Assignment 2

Team Name: InSync

Tutorial Group: A37

Contributions:

1) Lim Boon Hian (U2120791F)

- Programmed Knowledge Base and FOL rules into Prolog for Q1 and Q2.
- Improved code and documented formatting.

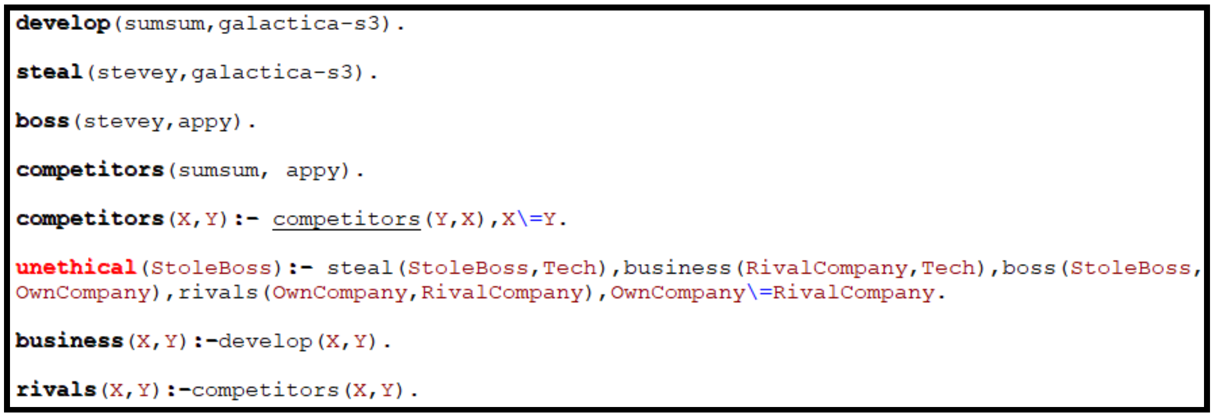
2) Tio Guo Yong (U2123181B)

- Defined FOL statements for Q1.
- Improved Prolog logic for Q2(a).

1(a)

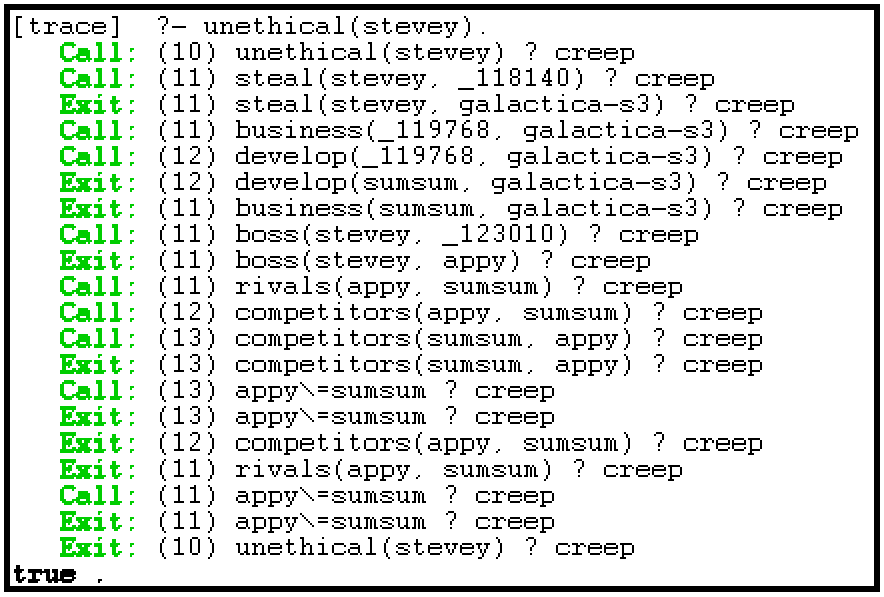
Statements	**FOL**
sumsum, a competitor of appy	Competitors(sumsum, appy)
sumsum developed some nice smartphone technology called galactica-s3	Develop(sumsum, galactica-s3)
smartphone technology, galactica-s3 stolen by stevey	Steal(stevey, galactica-s3)
stevey is a boss of appy	Boss(stevey, appy)
If company X is a competitor to company Y, then company Y is also a competitor to company X	$\forall x, \forall y, (x \neq y \land \text{Competitors}(x, y)) \Rightarrow \text{Competitors}(y, x)$
A competitor is a rival	$\forall x, \forall y, \text{Competitors}(x, y) \Rightarrow \text{Rivals}(x, y)$
Smartphone technology is a business: A technology, X developed by company Y is a business for Y	$\forall x, \forall y, \text{Develop}(x,y) \Rightarrow \text{Business}(x,y)$
It is unethical for a boss (A) of a company (X) to steal business (B) from rival companies (Y)	$\forall a, \forall b, \forall x, \forall y, (x \neq y) \land \text{Steal}(a, b) \land \text{Boss}(a, x) \land \text{Business}(y, b) \land \text{Rivals}(x, y) \Rightarrow \text{Unethical}(x)$

1(b)



```
develop(sumsum,galactica-s3).

steal(stevey,galactica-s3).

boss(stevey,appy).

competitors(sumsum, appy).

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

unethical(StoleBoss):- steal(StoleBoss,Tech),business(RivalCompany,Tech),boss(StoleBoss,
OwnCompany),rivals(OwnCompany,RivalCompany),OwnCompany\=RivalCompany.

business(X,Y):-develop(X,Y).

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

1(c)



```
[trace]  ?- unethical(stevey).
   Call: (10) unethical(stevey) ? creep
   Call: (11) steal(stevey, _118140) ? creep
   Exit: (11) steal(stevey, galactica-s3) ? creep
   Call: (11) business(_119768, galactica-s3) ? creep
   Call: (12) develop(_119768, galactica-s3) ? creep
   Exit: (12) develop(sumsum, galactica-s3) ? creep
   Exit: (11) business(sumsum, galactica-s3) ? creep
   Call: (11) boss(stevey, _123010) ? creep
   Exit: (11) boss(stevey, appy) ? creep
   Call: (11) rivals(appy, sumsum) ? creep
   Call: (12) competitors(appy, sumsum) ? creep
   Call: (13) competitors(sumsum, appy) ? creep
   Exit: (13) competitors(sumsum, appy) ? creep
   Call: (13) appy\=sumsum ? creep
   Exit: (13) appy\=sumsum ? creep
   Exit: (12) competitors(appy, sumsum) ? creep
   Exit: (11) rivals(appy, sumsum) ? creep
   Call: (11) appy\=sumsum ? creep
   Exit: (11) appy\=sumsum ? creep
   Exit: (10) unethical(stevey) ? creep
true .
```

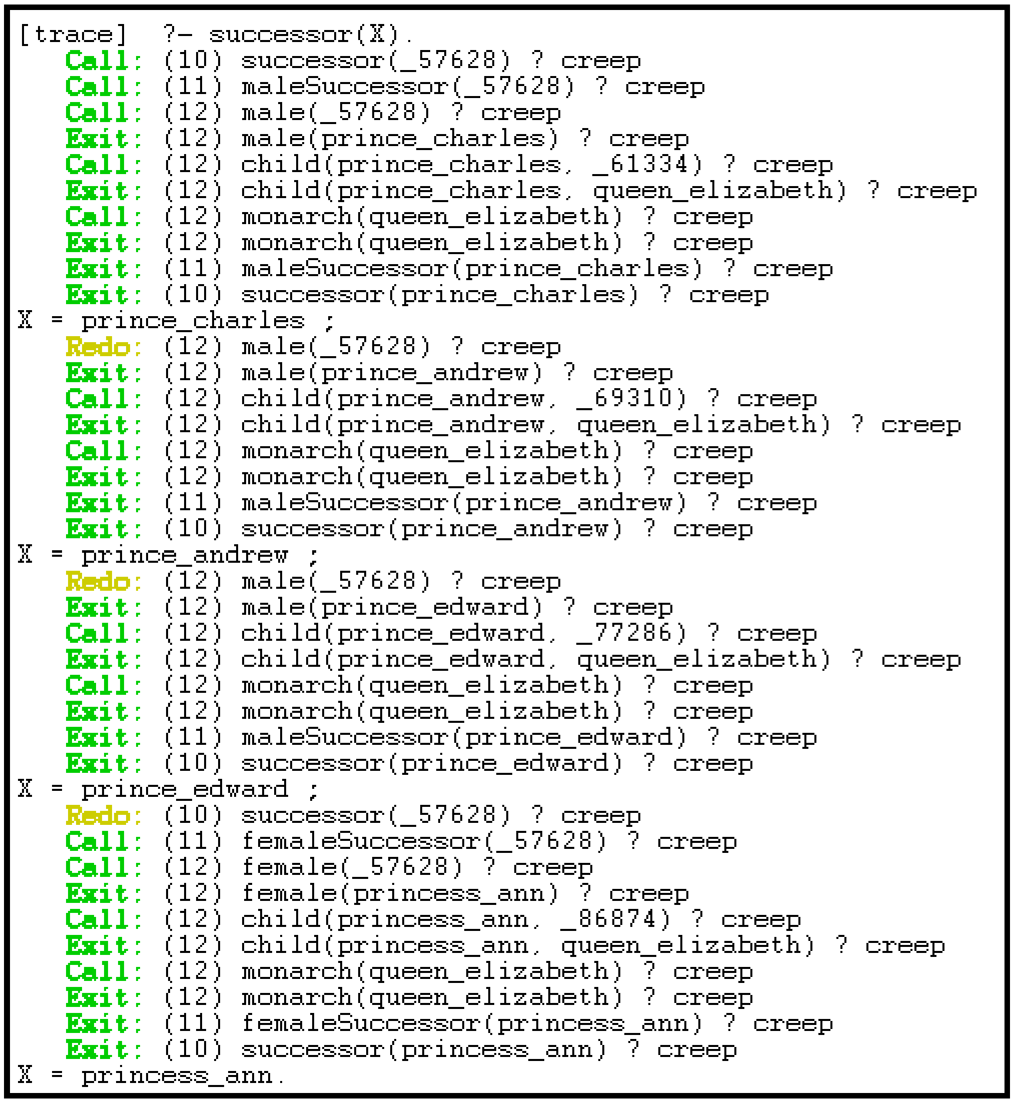
2

In the Prolog rule base, the order of birth is implied by the ordering of the Prolog code for the 'child' relationship due to Prolog's compiler. In our case, child(prince_charles, queen_elizabeth) in the first row [see Figure in (a)] means that Prince Charles is the eldest child of Queen Elizabeth. Likewise, child(prince_edward, queen_elizabeth) in the fourth row means that Prince Edward is the fourth child of Queen Elizabeth. Hence, we do not need to explicitly specify the order of birth in Prolog.

(a)

For this question, Prolog computes the line of male succession followed by the line of female succession according to the order of birth.

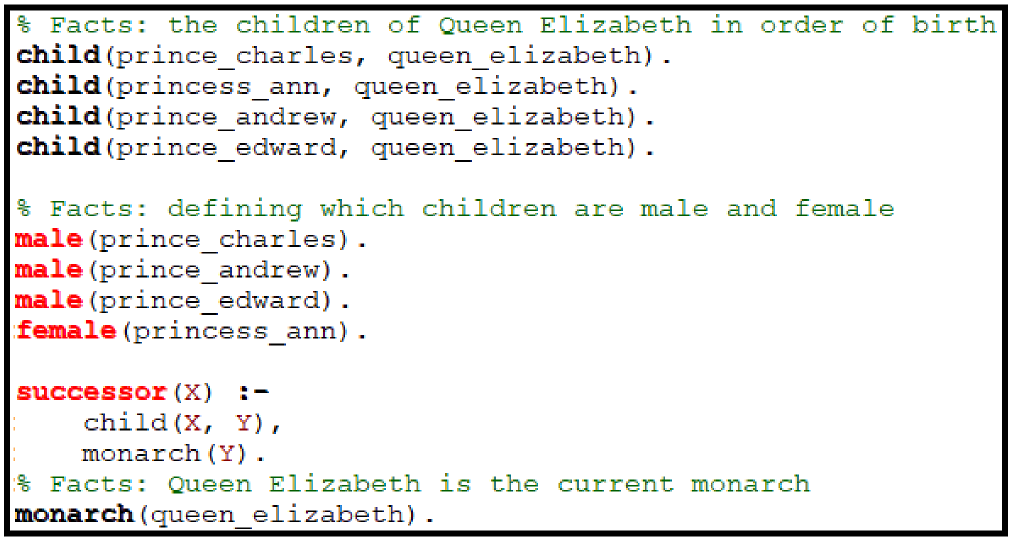
```
% Facts: the children of Queen Elizabeth in order of birth
child(prince_charles, queen_elizabeth).
child(princess_ann, queen_elizabeth).
child(prince_andrew, queen_elizabeth).
child(prince_edward, queen_elizabeth).

% Rules: the old Royal succession rule
successor(X) :-
    maleSuccessor(X);femaleSuccessor(X).
maleSuccessor(X) :-
    male(X),
    child(X, Y),
    monarch(Y).
femaleSuccessor(X):-
    female(X),
    child(X, Q),
    monarch(Q).

% Facts: Queen Elizabeth is the current monarch
monarch(queen_elizabeth).

% Facts: defining which children are male and female
male(prince_charles).
male(prince_andrew).
male(prince_edward).
female(princess_ann).
```



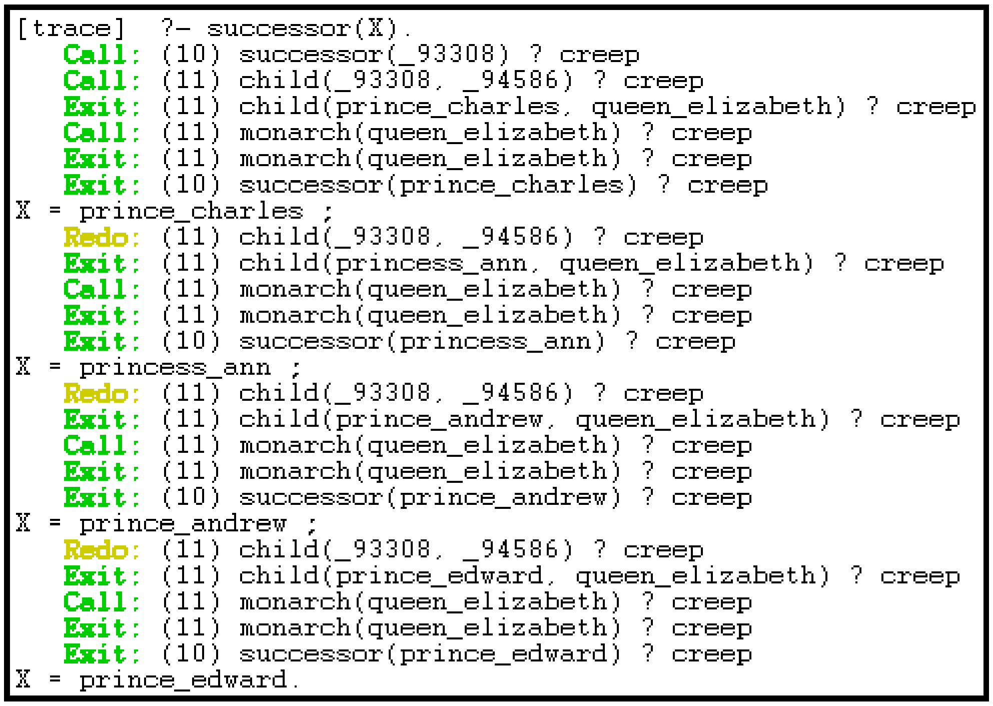
```
[trace]  ?- successor(X).
   Call: (10) successor(_57628) ? creep
   Call: (11) maleSuccessor(_57628) ? creep
   Call: (12) male(_57628) ? creep
   Exit: (12) male(prince_charles) ? creep
   Call: (12) child(prince_charles, _61334) ? creep
   Exit: (12) child(prince_charles, queen_elizabeth) ? creep
   Call: (12) monarch(queen_elizabeth) ? creep
   Exit: (12) monarch(queen_elizabeth) ? creep
   Exit: (11) maleSuccessor(prince_charles) ? creep
   Exit: (10) successor(prince_charles) ? creep
X = prince_charles ;
   Redo: (12) male(_57628) ? creep
   Exit: (12) male(prince_andrew) ? creep
   Call: (12) child(prince_andrew, _69310) ? creep
   Exit: (12) child(prince_andrew, queen_elizabeth) ? creep
   Call: (12) monarch(queen_elizabeth) ? creep
   Exit: (12) monarch(queen_elizabeth) ? creep
   Exit: (11) maleSuccessor(prince_andrew) ? creep
   Exit: (10) successor(prince_andrew) ? creep
X = prince_andrew ;
   Redo: (12) male(_57628) ? creep
   Exit: (12) male(prince_edward) ? creep
   Call: (12) child(prince_edward, _77286) ? creep
   Exit: (12) child(prince_edward, queen_elizabeth) ? creep
   Call: (12) monarch(queen_elizabeth) ? creep
   Exit: (12) monarch(queen_elizabeth) ? creep
   Exit: (11) maleSuccessor(prince_edward) ? creep
   Exit: (10) successor(prince_edward) ? creep
X = prince_edward ;
   Redo: (10) successor(_57628) ? creep
   Call: (11) femaleSuccessor(_57628) ? creep
   Call: (12) female(_57628) ? creep
   Exit: (12) female(princess_ann) ? creep
   Call: (12) child(princess_ann, _86874) ? creep
   Exit: (12) child(princess_ann, queen_elizabeth) ? creep
   Call: (12) monarch(queen_elizabeth) ? creep
   Exit: (12) monarch(queen_elizabeth) ? creep
   Exit: (11) femaleSuccessor(princess_ann) ? creep
   Exit: (10) successor(princess_ann) ? creep
X = princess_ann.
```

(b)

For this question, Prolog computes the line of succession according to the order of birth irrespective of the gender.



```
% Facts: the children of Queen Elizabeth in order of birth
child(prince_charles, queen_elizabeth).
child(princess_ann, queen_elizabeth).
child(prince_andrew, queen_elizabeth).
child(prince_edward, queen_elizabeth).

% Facts: defining which children are male and female
male(prince_charles).
male(prince_andrew).
male(prince_edward).
female(princess_ann).

successor(X) :-
    child(X, Y),
    monarch(Y).
% Facts: Queen Elizabeth is the current monarch
monarch(queen_elizabeth).
```



```
[trace]  ?- successor(X).
   Call: (10) successor(_93308) ? creep
   Call: (11) child(_93308, _94586) ? creep
   Exit: (11) child(prince_charles, queen_elizabeth) ? creep
   Call: (11) monarch(queen_elizabeth) ? creep
   Exit: (11) monarch(queen_elizabeth) ? creep
   Exit: (10) successor(prince_charles) ? creep
X = prince_charles ;
   Redo: (11) child(_93308, _94586) ? creep
   Exit: (11) child(princess_ann, queen_elizabeth) ? creep
   Call: (11) monarch(queen_elizabeth) ? creep
   Exit: (11) monarch(queen_elizabeth) ? creep
   Exit: (10) successor(princess_ann) ? creep
X = princess_ann ;
   Redo: (11) child(_93308, _94586) ? creep
   Exit: (11) child(prince_andrew, queen_elizabeth) ? creep
   Call: (11) monarch(queen_elizabeth) ? creep
   Exit: (11) monarch(queen_elizabeth) ? creep
   Exit: (10) successor(prince_andrew) ? creep
X = prince_andrew ;
   Redo: (11) child(_93308, _94586) ? creep
   Exit: (11) child(prince_edward, queen_elizabeth) ? creep
   Call: (11) monarch(queen_elizabeth) ? creep
   Exit: (11) monarch(queen_elizabeth) ? creep
   Exit: (10) successor(prince_edward) ? creep
X = prince_edward.
```