Group Members	Isaac Lim Jia Le	U2321202C
Lab group	SDAD	

Question 1:

sumsum, a competitor of appy, developed some nice smartphone 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. Smartphone technology is a business.

1. Translate the natural language statements above describing the dealing within the Smartphone 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: developed(Sumsum, Galactica-s3)

Statement: galactica-s3 was stolen by stevey
FOL: stolen(Galactica-s3, Stevey)

Statement: stevey is a boss of appy
FOL: boss(Stevey, Appy)

Statement: Unethical for a boss to steal business from rival companies
FOL: ∀x, ∀company, ∀competitor, ∀technology boss(x, company) ∧ competitor(competitor, company) ∧ developed(competitor, technology) ∧ stolen(x, technology) ⇒ unethical(x)

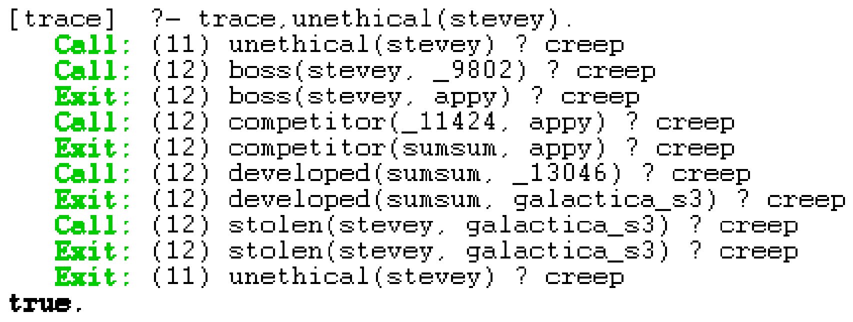
Statement: A competitor is a rival
FOL: ∀company, ∀competitor competitor(competitor, company) ⇒ rival(competitor, company)

Statement: Smartphone technology is a business
FOL: business(smart_phone_technology)

2. Write these FOL statements as Prolog clauses.

```
competitor(sumsum, appy).
developed(sumsum, galactica_s3).
stolen(stevey, galactica_s3).
boss(stevey, appy).
unethical(X) :- boss(X, Company), competitor(Competitor, Company),
developed(Competitor, Technology), stolen(X, Technology).
```

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, _9802) ? creep
   Exit: (12) boss(stevey, appy) ? creep
   Call: (12) competitor(_11424, appy) ? creep
   Exit: (12) competitor(sumsum, appy) ? creep
   Call: (12) developed(sumsum, _13046) ? creep
   Exit: (12) developed(sumsum, galactica_s3) ? creep
   Call: (12) stolen(stevey, galactica_s3) ? creep
   Exit: (12) stolen(stevey, galactica_s3) ? creep
   Exit: (11) unethical(stevey) ? creep
true.
```

Question 2:

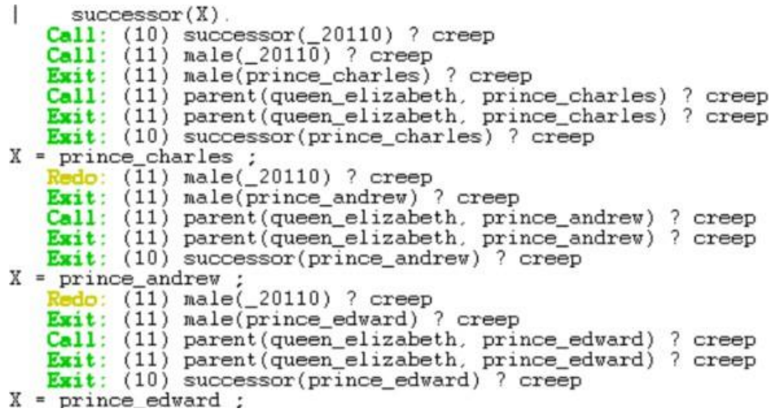
The old Royal succession rule states that the throne is passed down along the male line according to the order of birth before the consideration along the female line – similarly according to the order of birth. queen elizabeth, the monarch of the United Kingdom, has four offspring; namely:- prince charles,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

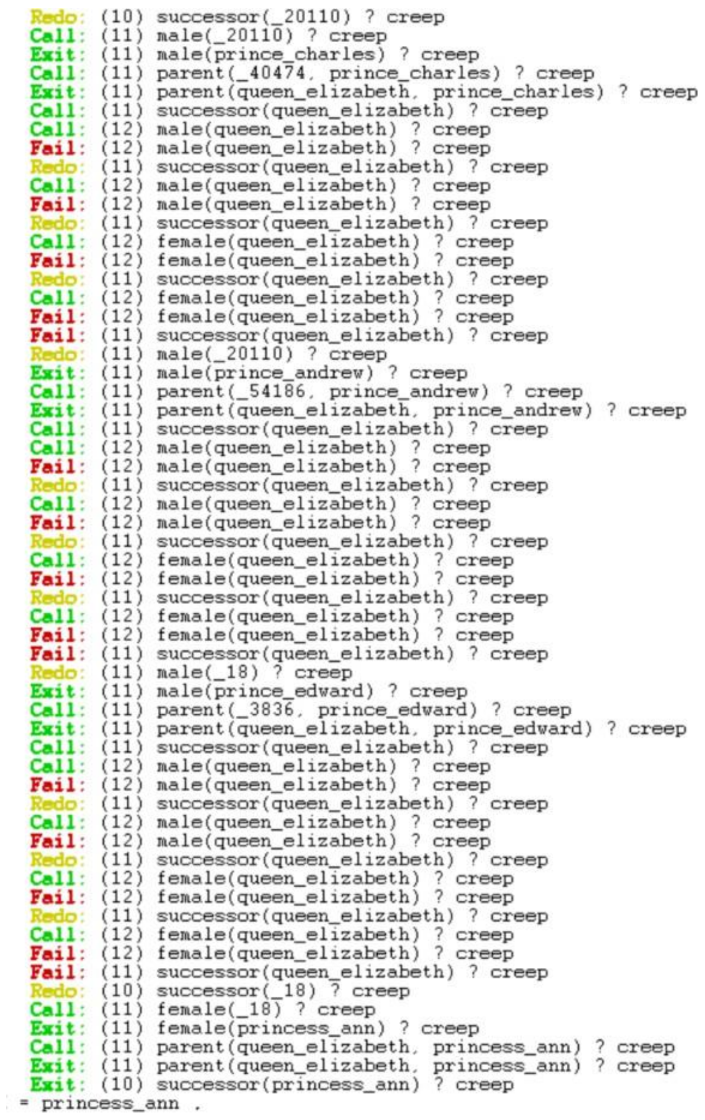
Prolog knowledge base:

```
parent(queen_elizabeth, prince_charles).
parent(queen_elizabeth, princess_ann).
parent(queen_elizabeth, prince_andrew).
parent(queen_elizabeth, prince_edward).

older(prince_charles, princess_ann).
older(princess_ann, prince_andrew).
older(prince_andrew, prince_edward).

male(prince_charles).
male(prince_andrew).
male(prince_edward).
female(princess_ann).

successor(X) :- male(X), parent(queen_elizabeth, X).
successor(X) :- male(X), parent(Y, X), successor(Y), older(Y, Z),
parent(Z, Y).
successor(X) :- female(X), parent(queen_elizabeth, X).
successor(X) :- female(X), parent(Y, X), successor(Y), older(Y, Z),
parent(Z, Y).
```

Trace for (1):



```
|     successor(X).
   Call: (10) successor(_20110) ? creep
   Call: (11) male(_20110) ? creep
   Exit: (11) male(prince_charles) ? creep
   Call: (11) parent(queen_elizabeth, prince_charles) ? creep
   Exit: (11) parent(queen_elizabeth, prince_charles) ? creep
   Exit: (10) successor(prince_charles) ? creep
X = prince_charles ;
   Redo: (11) male(_20110) ? creep
   Exit: (11) male(prince_andrew) ? creep
   Call: (11) parent(queen_elizabeth, prince_andrew) ? creep
   Exit: (11) parent(queen_elizabeth, prince_andrew) ? creep
   Exit: (10) successor(prince_andrew) ? creep
X = prince_andrew ;
   Redo: (11) male(_20110) ? creep
   Exit: (11) male(prince_edward) ? creep
   Call: (11) parent(queen_elizabeth, prince_edward) ? creep
   Exit: (11) parent(queen_elizabeth, prince_edward) ? creep
   Exit: (10) successor(prince_edward) ? creep
X = prince_edward ;
```



```
  Redo: (10) successor(_20110) ? creep
  Call: (11) male(_20110) ? creep
  Exit: (11) male(prince_charles) ? creep
  Call: (11) parent(_40474, prince_charles) ? creep
  Exit: (11) parent(queen_elizabeth, prince_charles) ? creep
  Call: (11) successor(queen_elizabeth) ? creep
  Call: (12) male(queen_elizabeth) ? creep
  Fail: (12) male(queen_elizabeth) ? creep
  Redo: (11) successor(queen_elizabeth) ? creep
  Call: (12) male(queen_elizabeth) ? creep
  Fail: (12) male(queen_elizabeth) ? creep
  Redo: (11) successor(queen_elizabeth) ? creep
  Call: (12) female(queen_elizabeth) ? creep
  Fail: (12) female(queen_elizabeth) ? creep
  Redo: (11) successor(queen_elizabeth) ? creep
  Call: (12) female(queen_elizabeth) ? creep
  Fail: (12) female(queen_elizabeth) ? creep
  Fail: (11) successor(queen_elizabeth) ? creep
  Redo: (11) male(_20110) ? creep
  Exit: (11) male(prince_andrew) ? creep
  Call: (11) parent(_54186, prince_andrew) ? creep
  Exit: (11) parent(queen_elizabeth, prince_andrew) ? creep
  Call: (11) successor(queen_elizabeth) ? creep
  Call: (12) male(queen_elizabeth) ? creep
  Fail: (12) male(queen_elizabeth) ? creep
  Redo: (11) successor(queen_elizabeth) ? creep
  Call: (12) male(queen_elizabeth) ? creep
  Fail: (12) male(queen_elizabeth) ? creep
  Redo: (11) successor(queen_elizabeth) ? creep
  Call: (12) female(queen_elizabeth) ? creep
  Fail: (12) female(queen_elizabeth) ? creep
  Redo: (11) successor(queen_elizabeth) ? creep
  Call: (12) female(queen_elizabeth) ? creep
  Fail: (12) female(queen_elizabeth) ? creep
  Fail: (11) successor(queen_elizabeth) ? creep
  Redo: (11) male(_18) ? creep
  Exit: (11) male(prince_edward) ? creep
  Call: (11) parent(_3836, prince_edward) ? creep
  Exit: (11) parent(queen_elizabeth, prince_edward) ? creep
  Call: (11) successor(queen_elizabeth) ? creep
  Call: (12) male(queen_elizabeth) ? creep
  Fail: (12) male(queen_elizabeth) ? creep
  Redo: (11) successor(queen_elizabeth) ? creep
  Call: (12) male(queen_elizabeth) ? creep
  Fail: (12) male(queen_elizabeth) ? creep
  Redo: (11) successor(queen_elizabeth) ? creep
  Call: (12) female(queen_elizabeth) ? creep
  Fail: (12) female(queen_elizabeth) ? creep
  Redo: (11) successor(queen_elizabeth) ? creep
  Call: (12) female(queen_elizabeth) ? creep
  Fail: (12) female(queen_elizabeth) ? creep
  Fail: (11) successor(queen_elizabeth) ? creep
  Redo: (10) successor(_18) ? creep
  Call: (11) female(_18) ? creep
  Exit: (11) female(princess_ann) ? creep
  Call: (11) parent(queen_elizabeth, princess_ann) ? creep
  Exit: (11) parent(queen_elizabeth, princess_ann) ? creep
  Exit: (10) successor(princess_ann) ? creep
 = 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.

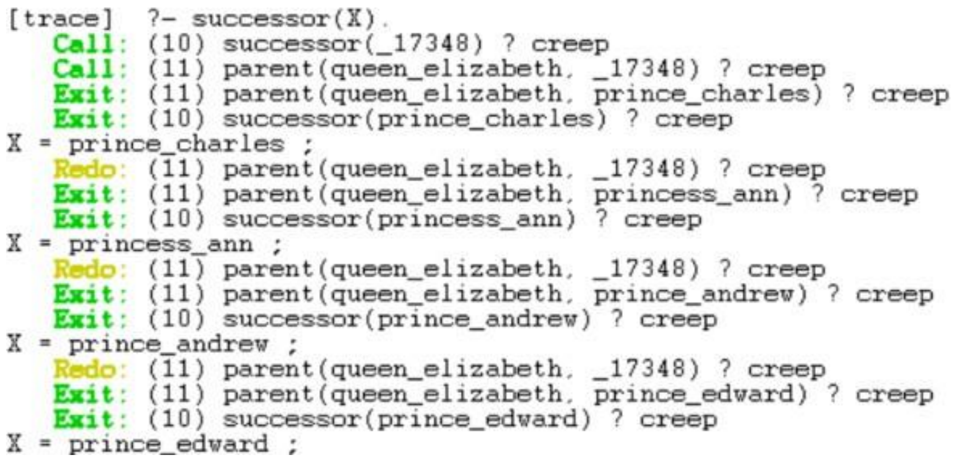
Modified Prolog base knowledge:

```
parent(queen_elizabeth, prince_charles).
parent(queen_elizabeth, princess_ann).
parent(queen_elizabeth, prince_andrew).
parent(queen_elizabeth, prince_edward).

older(prince_charles, princess_ann).
older(princess_ann, prince_andrew).
older(prince_andrew, prince_edward).

successor(X) :- parent(queen_elizabeth, X).
successor(X) :- parent(Y, X), successor(Y), older(Y, Z), parent(Z, Y).
```

Explanation:

We have removed the ‘male’ and ‘female’ predicates from the original code and modified the successor rule to only consider the order of birth, the first rule now says that a person is a successor if their parent is a successor, and they are older than their siblings. Also, the second rule states that Queen Elizabeth’s children are successors.

Trace for (2):



```
[trace]  ?- successor(X).
   Call: (10) successor(_17348) ? creep
   Call: (11) parent(queen_elizabeth, _17348) ? creep
   Exit: (11) parent(queen_elizabeth, prince_charles) ? creep
   Exit: (10) successor(prince_charles) ? creep
X = prince_charles ;
   Redo: (11) parent(queen_elizabeth, _17348) ? creep
   Exit: (11) parent(queen_elizabeth, princess_ann) ? creep
   Exit: (10) successor(princess_ann) ? creep
X = princess_ann ;
   Redo: (11) parent(queen_elizabeth, _17348) ? creep
   Exit: (11) parent(queen_elizabeth, prince_andrew) ? creep
   Exit: (10) successor(prince_andrew) ? creep
X = prince_andrew ;
   Redo: (11) parent(queen_elizabeth, _17348) ? creep
   Exit: (11) parent(queen_elizabeth, prince_edward) ? creep
   Exit: (10) successor(prince_edward) ? creep
X = prince_edward ;
```