Name: Ezra Koh See Hwa

Lab Group: SCSD

Matriculation Card Number: U2222028B

Exercise 1: The Smart Phone Rivalry

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. Smart phone technology is business.

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

    - Technology(Galactica-s3).
    - Competitor(Sumsum, Appy).
    - Developed(Sumsum, Galactica-s3).
    - Stole(Stevey, Galactica-s3).
    - Boss(Stevey, Appy).
    - Technology(w) ⇒ Business(w).
    - Competitor(x, y) ∨ Competitor(y,x) ⇒ Rival(x,y).
    - Boss(z, x) ∧ Stole(z, w) ∧ Business(w) ∧ Developed(y, w) ∧ Rival(x, y) ⇒ Unethical(z).

2. Write these FOL statements as Prolog clauses.

    Knowledge base attached as .pl file.

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

```
[trace] [5]  ?- unethical(stevey).
   Call: (422,049,393) unethical(stevey) ? creep
   Call: (422,049,394) boss(stevey, _39688) ? creep
   Exit: (422,049,394) boss(stevey, appy) ? creep
   Call: (422,049,394) stole(stevey, _41310) ? creep
   Exit: (422,049,394) stole(stevey, galactica-s3) ? creep
   Call: (422,049,394) business(galactica-s3) ? creep
   Call: (422,049,395) tech(galactica-s3) ? creep
   Exit: (422,049,395) tech(galactica-s3) ? creep
   Exit: (422,049,394) business(galactica-s3) ? creep
   Call: (422,049,394) develop(_46162, galactica-s3) ? creep
   Exit: (422,049,394) develop(sumsum, galactica-s3) ? creep
   Call: (422,049,394) rival(appy, sumsum) ? creep
   Call: (422,049,395) competitor(appy, sumsum) ? creep
   Fail: (422,049,395) competitor(appy, sumsum) ? creep
   Redo: (422,049,394) rival(appy, sumsum) ? creep
   Call: (422,049,395) competitor(sumsum, appy) ? creep
   Exit: (422,049,395) competitor(sumsum, appy) ? creep
   Exit: (422,049,394) rival(appy, sumsum) ? creep
   Exit: (422,049,393) 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.

   Knowledge base attached as .pl file.

   Successor(x,y) is interpreted as x is the successor of y.

```
[14]  ?- trace, successor(X,Y).
   Call: (221) successor(_30560, _30562) ? creep
   Call: (222) male_line(_30560, _30562) ? creep
   Call: (223) male(_30560) ? creep
   Exit: (223) male(prince_charles) ? creep
   Call: (223) male(_30562) ? creep
   Exit: (223) male(prince_charles) ? creep
   Call: (223) older(prince_charles, prince_charles) ? creep
   Fail: (223) older(prince_charles, prince_charles) ? creep
   Redo: (223) male(_30562) ? creep
   Exit: (223) male(prince_andrew) ? creep
   Call: (223) older(prince_andrew, prince_charles) ? creep
   Fail: (223) older(prince_andrew, prince_charles) ? creep
   Redo: (223) male(_30562) ? creep
   Exit: (223) male(prince_edward) ? creep
   Call: (223) older(prince_edward, prince_charles) ? creep
   Fail: (223) older(prince_edward, prince_charles) ? creep
   Redo: (222) male_line(prince_charles, _30562) ? creep
   Call: (223) reigning(_30562) ? creep
   Exit: (223) reigning(queen_elizabeth) ? creep
   Call: (223) older(queen_elizabeth, prince_charles) ? creep
   Exit: (223) older(queen_elizabeth, prince_charles) ? creep
   Exit: (222) male_line(prince_charles, queen_elizabeth) ? creep
   Exit: (221) successor(prince_charles, queen_elizabeth) ? creep
X = prince_charles,
Y = queen_elizabeth ;
```

```
   Redo: (222) male_line(prince_charles, _30562) ? creep
   Call: (223) skip_girl(prince_charles, _30562) ? creep
   Call: (224) older(_53316, prince_charles) ? creep
   Exit: (224) older(queen_elizabeth, prince_charles) ? creep
   Call: (224) female(queen_elizabeth) ? creep
   Fail: (224) female(queen_elizabeth) ? creep
   Redo: (224) older(_53316, prince_charles) ? creep
   Fail: (224) older(_53316, prince_charles) ? creep
   Fail: (223) skip_girl(prince_charles, _30562) ? creep
   Redo: (223) male(_30560) ? creep
   Exit: (223) male(prince_andrew) ? creep
   Call: (223) male(_30562) ? creep
   Exit: (223) male(prince_charles) ? creep
   Call: (223) older(prince_charles, prince_andrew) ? creep
   Fail: (223) older(prince_charles, prince_andrew) ? creep
   Redo: (223) male(_30562) ? creep
   Exit: (223) male(prince_andrew) ? creep
   Call: (223) older(prince_andrew, prince_andrew) ? creep
   Fail: (223) older(prince_andrew, prince_andrew) ? creep
   Redo: (223) male(_30562) ? creep
   Exit: (223) male(prince_edward) ? creep
   Call: (223) older(prince_edward, prince_andrew) ? creep
   Fail: (223) older(prince_edward, prince_andrew) ? creep
   Redo: (222) male_line(prince_andrew, _30562) ? creep
   Call: (223) reigning(_30562) ? creep
   Exit: (223) reigning(queen_elizabeth) ? creep
   Call: (223) older(queen_elizabeth, prince_andrew) ? creep
   Fail: (223) older(queen_elizabeth, prince_andrew) ? creep
   Redo: (222) male_line(prince_andrew, _30562) ? creep
   Call: (223) skip_girl(prince_andrew, _30562) ? creep
   Call: (224) older(_75950, prince_andrew) ? creep
   Exit: (224) older(princess_ann, prince_andrew) ? creep
   Call: (224) female(princess_ann) ? creep
   Exit: (224) female(princess_ann) ? creep
   Call: (224) older(_30562, princess_ann) ? creep
   Exit: (224) older(prince_charles, princess_ann) ? creep
   Call: (224) male(prince_charles) ? creep
   Exit: (224) male(prince_charles) ? creep
   Exit: (223) skip_girl(prince_andrew, prince_charles) ? creep
   Exit: (222) male_line(prince_andrew, prince_charles) ? creep
   Exit: (221) successor(prince_andrew, prince_charles) ? creep
X = prince_andrew,
Y = prince_charles ;

   Redo: (224) older(_30562, princess_ann) ? creep
   Fail: (224) older(_30562, princess_ann) ? creep
   Redo: (224) older(_75950, prince_andrew) ? creep
   Fail: (224) older(_75950, prince_andrew) ? creep
   Fail: (223) skip_girl(prince_andrew, _30562) ? creep
   Redo: (223) male(_30560) ? creep
   Exit: (223) male(prince_edward) ? creep
   Call: (223) male(_30562) ? creep
   Exit: (223) male(prince_charles) ? creep
   Call: (223) older(prince_charles, prince_edward) ? creep
   Fail: (223) older(prince_charles, prince_edward) ? creep
   Redo: (223) male(_30562) ? creep
   Exit: (223) male(prince_andrew) ? creep
   Call: (223) older(prince_andrew, prince_edward) ? creep
   Exit: (223) older(prince_andrew, prince_edward) ? creep
   Exit: (222) male_line(prince_edward, prince_andrew) ? creep
   Exit: (221) successor(prince_edward, prince_andrew) ? creep
X = prince_edward,
Y = prince_andrew ;
```

```
   Redo: (223) male(_30562) ? creep
   Exit: (223) male(prince_edward) ? creep
   Call: (223) older(prince_edward, prince_edward) ? creep
   Fail: (223) older(prince_edward, prince_edward) ? creep
   Redo: (222) male_line(prince_edward, _30562) ? creep
   Call: (223) reigning(_30562) ? creep
   Exit: (223) reigning(queen_elizabeth) ? creep
   Call: (223) older(queen_elizabeth, prince_edward) ? creep
   Fail: (223) older(queen_elizabeth, prince_edward) ? creep
   Redo: (222) male_line(prince_edward, _30562) ? creep
   Call: (223) skip_girl(prince_edward, _30562) ? creep
   Call: (224) older(_111302, prince_edward) ? creep
   Exit: (224) older(prince_andrew, prince_edward) ? creep
   Call: (224) female(prince_andrew) ? creep
   Fail: (224) female(prince_andrew) ? creep
   Fail: (223) skip_girl(prince_edward, _30562) ? creep
   Fail: (222) male_line(_30560, _30562) ? creep
   Redo: (221) successor(_30560, _30562) ? creep
   Call: (222) transition(_30560, _30562) ? creep
   Call: (223) female(_30560) ? creep
   Exit: (223) female(princess_ann) ? creep
   Call: (223) youngest_male(_30562) ? creep
   Exit: (223) youngest_male(prince_edward) ? creep
   Exit: (222) transition(princess_ann, prince_edward) ? creep
   Exit: (221) successor(princess_ann, prince_edward) ? creep
X = princess_ann,
Y = prince_edward ;
   Redo: (221) successor(_30560, _30562) ? creep
   Call: (222) female_line(_30560, _30562) ? creep
   Call: (223) female(_30560) ? creep
   Exit: (223) female(princess_ann) ? creep
   Call: (223) female(_30562) ? creep
   Exit: (223) female(princess_ann) ? creep
   Call: (223) older(princess_ann, princess_ann) ? creep
   Fail: (223) older(princess_ann, princess_ann) ? creep
   Fail: (222) female_line(_30560, _30562) ? creep
   Fail: (221) successor(_30560, _30562) ? creep
false.
```

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.

   Knowledge base attached as .pl file.

   Successor(x,y) is interpreted as x is the successor of y.

   Since the rule has been modified by the removal of a condition (gender), less knowledge is needed to represent the new information. A simple list of who is directly older than who is sufficient.

```
[trace] [7]  ?- successor(X,Y).
   Call: (149) successor(_2972, _2974) ? creep
   Call: (150) older(_2974, _2972) ? creep
   Exit: (150) older(queen_elizabeth, prince_charles) ? creep
   Exit: (149) successor(prince_charles, queen_elizabeth) ? creep
X = prince_charles,
Y = queen_elizabeth ;
   Redo: (150) older(_2974, _2972) ? creep
   Exit: (150) older(prince_charles, princess_ann) ? creep
   Exit: (149) successor(princess_ann, prince_charles) ? creep
X = princess_ann,
Y = prince_charles ;
   Redo: (150) older(_2974, _2972) ? creep
   Exit: (150) older(princess_ann, prince_andrew) ? creep
   Exit: (149) successor(prince_andrew, princess_ann) ? creep
X = prince_andrew,
Y = princess_ann ;
   Redo: (150) older(_2974, _2972) ? creep
   Exit: (150) older(prince_andrew, prince_edward) ? creep
   Exit: (149) successor(prince_edward, prince_andrew) ? creep
X = prince_edward,
Y = prince_andrew.
```