) insert(prince_edvard, [], [prince_edvard]) ? creep
   Exit: (15) insert(prince_edvard, [prince_andrew], [prince_andrew, prince_edvard]) ? creep
   Exit: (14) insert(prince_edvard, [princess_ann, prince_andrew], [princess_ann, prince_andrew, prince_edvard]) ? creep
   Exit: (13) insert(prince_edvard, [prince_charles, princess_ann, prince_andrew], [prince_charles, princess_ann, prince_andrew, prince_edvard]) ? creep
   Exit: (12) succession_list_sort([prince_edvard, princess_ann, prince_andrew, prince_charles], [prince_charles, princess_ann, prince_andrew, prince_edvar
d]) ? creep
   Exit: (11) sortedSuccessionList(queen_elizabeth, [prince_charles, princess_ann, prince_andrew, prince_edvard]) ? creep
X = [prince_charles, princess_ann, prince_andrew, prince_edvard].
```