NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

SC3000: ARTIFICIAL INTELLIGENCE

LAB ASSIGNMENT 2

Lab Group: SDAB

Group Members:

Name	Matric No.	Contributions
Charmain Ang Wan Theng	U2223590D	Exercise 1
Darrus Mok Khai Kiat	U2223371E	Exercise 2

Exercise 1: The Smart Phone Rivalry (Charmain)

sumsum, a competitor of appy, developed some nice smart phone 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. 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).**

"sumsum, a competitor of appy"

Company(sumsum)
Company(appy)

Competitor(sumsum,appy)
Competitor(appy, sumsum)

"developed some nice smart phone technology called galactica- s3"

Technology(galactica- s3)
Developed(sumsum, galactica- s3)

"all of which was stolen by stevey, who is a boss of appy."

Stolen(stevey, galactica- s3)
Boss(stevey, appy)

"A competitor is a rival. Smart phone technology is business."

Competitor(x,y) ⇒ Rival(x,y)
Technology(x) ⇒ Business(x)

"It is unethical for a boss to steal business from rival companies."
X is a boss of a company Y : Company(y) ∧ Boss(x,y)
X stole business Z: Business(z) ∧ Stolen(x,z)
Business Z is developed by another company a: Company(a) ∧ Developed(a,z) ∧ Rival(y,a)

Unethical(x) ⇔ Company(y) ∧ Boss(x,y) ∧ Business(z) ∧ Stolen(x,z) ∧ Company(a) ∧ Developed(a,z) ∧ Rival(y,a)

2. **Write these FOL statements as Prolog clauses.**

```
company(sumsum).
company(appy).

competitor(sumsum, appy).
competitor(appy, sumsum).

technology(galactica-s3).
developed(sumsum, galactica-s3).

stolen(stevey, galactica-s3).
boss(stevey, appy).

rival(X, Y) :-
   competitor(X, Y);
   competitor(Y, X).

business(X) :-
   technology(X).

unethical(X) :- company(Y), boss(X, Y), business(Z), stolen(X, Z),
company(A),developed(A, Z), rival(Y, A).
```

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

```
[trace]  ?- trace, unethical(stevey).
   Call: (13) unethical(stevey) ? creep
   Call: (14) boss(stevey, _78762) ? creep
   Exit: (14) boss(stevey, appy) ? creep
   Call: (14) stolen(stevey, _80384) ? creep
   Exit: (14) stolen(stevey, galactica_s3) ? creep
   Call: (14) company(appy) ? creep
   Exit: (14) company(appy) ? creep
   Call: (14) business(galactica_s3) ? creep
   Call: (15) technology(galactica_s3) ? creep
   Exit: (15) technology(galactica_s3) ? creep
   Exit: (14) business(galactica_s3) ? creep
   Call: (14) company(_86842) ? creep
   Exit: (14) company(sumsum) ? creep
   Call: (14) developed(sumsum, galactica_s3) ? creep
   Exit: (14) developed(sumsum, galactica_s3) ? creep
   Call: (14) rival(appy, sumsum) ? creep
   Call: (15) competitor(appy, sumsum) ? creep
   Exit: (15) competitor(appy, sumsum) ? creep
   Exit: (14) rival(appy, sumsum) ? creep
   Exit: (13) unethical(stevey) ? creep
true .
```

Exercise 2: The Royal Family (Darrus)

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.**

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


offspring(prince_charles,queen_elizabeth).
offspring(prince_andrew,queen_elizabeth).
offspring(prince_edward,queen_elizabeth).
offspring(princess_ann,queen_elizabeth).

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


successor(X) :- male(X), offspring(X,elizabeth).
successor(X) :- male(X), offspring(X,Y),successor(Y),older_than(Y,Z),offspring(Y,Z).
successor(X) :- female(X),parent(elizabeth,X).
successor(X) :- female(X),offspring(X,Y),successor(Y),older_than(Y,Z),offspring(Y,Z)
```

```
?- trace,successor(X).
   Call: (13) successor(_30802) ? creep
   Call: (14) male(_30802) ? creep
   Exit: (14) male(prince_charles) ? creep
   Call: (14) offspring(prince_charles, queen_elizabeth) ? creep
   Exit: (14) offspring(prince_charles, queen_elizabeth) ? creep
   Exit: (13) successor(prince_charles) ? creep
X = prince_charles ;
   Redo: (14) male(_30802) ? creep
   Exit: (14) male(prince_andrew) ? creep
   Call: (14) offspring(prince_andrew, queen_elizabeth) ? creep
   Exit: (14) offspring(prince_andrew, queen_elizabeth) ? creep
   Exit: (13) successor(prince_andrew) ? creep
X = prince_andrew ;
   Redo: (14) male(_30802) ? creep
   Exit: (14) male(prince_edward) ? creep
   Call: (14) offspring(prince_edward, queen_elizabeth) ? creep
   Exit: (14) offspring(prince_edward, queen_elizabeth) ? creep
   Exit: (13) successor(prince_edward) ? creep
X = prince_edward ;
```

```
    Redo: (13) successor(_30802) ? creep
    Call: (14) male(_30802) ? creep
    Exit: (14) male(prince_charles) ? creep
    Call: (14) offspring(prince_charles, _51282) ? creep
    Exit: (14) offspring(prince_charles, queen_elizabeth) ? creep
    Call: (14) successor(queen_elizabeth) ? creep
    Call: (15) male(queen_elizabeth) ? creep
    Fail: (15) male(queen_elizabeth) ? creep
    Redo: (14) successor(queen_elizabeth) ? creep
    Call: (15) male(queen_elizabeth) ? creep
    Fail: (15) male(queen_elizabeth) ? creep
    Redo: (14) successor(queen_elizabeth) ? creep
    Call: (15) female(queen_elizabeth) ? creep
    Fail: (15) female(queen_elizabeth) ? creep
    Redo: (14) successor(queen_elizabeth) ? creep
    Call: (15) female(queen_elizabeth) ? creep
    Fail: (15) female(queen_elizabeth) ? creep
    Fail: (14) successor(queen_elizabeth) ? creep
    Redo: (14) male(_30802) ? creep
    Exit: (14) male(prince_andrew) ? creep
    Call: (14) offspring(prince_andrew, _64994) ? creep
    Exit: (14) offspring(prince_andrew, queen_elizabeth) ? creep
    Call: (14) successor(queen_elizabeth) ? creep
    Call: (15) male(queen_elizabeth) ? creep
    Fail: (15) male(queen_elizabeth) ? creep
    Redo: (14) successor(queen_elizabeth) ? creep
    Call: (15) male(queen_elizabeth) ? creep
    Fail: (15) male(queen_elizabeth) ? creep
    Redo: (14) successor(queen_elizabeth) ? creep
    Call: (15) female(queen_elizabeth) ? creep
    Fail: (15) female(queen_elizabeth) ? creep
    Redo: (14) successor(queen_elizabeth) ? creep
    Call: (15) female(queen_elizabeth) ? creep
    Fail: (15) female(queen_elizabeth) ? creep
    Fail: (14) successor(queen_elizabeth) ? creep
    Redo: (14) male(_30802) ? creep
    Exit: (14) male(prince_edward) ? creep
    Call: (14) offspring(prince_edward, _78706) ? creep
    Exit: (14) offspring(prince_edward, queen_elizabeth) ? creep
    Call: (14) successor(queen_elizabeth) ? creep
    Call: (15) male(queen_elizabeth) ? creep
    Fail: (15) male(queen_elizabeth) ? creep
    Redo: (14) successor(queen_elizabeth) ? creep
    Call: (15) male(queen_elizabeth) ? creep
    Fail: (15) male(queen_elizabeth) ? creep
    Redo: (14) successor(queen_elizabeth) ? creep
    Call: (15) female(queen_elizabeth) ? creep
    Fail: (15) female(queen_elizabeth) ? creep
    Redo: (14) successor(queen_elizabeth) ? creep
    Call: (15) female(queen_elizabeth) ? creep
    Fail: (15) female(queen_elizabeth) ? creep
    Fail: (14) successor(queen_elizabeth) ? creep
    Redo: (13) successor(_30802) ? creep
    Call: (14) female(_30802) ? creep
    Exit: (14) female(princess_ann) ? creep
    Call: (14) offspring(princess_ann, queen_elizabeth) ? creep
    Exit: (14) offspring(princess_ann, queen_elizabeth) ? creep
    Exit: (13) successor(princess_ann) ? creep
X = 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.**

   In question 2, gender is not considered in choosing the successor, so we have removed the 'male' and 'female' predicates from the code and retain the order of birth, the first rule now states that Queen Elizabeth's offsprings are successors and says that a person is a successor if their parent is a successor, and they are older than their siblings.

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


offspring(prince_charles,queen_elizabeth).
offspring(prince_andrew,queen_elizabeth).
offspring(prince_edward,queen_elizabeth).
offspring(princess_ann,queen_elizabeth).

older_than(prince_charles,princess_ann).
older_than(prince_ann,prince_andrew).
older_than(prince_andrew,prince_edward).


successor(X) :- offspring(X,queen_elizabeth).
successor(X) :- offspring(X,Y),successor(Y),older_than(Y,Z),offspring(Y,Z).
```

```
?- trace,successor(X).
   Call: (13) successor(_11052) ? creep
   Call: (14) offspring(_11052, queen_elizabeth) ? creep
   Exit: (14) offspring(prince_charles, queen_elizabeth) ? creep
   Exit: (13) successor(prince_charles) ? creep
X = prince_charles ;
   Redo: (14) offspring(_11052, queen_elizabeth) ? creep
   Exit: (14) offspring(prince_andrew, queen_elizabeth) ? creep
   Exit: (13) successor(prince_andrew) ? creep
X = prince_andrew ;
   Redo: (14) offspring(_11052, queen_elizabeth) ? creep
   Exit: (14) offspring(prince_edward, queen_elizabeth) ? creep
   Exit: (13) successor(prince_edward) ? creep
X = prince_edward ;
   Redo: (14) offspring(_11052, queen_elizabeth) ? creep
   Exit: (14) offspring(princess_ann, queen_elizabeth) ? creep
   Exit: (13) successor(princess_ann) ? creep
X = princess_ann .
```