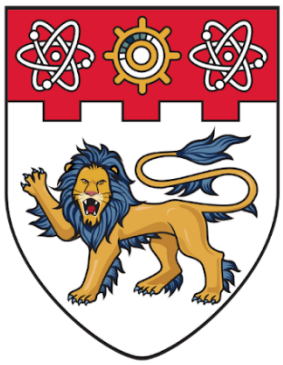

NANYANG TECHNOLOGICAL UNIVERSITY
SINGAPORE

School of Computer Science & Engineering

CZ3005 / SC3000 - Artificial Intelligence

Group SCSD

Name	Matric No	Contribution
Chee Wen Zhan	U2122475L	Question 1
Agarwal Anusha	U2023105H	Question 2

Table of Contents

Exercise 1 - The Smart Phone Rivalry **3**

a. Translation of given natural language statements into First Order Logic (FOL): 3

b. FOL statements to Prolog clauses (Code): 4

c. Simple Output 5

d. Trace showing Stevey is unethical: 5

Exercise 2 - The Royal Family: **6**

a. Line of Succession as per original rule - 6

A. Rules in Knowledge Base - 6

1. Translation of given natural language statements into First Order Logic (FOL) Predicates: 6

2. Rule Predicates 7

3. Functions 8

4. FOL Statements to Prolog clauses (Code) 9

5. Simple Output 10

6. Trace 11

b. Line of Succession as per new rule - 23

A. Rules in Knowledge Base - 23

Change From Old succession rule - 23

1. Translation of given natural language statements into First Order Logic (FOL) Predicates: 23

2. Rule Predicates 24

3. Functions 24

4. FOL Statements to Prolog clauses (Code) 26

5. Simple Output 27

6. Trace 27

Exercise 1 - The Smart Phone Rivalry

Predicates -

1. competitor(X, Y): X is a competitor of Y
2. develop(X, T): X company developed T technology
3. is_smart_phone_technology(T): T is a smartphone technology
4. steal(X, T): X has stolen T technology
5. boss(X, Y): X is the boss of company Y
6. business(X) :- smartphoneTechnology(X).
7. rival(X, Y) :- competitor(X, Y), X!=Y.
8. unethical(X) :- X is unethical

a. Translation of **given natural language** statements into **First Order Logic (FOL):**

S. No.	Natural Language	First Order Logic
1.	Sumsum is a competitor of Appy.	`Competitor(sumsum, appy)`
2.	Galacticas3 is a smartphone technology developed by Sumsum.	`Develop(sumsum, galactica-s3)` `Is_Smart_Phone_Technology(galactica-s3)`
3.	Stevey stole Galacticas3.	`Steal(stevey, galactica-s3)`
4.	Stevey is the boss of Appy.	`Boss(stevey, appy)`
5.	It is unethical for a boss to steal business from rival companies.	`∀p, ∀x, ∀y, ∀t, Boss(p, x) ^ Steal(p, t) ^ Is_Business(t) ^ Rival(x, y) ^ Develop(y, t) => Unethical(p)`
6.	A competitor is a rival.	`∀x, ∀y, Competitor(x, y) ^ x != y => Rival(x, y)`
7.	Smartphone technology is a business.	`∀t, Is_Smart_Phone_Technology(t) => Is_Business(t)`

b. **FOL statements to Prolog clauses (Code):**

Please refer to the attached pl file for the full code.

```
/* ------------ Facts and Definitions -----------
    company(x) -> x is a company
    competitor(x, y) -> x and y are competitors
    develop(x, y) -> x develops Product y
    steal(x, y) -> x steals Product y
    boss(x, y) -> x is the boss of y

*/

company(sumsum).
company(appy).
competitor(sumsum, appy).
develop(sumsum, galactica-s3).
smart_phone_tech(galactica-S3).
steal(stevey, galactica-s3).
boss(stevey, appy).
```

```
/* ------------ Rules and Definitions -----------*/
/* Defined by logic */
competitor(X, Y) :- company(X), company(Y), X \= Y.
boss(X, Y) :- company(Y).
develop(X, Y) :- company(X), business(Y).
steal(X, Y) :- business(Y).

/* Defined by question */
rival(X, Y) :- competitor(X, Y), X \= Y.
business(Product) :- smart_phone_tech(Product).
unethical(Person) :- boss(Person, X), steal(Person, Product),
business(Product), rival(X, Y), develop(Y, Product).
```

c. **Simple Output**

```
[debug] 4 ?- unethical(stevey).
true .
```

d. **Trace showing Stevey is unethical:**

```
   Call: (12) unethical(stevey) ? creep
   Call: (13) boss(stevey, _9840) ? creep
   Exit: (13) boss(stevey, appy) ? creep
   Call: (13) steal(stevey, _11462) ? creep
   Exit: (13) steal(stevey, galactica-s3) ? creep
   Call: (13) business(galactica-s3) ? creep
   Call: (14) smart_phone_tech(galactica-s3) ? creep
   Exit: (14) smart_phone_tech(galactica-s3) ? creep
   Exit: (13) business(galactica-s3) ? creep
   Call: (13) rival(appy, _16314) ? creep
   Call: (14) competitor(appy, _16314) ? creep
   Call: (15) company(appy) ? creep
   Exit: (15) company(appy) ? creep
   Call: (15) company(_16314) ? creep
   Exit: (15) company(sumsum) ? creep
   Call: (15) appy\=sumsum ? creep
   Exit: (15) appy\=sumsum ? creep
   Exit: (14) competitor(appy, sumsum) ? creep
   Call: (14) appy\=sumsum ? creep
   Exit: (14) appy\=sumsum ? creep
   Exit: (13) rival(appy, sumsum) ? creep
   Call: (13) develop(sumsum, galactica-s3) ? creep
   Exit: (13) develop(sumsum, galactica-s3) ? creep
   Exit: (12) unethical(stevey) ? creep
true .
```

Conclusion - Stevey is unethical.

Exercise 2 - The Royal Family:

Predicates -

1. female(X) - X is a female
2. male(X) - X is a male
3. parent_of(X,Y) - X is a parent of Y
4. child_of(X,Y) - X is child of Y
5. successor_of(X,Y) - X is successor of Y
6. predecessor_of(X,Y) - X is predecessor of Y

a. Line of Succession as per original rule -

Assumptions -

1. The order of birth of the royals is - Prince Charles, Princess Ann, Prince Andrew, Prince Edward.
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

A. Rules in Knowledge Base -

The rules defined in prolog according to the old Royal succession rule -

1. If X is male and Y is female, then X precedes Y i.e. male precedes female.
2. If X and Y both are male, and X is older than Y, then X precedes Y i.e.if both are male, older sibling precedes.
3. If X and Y are both females, and X is older than Y, then X is the predecessor of Y i.e. if both are female then older sibling precedes.

These rules define the order of succession based on gender and age. The prolog code uses these rules to determine the line of succession to the British throne after monarch Queen Elizabeth according to the old succession rule.

1. Translation of **given natural language** statements into **First Order Logic (FOL) Predicates:**

S. No.	Natural Language	First Order Logic
1.	Queen_elizabeth, the monarch of United Kingdom	female(queen_elizabeth)
2.	Has four offsprings; namely:- prince charles, princess ann, prince andrew and prince edward	female(princess_ann). male(prince_charles). male(prince_andrew).

		male(prince_edward). parent_of(queen_elizabeth, prince_charles). parent_of(queen_elizabeth, prince_andrew). parent_of(queen_elizabeth, prince_edward). parent_of(queen_elizabeth, princess_ann).
3.	Sibling relationship in order of birth	older_than(prince_charles, princess_ann). older_than(prince_charles, prince_andrew). older_than(prince_charles, prince_edward). older_than(princess_ann, prince_andrew). older_than(princess_ann, prince_edward). older_than(prince_andrew, prince_edward).

2. Rule Predicates

S. No.	Rule	First Order Logic Rule Predicate
1.	If X is male and Y is female, then X precedes Y i.e. male precedes female.	$\forall x \forall y$ predecessor_of$(X,Y) \rightarrow$ (male$(X) \wedge$ Female(Y))
2.	If X and Y both are male, and X is older than Y, then X precedes Y i.e.if both are male, older sibling precedes.	$\forall x \forall y$ predecessor_of$(X,Y) \rightarrow$ (male$(X) \wedge$ male$(Y) \wedge$ older_than(X,Y))
3.	If X and Y are both females, and X is older than Y, then X is the predecessor of Y i.e. if both are female then older sibling precedes.	$\forall x \forall y$ predecessor_of$(X,Y) \rightarrow$ (female$(X) \wedge$ female$(Y) \wedge$ older_than(X,Y))

3. Functions

S. No.	FOL Function Code Snippet	Explanation
1.	`count_predecessor(X, Count):- findall(X, predecessor_of(X, _), PredecessorList), length(PredecessorList, Count).`	● `findall` finds the list of predecessors of X (`PredecessorList`) using `predecessor_of(X,_)` predicate. ● `length (PredecessorList, Count)` finds the length of PredecessorList for each X and stores it into Count. It finds the number of monarchs who X preceded. ● For example, `count_predecessor(prince_charles, X)` gives X = 3 as X precedes 3 royals.. `count_predecessor(princess_ann, X)`. gives X = 0 because ann does not precede anyone,

2.	`line_of_succession(Monarch, SuccessorList):-`
	
	1. `findall(X-Count, (member(X, [princess_ann, prince_charles, prince_andrew, prince_edward]),count_predecessor(X, Count)), Monarch_Count_Pair),`
	2. `sort(2, @>=,Monarch_Count_Pair, SortedList),`
	
	`maplist([X-_,X]>>true, SortedList, SuccessorList).`
		Line_of_succession function is used to find the successor list of monarch.
		
		1. X-Count represents
		X: Royal Name
		Count: Number of royals X precedes
		
		`Findall` function takes the `X-Count`, from the list of royals and returns the `Monarch_Count_Pair` as
		`[(prince_charles,3), (prince_andrew,2), (prince_edward, 1), (princess_ann,0)]`
		
		2. `Sort` the `Monarch_Count_Pair` according to the second element in the pair (i.e. the `count` part).
		`@>=` indicates descending order.
		Result stored in `SortedList`.

		Therefore, `line_of_succession(queen_elizabeth,X)` gives output - `X = [prince_charles, prince_andrew, prince_edward, princess_ann]`

4. FOL Statements to Prolog clauses (Code)

```
% Define the royal family members
female(queen_elizabeth).
female(princess_ann).
male(prince_charles).
male(prince_andrew).
male(prince_edward).
male(X):- not(female(X)).

% Define parent child relationships
parent_of(queen_elizabeth, prince_charles).
parent_of(queen_elizabeth, princess_ann).
parent_of(queen_elizabeth, prince_andrew).
parent_of(queen_elizabeth, prince_edward).
child_of(X,Y):- parent_of(Y,X).
```

```
% Define sibling relationships
older_than(prince_charles, princess_ann).
older_than(prince_charles, prince_andrew).
older_than(prince_charles, prince_edward).
older_than(princess_ann, prince_andrew).
older_than(princess_ann, prince_edward).
older_than(prince_andrew, prince_edward).

% Define son daughter relationships
son_of(X, Y):- child_of(X, Y), male(X).
daughter_of(X, Y):- child_of(X, Y), female(X).

% X is successor of Y when X is child of Y.
successor_of(X,Y) :- child_of(X,Y).
```

```
% Order of precedence according to old rule.
% Define rules
predecessor_of(X,Y) :-
    % Rule1 - X predecessor of Y if X and Y are siblings and X is male and Y is female
    successor_of(X,W), successor_of(Y,W), male(X), female(Y);

    % Rule2 - X predecessor of Y if X and Y are siblings and X,Y are males and X
    older than Y
    successor_of(X,W), successor_of(Y,W), older_than(X,Y), male(X), male(Y);

    % Rule3 - X predecessor of Y if X and Y are siblings and X,Y are females and X
    older than Y
    successor_of(X,W), successor_of(Y,W), older_than(X,Y), female(X), female(Y).
```

```
% Count number of Predecessors of X
count_predecessor(X, Count):- findall(X, predecessor_of(X, _), PredecessorList),
length(PredecessorList, Count).

% Find the line of Succession of the Monarch
line_of_succession(Monarch, SuccessorList):-
    findall(X-Count, (member(X, [princess_ann, prince_charles, prince_andrew,
    prince_edward]),count_predecessor(X, Count)), Monarch_Count_Pair),
    sort(2, @>=,Monarch_Count_Pair, SortedList),
    maplist([X-_,X]>>true, SortedList, SuccessorList).
```

5. Simple Output

```
[debug] 3 ?- line_of_succession(queen_elizabeth,X).
X = [prince_charles, prince_andrew, prince_edward, princess_ann].
```

6. Trace

A trace of the Prolog program will show the order of execution and how the succession line is computed based on the old Royal succession rule.

```
[trace] 4 ?- line_of_succession(queen_elizabeth,X).
   Call: (12) line_of_succession(queen_elizabeth, _2232) ? creep
^  Call: (13) findall(_3534-_3536, (member(_3534, [princess_ann, prince_charles, prince_andrew, prince_edward]), count_p
redecessor(_3534, _3536)), _3580) ? creep
   Call: (18) lists:member(_3534, [princess_ann, prince_charles, prince_andrew, prince_edward]) ? creep
   Exit: (18) lists:member(princess_ann, [princess_ann, prince_charles, prince_andrew, prince_edward]) ? creep
   Call: (18) count_predecessor(princess_ann, _3536) ? creep
^  Call: (19) findall(princess_ann, predecessor_of(princess_ann, _6894), _6896) ? creep
   Call: (23) predecessor_of(princess_ann, _6894) ? creep
   Call: (24) successor_of(princess_ann, _8562) ? creep
   Call: (25) child_of(princess_ann, _8562) ? creep
   Call: (26) parent_of(_8562, princess_ann) ? creep
   Exit: (26) parent_of(queen_elizabeth, princess_ann) ? creep
   Exit: (25) child_of(princess_ann, queen_elizabeth) ? creep
   Exit: (24) successor_of(princess_ann, queen_elizabeth) ? creep
   Call: (24) successor_of(_6894, queen_elizabeth) ? creep
   Call: (25) child_of(_6894, queen_elizabeth) ? creep
   Call: (26) parent_of(queen_elizabeth, _6894) ? creep
   Exit: (26) parent_of(queen_elizabeth, prince_charles) ? creep
   Exit: (25) child_of(prince_charles, queen_elizabeth) ? creep
   Exit: (24) successor_of(prince_charles, queen_elizabeth) ? creep
   Call: (24) male(princess_ann) ? creep
```

```
^  Call: (25) not(female(princess_ann)) ? creep
   Call: (26) female(princess_ann) ? creep
   Exit: (26) female(princess_ann) ? creep
^  Fail: (25) not(user:female(princess_ann)) ? creep
   Fail: (24) male(princess_ann) ? creep
   Redo: (26) parent_of(queen_elizabeth, _6894) ? creep
   Exit: (26) parent_of(queen_elizabeth, princess_ann) ? creep
   Exit: (25) child_of(princess_ann, queen_elizabeth) ? creep
   Exit: (24) successor_of(princess_ann, queen_elizabeth) ? creep
   Call: (24) male(princess_ann) ? creep
^  Call: (25) not(female(princess_ann)) ? creep
   Call: (26) female(princess_ann) ? creep
   Exit: (26) female(princess_ann) ? creep
^  Fail: (25) not(user:female(princess_ann)) ? creep
   Fail: (24) male(princess_ann) ? creep
   Redo: (26) parent_of(queen_elizabeth, _198) ? creep
   Exit: (26) parent_of(queen_elizabeth, prince_andrew) ? creep
   Exit: (25) child_of(prince_andrew, queen_elizabeth) ? creep
   Exit: (24) successor_of(prince_andrew, queen_elizabeth) ? creep
   Call: (24) male(princess_ann) ? creep
^  Call: (25) not(female(princess_ann)) ? creep
   Call: (26) female(princess_ann) ? creep
   Exit: (26) female(princess_ann) ? creep
```

```
^  Fail: (25) not(user:female(princess_ann)) ? creep
   Fail: (24) male(princess_ann) ? creep
   Redo: (26) parent_of(queen_elizabeth, _198) ? creep
   Exit: (26) parent_of(queen_elizabeth, prince_edward) ? creep
   Exit: (25) child_of(prince_edward, queen_elizabeth) ? creep
   Exit: (24) successor_of(prince_edward, queen_elizabeth) ? creep
   Call: (24) male(princess_ann) ? creep
^  Call: (25) not(female(princess_ann)) ? creep
   Call: (26) female(princess_ann) ? creep
   Exit: (26) female(princess_ann) ? creep
^  Fail: (25) not(user:female(princess_ann)) ? creep
   Fail: (24) male(princess_ann) ? creep
   Redo: (23) predecessor_of(princess_ann, _198) ? creep
   Call: (24) successor_of(princess_ann, _17128) ? creep
   Call: (25) child_of(princess_ann, _17128) ? creep
   Call: (26) parent_of(_17128, princess_ann) ? creep
   Exit: (26) parent_of(queen_elizabeth, princess_ann) ? creep
   Exit: (25) child_of(princess_ann, queen_elizabeth) ? creep
   Exit: (24) successor_of(princess_ann, queen_elizabeth) ? creep
   Call: (24) successor_of(_198, queen_elizabeth) ? creep
   Call: (25) child_of(_198, queen_elizabeth) ? creep
   Call: (26) parent_of(queen_elizabeth, _198) ? creep
   Exit: (26) parent_of(queen_elizabeth, prince_charles) ? creep
   Exit: (25) child_of(prince_charles, queen_elizabeth) ? creep
   Exit: (24) successor_of(prince_charles, queen_elizabeth) ? creep
   Call: (24) older_than(princess_ann, prince_charles) ? creep
   Fail: (24) older_than(princess_ann, prince_charles) ? creep
```

```
   Redo: (26) parent_of(queen_elizabeth, _198) ? creep
   Exit: (26) parent_of(queen_elizabeth, princess_ann) ? creep
   Exit: (25) child_of(princess_ann, queen_elizabeth) ? creep
   Exit: (24) successor_of(princess_ann, queen_elizabeth) ? creep
   Call: (24) older_than(princess_ann, princess_ann) ? creep
   Fail: (24) older_than(princess_ann, princess_ann) ? creep
   Redo: (26) parent_of(queen_elizabeth, _198) ? creep
   Exit: (26) parent_of(queen_elizabeth, prince_andrew) ? creep
   Exit: (25) child_of(prince_andrew, queen_elizabeth) ? creep
   Exit: (24) successor_of(prince_andrew, queen_elizabeth) ? creep
   Call: (24) older_than(princess_ann, prince_andrew) ? creep
   Exit: (24) older_than(princess_ann, prince_andrew) ? creep
   Call: (24) male(princess_ann) ? creep
^  Call: (25) not(female(princess_ann)) ? creep
   Call: (26) female(princess_ann) ? creep
   Exit: (26) female(princess_ann) ? creep
^  Fail: (25) not(user:female(princess_ann)) ? creep
   Fail: (24) male(princess_ann) ? creep
   Redo: (24) older_than(princess_ann, prince_andrew) ? creep
   Fail: (24) older_than(princess_ann, prince_andrew) ? creep
   Redo: (26) parent_of(queen_elizabeth, _198) ? creep
   Exit: (26) parent_of(queen_elizabeth, prince_edward) ? creep
   Exit: (25) child_of(prince_edward, queen_elizabeth) ? creep
   Exit: (24) successor_of(prince_edward, queen_elizabeth) ? creep
   Call: (24) older_than(princess_ann, prince_edward) ? creep
   Exit: (24) older_than(princess_ann, prince_edward) ? creep
   Call: (24) male(princess_ann) ? creep
```

```
^  Call: (25) not(female(princess_ann)) ? creep
   Call: (26) female(princess_ann) ? creep
   Exit: (26) female(princess_ann) ? creep
^  Fail: (25) not(user:female(princess_ann)) ? creep
   Fail: (24) male(princess_ann) ? creep
   Redo: (23) predecessor_of(princess_ann, _198) ? creep
   Call: (24) successor_of(princess_ann, _23744) ? creep
   Call: (25) child_of(princess_ann, _23744) ? creep
   Call: (26) parent_of(_23744, princess_ann) ? creep
   Exit: (26) parent_of(queen_elizabeth, princess_ann) ? creep
   Exit: (25) child_of(princess_ann, queen_elizabeth) ? creep
   Exit: (24) successor_of(princess_ann, queen_elizabeth) ? creep
   Call: (24) successor_of(_198, queen_elizabeth) ? creep
   Call: (25) child_of(_198, queen_elizabeth) ? creep
   Call: (26) parent_of(queen_elizabeth, _198) ? creep
   Exit: (26) parent_of(queen_elizabeth, prince_charles) ? creep
   Exit: (25) child_of(prince_charles, queen_elizabeth) ? creep
   Exit: (24) successor_of(prince_charles, queen_elizabeth) ? creep
   Call: (24) older_than(princess_ann, prince_charles) ? creep
   Fail: (24) older_than(princess_ann, prince_charles) ? creep
   Redo: (26) parent_of(queen_elizabeth, _198) ? creep
   Exit: (26) parent_of(queen_elizabeth, princess_ann) ? creep
   Exit: (25) child_of(princess_ann, queen_elizabeth) ? creep
   Exit: (24) successor_of(princess_ann, queen_elizabeth) ? creep
   Call: (24) older_than(princess_ann, princess_ann) ? creep
   Fail: (24) older_than(princess_ann, princess_ann) ? creep
```

```
   Redo: (26) parent_of(queen_elizabeth, _198) ? creep
   Exit: (26) parent_of(queen_elizabeth, princess_ann) ? creep
   Exit: (25) child_of(princess_ann, queen_elizabeth) ? creep
   Exit: (24) successor_of(princess_ann, queen_elizabeth) ? creep
   Call: (24) older_than(princess_ann, princess_ann) ? creep
   Fail: (24) older_than(princess_ann, princess_ann) ? creep
   Redo: (26) parent_of(queen_elizabeth, _198) ? creep
   Exit: (26) parent_of(queen_elizabeth, prince_andrew) ? creep
   Exit: (25) child_of(prince_andrew, queen_elizabeth) ? creep
   Exit: (24) successor_of(prince_andrew, queen_elizabeth) ? creep
   Call: (24) older_than(princess_ann, prince_andrew) ? creep
   Exit: (24) older_than(princess_ann, prince_andrew) ? creep
   Call: (24) female(princess_ann) ? creep
   Exit: (24) female(princess_ann) ? creep
   Call: (24) female(prince_andrew) ? creep
   Fail: (24) female(prince_andrew) ? creep
   Redo: (24) older_than(princess_ann, prince_andrew) ? creep
   Fail: (24) older_than(princess_ann, prince_andrew) ? creep
   Redo: (26) parent_of(queen_elizabeth, _198) ? creep
   Exit: (26) parent_of(queen_elizabeth, prince_edward) ? creep
   Exit: (25) child_of(prince_edward, queen_elizabeth) ? creep
   Exit: (24) successor_of(prince_edward, queen_elizabeth) ? creep
   Call: (24) older_than(princess_ann, prince_edward) ? creep
   Exit: (24) older_than(princess_ann, prince_edward) ? creep
   Call: (24) female(princess_ann) ? creep
   Exit: (24) female(princess_ann) ? creep
   Call: (24) female(prince_edward) ? creep
   Fail: (24) female(prince_edward) ? creep
```

```
   Fail: (23) predecessor_of(princess_ann, _198) ? creep
^  Exit: (19) findall(princess_ann, user:predecessor_of(princess_ann, _198), []) ? creep
   Call: (19) length([], _120) ? creep
   Exit: (19) length([], 0) ? creep
   Exit: (18) count_predecessor(princess_ann, 0) ? creep
   Redo: (18) lists:member(_118, [princess_ann, prince_charles, prince_andrew, prince_edward]) ? creep
   Exit: (18) lists:member(prince_charles, [princess_ann, prince_charles, prince_andrew, prince_edward]) ? creep
   Call: (18) count_predecessor(prince_charles, _120) ? creep
^  Call: (19) findall(prince_charles, predecessor_of(prince_charles, _1802), _1804) ? creep
   Call: (23) predecessor_of(prince_charles, _1802) ? creep
   Call: (24) successor_of(prince_charles, _3470) ? creep
   Call: (25) child_of(prince_charles, _3470) ? creep
   Call: (26) parent_of(_3470, prince_charles) ? creep
   Exit: (26) parent_of(queen_elizabeth, prince_charles) ? creep
   Exit: (25) child_of(prince_charles, queen_elizabeth) ? creep
   Exit: (24) successor_of(prince_charles, queen_elizabeth) ? creep
   Call: (24) successor_of(_1802, queen_elizabeth) ? creep
   Call: (25) child_of(_1802, queen_elizabeth) ? creep
   Call: (26) parent_of(queen_elizabeth, _1802) ? creep
   Exit: (26) parent_of(queen_elizabeth, prince_charles) ? creep
   Exit: (25) child_of(prince_charles, queen_elizabeth) ? creep
   Exit: (24) successor_of(prince_charles, queen_elizabeth) ? creep
   Call: (24) male(prince_charles) ? creep
   Exit: (24) male(prince_charles) ? creep
   Call: (24) female(prince_charles) ? creep
   Fail: (24) female(prince_charles) ? creep
   Redo: (24) male(prince_charles) ? creep
```

```
^  Call: (25) not(female(prince_charles)) ? creep
   Call: (26) female(prince_charles) ? creep
   Fail: (26) female(prince_charles) ? creep
^  Exit: (25) not(user:female(prince_charles)) ? creep
   Exit: (24) male(prince_charles) ? creep
   Call: (24) female(prince_charles) ? creep
   Fail: (24) female(prince_charles) ? creep
   Redo: (26) parent_of(queen_elizabeth, _1802) ? creep
   Exit: (26) parent_of(queen_elizabeth, princess_ann) ? creep
   Exit: (25) child_of(princess_ann, queen_elizabeth) ? creep
   Exit: (24) successor_of(princess_ann, queen_elizabeth) ? creep
   Call: (24) male(prince_charles) ? creep
   Exit: (24) male(prince_charles) ? creep
   Call: (24) female(princess_ann) ? creep
   Exit: (24) female(princess_ann) ? creep
   Exit: (23) predecessor_of(prince_charles, princess_ann) ? creep
   Redo: (24) male(prince_charles) ? creep
^  Call: (25) not(female(prince_charles)) ? creep
   Call: (26) female(prince_charles) ? creep
   Fail: (26) female(prince_charles) ? creep
^  Exit: (25) not(user:female(prince_charles)) ? creep
   Exit: (24) male(prince_charles) ? creep
   Call: (24) female(princess_ann) ? creep
   Exit: (24) female(princess_ann) ? creep
   Exit: (23) predecessor_of(prince_charles, princess_ann) ? creep
```

```
   Redo: (26) parent_of(queen_elizabeth, _198) ? creep
   Exit: (26) parent_of(queen_elizabeth, prince_andrew) ? creep
   Exit: (25) child_of(prince_andrew, queen_elizabeth) ? creep
   Exit: (24) successor_of(prince_andrew, queen_elizabeth) ? creep
   Call: (24) male(prince_charles) ? creep
   Exit: (24) male(prince_charles) ? creep
   Call: (24) female(prince_andrew) ? creep
   Fail: (24) female(prince_andrew) ? creep
   Redo: (24) male(prince_charles) ? creep
^  Call: (25) not(female(prince_charles)) ? creep
   Call: (26) female(prince_charles) ? creep
   Fail: (26) female(prince_charles) ? creep
^  Exit: (25) not(user:female(prince_charles)) ? creep
   Exit: (24) male(prince_charles) ? creep
   Call: (24) female(prince_andrew) ? creep
   Fail: (24) female(prince_andrew) ? creep
   Redo: (26) parent_of(queen_elizabeth, _198) ? creep
   Exit: (26) parent_of(queen_elizabeth, prince_edward) ? creep
   Exit: (25) child_of(prince_edward, queen_elizabeth) ? creep
   Exit: (24) successor_of(prince_edward, queen_elizabeth) ? creep
   Call: (24) male(prince_charles) ? creep
   Exit: (24) male(prince_charles) ? creep
   Call: (24) female(prince_edward) ? creep
   Fail: (24) female(prince_edward) ? creep
   Redo: (24) male(prince_charles) ? creep
^  Call: (25) not(female(prince_charles)) ? creep
   Call: (26) female(prince_charles) ? creep
   Fail: (26) female(prince_charles) ? creep
```

```
^  Exit: (25) not(user:female(prince_charles)) ? creep
   Exit: (24) male(prince_charles) ? creep
   Call: (24) female(prince_edward) ? creep
   Fail: (24) female(prince_edward) ? creep
   Redo: (23) predecessor_of(prince_charles, _198) ? creep
   Call: (24) successor_of(prince_charles, _1718) ? creep
   Call: (25) child_of(prince_charles, _1718) ? creep
   Call: (26) parent_of(_1718, prince_charles) ? creep
   Exit: (26) parent_of(queen_elizabeth, prince_charles) ? creep
   Exit: (25) child_of(prince_charles, queen_elizabeth) ? creep
   Exit: (24) successor_of(prince_charles, queen_elizabeth) ? creep
   Call: (24) successor_of(_198, queen_elizabeth) ? creep
   Call: (25) child_of(_198, queen_elizabeth) ? creep
   Call: (26) parent_of(queen_elizabeth, _198) ? creep
   Exit: (26) parent_of(queen_elizabeth, prince_charles) ? creep
   Exit: (25) child_of(prince_charles, queen_elizabeth) ? creep
   Exit: (24) successor_of(prince_charles, queen_elizabeth) ? creep
   Call: (24) older_than(prince_charles, prince_charles) ? creep
   Fail: (24) older_than(prince_charles, prince_charles) ? creep
   Redo: (26) parent_of(queen_elizabeth, _198) ? creep
   Exit: (26) parent_of(queen_elizabeth, princess_ann) ? creep
   Exit: (25) child_of(princess_ann, queen_elizabeth) ? creep
   Exit: (24) successor_of(princess_ann, queen_elizabeth) ? creep
   Call: (24) older_than(prince_charles, princess_ann) ? creep
   Exit: (24) older_than(prince_charles, princess_ann) ? creep
   Call: (24) male(prince_charles) ? creep
   Exit: (24) male(prince_charles) ? creep
   Call: (24) male(princess_ann) ? creep
```

```
^  Call: (25) not(female(princess_ann)) ? creep
   Call: (26) female(princess_ann) ? creep
   Exit: (26) female(princess_ann) ? creep
^  Fail: (25) not(user:female(princess_ann)) ? creep
   Fail: (24) male(princess_ann) ? creep
   Redo: (24) male(prince_charles) ? creep
^  Call: (25) not(female(prince_charles)) ? creep
   Call: (26) female(prince_charles) ? creep
   Fail: (26) female(prince_charles) ? creep
^  Exit: (25) not(user:female(prince_charles)) ? creep
   Exit: (24) male(prince_charles) ? creep
   Call: (24) male(princess_ann) ? creep
^  Call: (25) not(female(princess_ann)) ? creep
   Call: (26) female(princess_ann) ? creep
   Exit: (26) female(princess_ann) ? creep
^  Fail: (25) not(user:female(princess_ann)) ? creep
   Fail: (24) male(princess_ann) ? creep
   Redo: (24) older_than(prince_charles, princess_ann) ? creep
   Fail: (24) older_than(prince_charles, princess_ann) ? creep
   Redo: (26) parent_of(queen_elizabeth, _198) ? creep
   Exit: (26) parent_of(queen_elizabeth, prince_andrew) ? creep
   Exit: (25) child_of(prince_andrew, queen_elizabeth) ? creep
   Exit: (24) successor_of(prince_andrew, queen_elizabeth) ? creep
   Call: (24) older_than(prince_charles, prince_andrew) ? creep
   Exit: (24) older_than(prince_charles, prince_andrew) ? creep
   Call: (24) male(prince_charles) ? creep
   Exit: (24) male(prince_charles) ? creep
   Call: (24) male(prince_andrew) ? creep
   Exit: (24) male(prince_andrew) ? creep
   Exit: (23) predecessor_of(prince_charles, prince_andrew) ? creep
```

```
   Redo: (24) male(prince_andrew) ? creep
^  Call: (25) not(female(prince_andrew)) ? creep
   Call: (26) female(prince_andrew) ? creep
   Fail: (26) female(prince_andrew) ? creep
^  Exit: (25) not(user:female(prince_andrew)) ? creep
   Exit: (24) male(prince_andrew) ? creep
   Exit: (23) predecessor_of(prince_charles, prince_andrew) ? creep
   Redo: (24) male(prince_charles) ? creep
^  Call: (25) not(female(prince_charles)) ? creep
   Call: (26) female(prince_charles) ? creep
   Fail: (26) female(prince_charles) ? creep
^  Exit: (25) not(user:female(prince_charles)) ? creep
   Exit: (24) male(prince_charles) ? creep
   Call: (24) male(prince_andrew) ? creep
   Exit: (24) male(prince_andrew) ? creep
   Exit: (23) predecessor_of(prince_charles, prince_andrew) ? creep
   Redo: (24) male(prince_andrew) ? creep
^  Call: (25) not(female(prince_andrew)) ? creep
   Call: (26) female(prince_andrew) ? creep
   Fail: (26) female(prince_andrew) ? creep
^  Exit: (25) not(user:female(prince_andrew)) ? creep
   Exit: (24) male(prince_andrew) ? creep
   Exit: (23) predecessor_of(prince_charles, prince_andrew) ? creep
   Redo: (24) older_than(prince_charles, prince_andrew) ? creep
   Fail: (24) older_than(prince_charles, prince_andrew) ? creep
```

```
   Redo: (26) parent_of(queen_elizabeth, _198) ? creep
   Exit: (26) parent_of(queen_elizabeth, prince_edward) ? creep
   Exit: (25) child_of(prince_edward, queen_elizabeth) ? creep
   Exit: (24) successor_of(prince_edward, queen_elizabeth) ? creep
   Call: (24) older_than(prince_charles, prince_edward) ? creep
   Exit: (24) older_than(prince_charles, prince_edward) ? creep
   Call: (24) male(prince_charles) ? creep
   Exit: (24) male(prince_charles) ? creep
   Call: (24) male(prince_edward) ? creep
   Exit: (24) male(prince_edward) ? creep
   Exit: (23) predecessor_of(prince_charles, prince_edward) ? creep
   Redo: (24) male(prince_edward) ? creep
^  Call: (25) not(female(prince_edward)) ? creep
   Call: (26) female(prince_edward) ? creep
   Fail: (26) female(prince_edward) ? creep
^  Exit: (25) not(user:female(prince_edward)) ? creep
   Exit: (24) male(prince_edward) ? creep
   Exit: (23) predecessor_of(prince_charles, prince_edward) ? creep
   Redo: (24) male(prince_charles) ? creep
^  Call: (25) not(female(prince_charles)) ? creep
   Call: (26) female(prince_charles) ? creep
   Fail: (26) female(prince_charles) ? creep
^  Exit: (25) not(user:female(prince_charles)) ? creep
   Exit: (24) male(prince_charles) ? creep
   Call: (24) male(prince_edward) ? creep
   Exit: (24) male(prince_edward) ? creep
   Exit: (23) predecessor_of(prince_charles, prince_edward) ? creep
   Redo: (24) male(prince_edward) ? creep
```

```
^  Call: (25) not(female(prince_edward)) ? creep
   Call: (26) female(prince_edward) ? creep
   Fail: (26) female(prince_edward) ? creep
^  Exit: (25) not(user:female(prince_edward)) ? creep
   Exit: (24) male(prince_edward) ? creep
   Exit: (23) predecessor_of(prince_charles, prince_edward) ? creep
   Redo: (23) predecessor_of(prince_charles, _198) ? creep
   Call: (24) successor_of(prince_charles, _30676) ? creep
   Call: (25) child_of(prince_charles, _230) ? creep
   Call: (26) parent_of(_230, prince_charles) ? creep
   Exit: (26) parent_of(queen_elizabeth, prince_charles) ? creep
   Exit: (25) child_of(prince_charles, queen_elizabeth) ? creep
   Exit: (24) successor_of(prince_charles, queen_elizabeth) ? creep
   Call: (24) successor_of(_198, queen_elizabeth) ? creep
   Call: (25) child_of(_198, queen_elizabeth) ? creep
   Call: (26) parent_of(queen_elizabeth, _198) ? creep
   Exit: (26) parent_of(queen_elizabeth, prince_charles) ? creep
   Exit: (25) child_of(prince_charles, queen_elizabeth) ? creep
   Exit: (24) successor_of(prince_charles, queen_elizabeth) ? creep
   Call: (24) older_than(prince_charles, prince_charles) ? creep
   Fail: (24) older_than(prince_charles, prince_charles) ? creep
```

```
   Redo: (26) parent_of(queen_elizabeth, _198) ? creep
   Exit: (26) parent_of(queen_elizabeth, princess_ann) ? creep
   Exit: (25) child_of(princess_ann, queen_elizabeth) ? creep
   Exit: (24) successor_of(princess_ann, queen_elizabeth) ? creep
   Call: (24) older_than(prince_charles, princess_ann) ? creep
   Exit: (24) older_than(prince_charles, princess_ann) ? creep
   Call: (24) female(prince_charles) ? creep
   Fail: (24) female(prince_charles) ? creep
   Redo: (24) older_than(prince_charles, princess_ann) ? creep
   Fail: (24) older_than(prince_charles, princess_ann) ? creep
   Redo: (26) parent_of(queen_elizabeth, _198) ? creep
   Exit: (26) parent_of(queen_elizabeth, prince_andrew) ? creep
   Exit: (25) child_of(prince_andrew, queen_elizabeth) ? creep
   Exit: (24) successor_of(prince_andrew, queen_elizabeth) ? creep
   Call: (24) older_than(prince_charles, prince_andrew) ? creep
   Exit: (24) older_than(prince_charles, prince_andrew) ? creep
   Call: (24) female(prince_charles) ? creep
   Fail: (24) female(prince_charles) ? creep
   Redo: (24) older_than(prince_charles, prince_andrew) ? creep
   Fail: (24) older_than(prince_charles, prince_andrew) ? creep
   Redo: (26) parent_of(queen_elizabeth, _198) ? creep
   Exit: (26) parent_of(queen_elizabeth, prince_edward) ? creep
   Exit: (25) child_of(prince_edward, queen_elizabeth) ? creep
   Exit: (24) successor_of(prince_edward, queen_elizabeth) ? creep
   Call: (24) older_than(prince_charles, prince_edward) ? creep
   Exit: (24) older_than(prince_charles, prince_edward) ? creep
   Call: (24) female(prince_charles) ? creep
   Fail: (24) female(prince_charles) ? creep
```

```
   Fail: (23) predecessor_of(prince_charles, _198) ? creep
^  Exit: (19) findall(prince_charles, user:predecessor_of(prince_charles, _198), [prince_charles, prince_charles, prince
_charles, prince_charles, prince_charles, prince_charles, prince_charles, prince_charles|...]) ? creep
   Call: (19) length([prince_charles, prince_charles, prince_charles, prince_charles, prince_charles, prince_charles, pr
ince_charles, prince_charles|...], _120) ? creep
   Exit: (19) length([prince_charles, prince_charles, prince_charles, prince_charles, prince_charles, prince_charles, pr
ince_charles, prince_charles|...], 10) ? creep
   Exit: (18) count_predecessor(prince_charles, 10) ? creep
   Redo: (18) lists:member(_118, [princess_ann, prince_charles, prince_andrew, prince_edward]) ? creep
   Exit: (18) lists:member(prince_andrew, [princess_ann, prince_charles, prince_andrew, prince_edward]) ? creep
   Call: (18) count_predecessor(prince_andrew, _120) ? creep
^  Call: (19) findall(prince_andrew, predecessor_of(prince_andrew, _8802), _8804) ? creep
   Call: (23) predecessor_of(prince_andrew, _8802) ? creep
   Call: (24) successor_of(prince_andrew, _10470) ? creep
   Call: (25) child_of(prince_andrew, _10470) ? creep
   Call: (26) parent_of(_10470, prince_andrew) ? creep
   Exit: (26) parent_of(queen_elizabeth, prince_andrew) ? creep
   Exit: (25) child_of(prince_andrew, queen_elizabeth) ? creep
   Exit: (24) successor_of(prince_andrew, queen_elizabeth) ? creep
   Call: (24) successor_of(_8802, queen_elizabeth) ? creep
   Call: (25) child_of(_8802, queen_elizabeth) ? creep
   Call: (26) parent_of(queen_elizabeth, _8802) ? creep
   Exit: (26) parent_of(queen_elizabeth, prince_charles) ? creep
   Exit: (25) child_of(prince_charles, queen_elizabeth) ? creep
   Exit: (24) successor_of(prince_charles, queen_elizabeth) ? creep
   Call: (24) male(prince_andrew) ? creep
   Exit: (24) male(prince_andrew) ? creep
   Call: (24) female(prince_charles) ? creep
   Fail: (24) female(prince_charles) ? creep
```

```
   Redo: (24) male(prince_andrew) ? creep
^  Call: (25) not(female(prince_andrew)) ? creep
   Call: (26) female(prince_andrew) ? creep
   Fail: (26) female(prince_andrew) ? creep
^  Exit: (25) not(user:female(prince_andrew)) ? creep
   Exit: (24) male(prince_andrew) ? creep
   Call: (24) female(prince_charles) ? creep
   Fail: (24) female(prince_charles) ? creep
   Redo: (26) parent_of(queen_elizabeth, _8802) ? creep
   Exit: (26) parent_of(queen_elizabeth, princess_ann) ? creep
   Exit: (25) child_of(princess_ann, queen_elizabeth) ? creep
   Exit: (24) successor_of(princess_ann, queen_elizabeth) ? creep
   Call: (24) male(prince_andrew) ? creep
   Exit: (24) male(prince_andrew) ? creep
   Call: (24) female(princess_ann) ? creep
   Exit: (24) female(princess_ann) ? creep
   Exit: (23) predecessor_of(prince_andrew, princess_ann) ? creep
   Redo: (24) male(prince_andrew) ? creep
^  Call: (25) not(female(prince_andrew)) ? creep
   Call: (26) female(prince_andrew) ? creep
   Fail: (26) female(prince_andrew) ? creep
^  Exit: (25) not(user:female(prince_andrew)) ? creep
   Exit: (24) male(prince_andrew) ? creep
   Call: (24) female(princess_ann) ? creep
   Exit: (24) female(princess_ann) ? creep
   Exit: (23) predecessor_of(prince_andrew, princess_ann) ? creep
```

```
   Redo: (26) parent_of(queen_elizabeth, _198) ? creep
   Exit: (26) parent_of(queen_elizabeth, prince_andrew) ? creep
   Exit: (25) child_of(prince_andrew, queen_elizabeth) ? creep
   Exit: (24) successor_of(prince_andrew, queen_elizabeth) ? creep
   Call: (24) male(prince_andrew) ? creep
   Exit: (24) male(prince_andrew) ? creep
   Call: (24) female(prince_andrew) ? creep
   Fail: (24) female(prince_andrew) ? creep
   Redo: (24) male(prince_andrew) ? creep
^  Call: (25) not(female(prince_andrew)) ? creep
   Call: (26) female(prince_andrew) ? creep
   Fail: (26) female(prince_andrew) ? creep
^  Exit: (25) not(user:female(prince_andrew)) ? creep
   Exit: (24) male(prince_andrew) ? creep
   Call: (24) female(prince_andrew) ? creep
   Fail: (24) female(prince_andrew) ? creep
   Redo: (26) parent_of(queen_elizabeth, _198) ? creep
   Exit: (26) parent_of(queen_elizabeth, prince_edward) ? creep
   Exit: (25) child_of(prince_edward, queen_elizabeth) ? creep
   Exit: (24) successor_of(prince_edward, queen_elizabeth) ? creep
   Call: (24) male(prince_andrew) ? creep
   Exit: (24) male(prince_andrew) ? creep
   Call: (24) female(prince_edward) ? creep
   Fail: (24) female(prince_edward) ? creep
   Redo: (24) male(prince_andrew) ? creep
^  Call: (25) not(female(prince_andrew)) ? creep
   Call: (26) female(prince_andrew) ? creep
   Fail: (26) female(prince_andrew) ? creep
```

```
^  Exit: (25) not(user:female(prince_andrew)) ? creep
   Exit: (24) male(prince_andrew) ? creep
   Call: (24) female(prince_edward) ? creep
   Fail: (24) female(prince_edward) ? creep
   Redo: (23) predecessor_of(prince_andrew, _198) ? creep
   Call: (24) successor_of(prince_andrew, _8698) ? creep
   Call: (25) child_of(prince_andrew, _8698) ? creep
   Call: (26) parent_of(_8698, prince_andrew) ? creep
   Exit: (26) parent_of(queen_elizabeth, prince_andrew) ? creep
   Exit: (25) child_of(prince_andrew, queen_elizabeth) ? creep
   Exit: (24) successor_of(prince_andrew, queen_elizabeth) ? creep
   Call: (24) successor_of(_198, queen_elizabeth) ? creep
   Call: (25) child_of(_198, queen_elizabeth) ? creep
   Call: (26) parent_of(queen_elizabeth, _198) ? creep
   Exit: (26) parent_of(queen_elizabeth, prince_charles) ? creep
   Exit: (25) child_of(prince_charles, queen_elizabeth) ? creep
   Exit: (24) successor_of(prince_charles, queen_elizabeth) ? creep
   Call: (24) older_than(prince_andrew, prince_charles) ? creep
   Fail: (24) older_than(prince_andrew, prince_charles) ? creep
   Redo: (26) parent_of(queen_elizabeth, _198) ? creep
   Exit: (26) parent_of(queen_elizabeth, princess_ann) ? creep
   Exit: (25) child_of(princess_ann, queen_elizabeth) ? creep
   Exit: (24) successor_of(princess_ann, queen_elizabeth) ? creep
   Call: (24) older_than(prince_andrew, princess_ann) ? creep
   Fail: (24) older_than(prince_andrew, princess_ann) ? creep
```

```
   Redo: (26) parent_of(queen_elizabeth, _198) ? creep
   Exit: (26) parent_of(queen_elizabeth, prince_andrew) ? creep
   Exit: (25) child_of(prince_andrew, queen_elizabeth) ? creep
   Exit: (24) successor_of(prince_andrew, queen_elizabeth) ? creep
   Call: (24) older_than(prince_andrew, prince_andrew) ? creep
   Fail: (24) older_than(prince_andrew, prince_andrew) ? creep
   Redo: (26) parent_of(queen_elizabeth, _198) ? creep
   Exit: (26) parent_of(queen_elizabeth, prince_edward) ? creep
   Exit: (25) child_of(prince_edward, queen_elizabeth) ? creep
   Exit: (24) successor_of(prince_edward, queen_elizabeth) ? creep
   Call: (24) older_than(prince_andrew, prince_edward) ? creep
   Exit: (24) older_than(prince_andrew, prince_edward) ? creep
   Call: (24) male(prince_andrew) ? creep
   Exit: (24) male(prince_andrew) ? creep
   Call: (24) male(prince_edward) ? creep
   Exit: (24) male(prince_edward) ? creep
   Exit: (23) predecessor_of(prince_andrew, prince_edward) ? creep
   Redo: (24) male(prince_edward) ? creep
^  Call: (25) not(female(prince_edward)) ? creep
   Call: (26) female(prince_edward) ? creep
   Fail: (26) female(prince_edward) ? creep
^  Exit: (25) not(user:female(prince_edward)) ? creep
   Exit: (24) male(prince_edward) ? creep
   Exit: (23) predecessor_of(prince_andrew, prince_edward) ? creep
   Redo: (24) male(prince_andrew) ? creep
```

```
^  Call: (25) not(female(prince_andrew)) ? creep
   Call: (26) female(prince_andrew) ? creep
   Fail: (26) female(prince_andrew) ? creep
^  Exit: (25) not(user:female(prince_andrew)) ? creep
   Exit: (24) male(prince_andrew) ? creep
   Call: (24) male(prince_edward) ? creep
   Exit: (24) male(prince_edward) ? creep
   Exit: (23) predecessor_of(prince_andrew, prince_edward) ? creep
   Redo: (24) male(prince_edward) ? creep
^  Call: (25) not(female(prince_edward)) ? creep
   Call: (26) female(prince_edward) ? creep
   Fail: (26) female(prince_edward) ? creep
^  Exit: (25) not(user:female(prince_edward)) ? creep
   Exit: (24) male(prince_edward) ? creep
   Exit: (23) predecessor_of(prince_andrew, prince_edward) ? creep
```

```
   Redo: (24) male(prince_edward) ? creep
^  Call: (25) not(female(prince_edward)) ? creep
   Call: (26) female(prince_edward) ? creep
   Fail: (26) female(prince_edward) ? creep
^  Exit: (25) not(user:female(prince_edward)) ? creep
   Exit: (24) male(prince_edward) ? creep
   Exit: (23) predecessor_of(prince_andrew, prince_edward) ? creep
   Redo: (23) predecessor_of(prince_andrew, _198) ? creep
   Call: (24) successor_of(prince_andrew, _26866) ? creep
   Call: (25) child_of(prince_andrew, _26866) ? creep
   Call: (26) parent_of(_26866, prince_andrew) ? creep
   Exit: (26) parent_of(queen_elizabeth, prince_andrew) ? creep
   Exit: (25) child_of(prince_andrew, queen_elizabeth) ? creep
   Exit: (24) successor_of(prince_andrew, queen_elizabeth) ? creep
   Call: (24) successor_of(_198, queen_elizabeth) ? creep
   Call: (25) child_of(_198, queen_elizabeth) ? creep
   Call: (26) parent_of(queen_elizabeth, _198) ? creep
   Exit: (26) parent_of(queen_elizabeth, prince_charles) ? creep
   Exit: (25) child_of(prince_charles, queen_elizabeth) ? creep
   Exit: (24) successor_of(prince_charles, queen_elizabeth) ? creep
   Call: (24) older_than(prince_andrew, prince_charles) ? creep
   Fail: (24) older_than(prince_andrew, prince_charles) ? creep
```

```
   Redo: (26) parent_of(queen_elizabeth, _198) ? creep
   Exit: (26) parent_of(queen_elizabeth, princess_ann) ? creep
   Exit: (25) child_of(princess_ann, queen_elizabeth) ? creep
   Exit: (24) successor_of(princess_ann, queen_elizabeth) ? creep
   Call: (24) older_than(prince_andrew, princess_ann) ? creep
   Fail: (24) older_than(prince_andrew, princess_ann) ? creep
   Redo: (26) parent_of(queen_elizabeth, _198) ? creep
   Exit: (26) parent_of(queen_elizabeth, prince_andrew) ? creep
   Exit: (25) child_of(prince_andrew, queen_elizabeth) ? creep
   Exit: (24) successor_of(prince_andrew, queen_elizabeth) ? creep
   Call: (24) older_than(prince_andrew, prince_andrew) ? creep
   Fail: (24) older_than(prince_andrew, prince_andrew) ? creep
   Redo: (26) parent_of(queen_elizabeth, _198) ? creep
   Exit: (26) parent_of(queen_elizabeth, prince_edward) ? creep
   Exit: (25) child_of(prince_edward, queen_elizabeth) ? creep
   Exit: (24) successor_of(prince_edward, queen_elizabeth) ? creep
   Call: (24) older_than(prince_andrew, prince_edward) ? creep
   Exit: (24) older_than(prince_andrew, prince_edward) ? creep
   Call: (24) female(prince_andrew) ? creep
   Fail: (24) female(prince_andrew) ? creep
   Fail: (23) predecessor_of(prince_andrew, _198) ? creep
^  Exit: (19) findall(prince_andrew, user:predecessor_of(prince_andrew, _198), [prince_andrew, prince_andrew, prince_and
rew, prince_andrew, prince_andrew, prince_andrew]) ? creep
   Call: (19) length([prince_andrew, prince_andrew, prince_andrew, prince_andrew, prince_andrew, prince_andrew], _120) ?
 creep
   Exit: (19) length([prince_andrew, prince_andrew, prince_andrew, prince_andrew, prince_andrew, prince_andrew], 6) ? cr
eep
   Exit: (18) count_predecessor(prince_andrew, 6) ? creep
```

```
   Redo: (18) lists:member(_118, [princess_ann, prince_charles, prince_andrew, prince_edward]) ? creep
   Exit: (18) lists:member(prince_edward, [princess_ann, prince_charles, prince_andrew, prince_edward]) ? creep
   Call: (18) count_predecessor(prince_edward, _120) ? creep
^  Call: (19) findall(prince_edward, predecessor_of(prince_edward, _29518), _29520) ? creep
   Call: (23) predecessor_of(prince_edward, _29518) ? creep
   Call: (24) successor_of(prince_edward, _232) ? creep
   Call: (25) child_of(prince_edward, _232) ? creep
   Call: (26) parent_of(_232, prince_edward) ? creep
   Exit: (26) parent_of(queen_elizabeth, prince_edward) ? creep
   Exit: (25) child_of(prince_edward, queen_elizabeth) ? creep
   Exit: (24) successor_of(prince_edward, queen_elizabeth) ? creep
   Call: (24) successor_of(_198, queen_elizabeth) ? creep
   Call: (25) child_of(_198, queen_elizabeth) ? creep
   Call: (26) parent_of(queen_elizabeth, _198) ? creep
   Exit: (26) parent_of(queen_elizabeth, prince_charles) ? creep
   Exit: (25) child_of(prince_charles, queen_elizabeth) ? creep
   Exit: (24) successor_of(prince_charles, queen_elizabeth) ? creep
   Call: (24) male(prince_edward) ? creep
   Exit: (24) male(prince_edward) ? creep
   Call: (24) female(prince_charles) ? creep
   Fail: (24) female(prince_charles) ? creep
   Redo: (24) male(prince_edward) ? creep
^  Call: (25) not(female(prince_edward)) ? creep
   Call: (26) female(prince_edward) ? creep
   Fail: (26) female(prince_edward) ? creep
```

```
^  Exit: (25) not(user:female(prince_edward)) ? creep
   Exit: (24) male(prince_edward) ? creep
   Call: (24) female(prince_charles) ? creep
   Fail: (24) female(prince_charles) ? creep
   Redo: (26) parent_of(queen_elizabeth, _198) ? creep
   Exit: (26) parent_of(queen_elizabeth, princess_ann) ? creep
   Exit: (25) child_of(princess_ann, queen_elizabeth) ? creep
   Exit: (24) successor_of(princess_ann, queen_elizabeth) ? creep
   Call: (24) male(prince_edward) ? creep
   Exit: (24) male(prince_edward) ? creep
   Call: (24) female(princess_ann) ? creep
   Exit: (24) female(princess_ann) ? creep
   Exit: (23) predecessor_of(prince_edward, princess_ann) ? creep
   Redo: (24) male(prince_edward) ? creep
^  Call: (25) not(female(prince_edward)) ? creep
   Call: (26) female(prince_edward) ? creep
   Fail: (26) female(prince_edward) ? creep
^  Exit: (25) not(user:female(prince_edward)) ? creep
   Exit: (24) male(prince_edward) ? creep
   Call: (24) female(princess_ann) ? creep
   Exit: (24) female(princess_ann) ? creep
   Exit: (23) predecessor_of(prince_edward, princess_ann) ? creep
```

```
   Redo: (26) parent_of(queen_elizabeth, _198) ? creep
   Exit: (26) parent_of(queen_elizabeth, prince_andrew) ? creep
   Exit: (25) child_of(prince_andrew, queen_elizabeth) ? creep
   Exit: (24) successor_of(prince_andrew, queen_elizabeth) ? creep
   Call: (24) male(prince_edward) ? creep
   Exit: (24) male(prince_edward) ? creep
   Call: (24) female(prince_andrew) ? creep
   Fail: (24) female(prince_andrew) ? creep
   Redo: (24) male(prince_edward) ? creep
^  Call: (25) not(female(prince_edward)) ? creep
   Call: (26) female(prince_edward) ? creep
   Fail: (26) female(prince_edward) ? creep
^  Exit: (25) not(user:female(prince_edward)) ? creep
   Exit: (24) male(prince_edward) ? creep
   Call: (24) female(prince_andrew) ? creep
   Fail: (24) female(prince_andrew) ? creep
   Redo: (26) parent_of(queen_elizabeth, _198) ? creep
   Exit: (26) parent_of(queen_elizabeth, prince_edward) ? creep
   Exit: (25) child_of(prince_edward, queen_elizabeth) ? creep
   Exit: (24) successor_of(prince_edward, queen_elizabeth) ? creep
   Call: (24) male(prince_edward) ? creep
   Exit: (24) male(prince_edward) ? creep
   Call: (24) female(prince_edward) ? creep
   Fail: (24) female(prince_edward) ? creep
   Redo: (24) male(prince_edward) ? creep
```

```
^  Call: (25) not(female(prince_edward)) ? creep
   Call: (26) female(prince_edward) ? creep
   Fail: (26) female(prince_edward) ? creep
^  Exit: (25) not(user:female(prince_edward)) ? creep
   Exit: (24) male(prince_edward) ? creep
   Call: (24) female(prince_edward) ? creep
   Fail: (24) female(prince_edward) ? creep
   Redo: (23) predecessor_of(prince_edward, _198) ? creep
   Call: (24) successor_of(prince_edward, _29352) ? creep
   Call: (25) child_of(prince_edward, _29352) ? creep
   Call: (26) parent_of(_230, prince_edward) ? creep
   Exit: (26) parent_of(queen_elizabeth, prince_edward) ? creep
   Exit: (25) child_of(prince_edward, queen_elizabeth) ? creep
   Exit: (24) successor_of(prince_edward, queen_elizabeth) ? creep
   Call: (24) successor_of(_198, queen_elizabeth) ? creep
   Call: (25) child_of(_198, queen_elizabeth) ? creep
   Call: (26) parent_of(queen_elizabeth, _198) ? creep
   Exit: (26) parent_of(queen_elizabeth, prince_charles) ? creep
   Exit: (25) child_of(prince_charles, queen_elizabeth) ? creep
   Exit: (24) successor_of(prince_charles, queen_elizabeth) ? creep
   Call: (24) older_than(prince_edward, prince_charles) ? creep
   Fail: (24) older_than(prince_edward, prince_charles) ? creep
```

```
   Redo: (26) parent_of(queen_elizabeth, _198) ? creep
   Exit: (26) parent_of(queen_elizabeth, princess_ann) ? creep
   Exit: (25) child_of(princess_ann, queen_elizabeth) ? creep
   Exit: (24) successor_of(princess_ann, queen_elizabeth) ? creep
   Call: (24) older_than(prince_edward, princess_ann) ? creep
   Fail: (24) older_than(prince_edward, princess_ann) ? creep
   Redo: (26) parent_of(queen_elizabeth, _198) ? creep
   Exit: (26) parent_of(queen_elizabeth, prince_andrew) ? creep
   Exit: (25) child_of(prince_andrew, queen_elizabeth) ? creep
   Exit: (24) successor_of(prince_andrew, queen_elizabeth) ? creep
   Call: (24) older_than(prince_edward, prince_andrew) ? creep
   Fail: (24) older_than(prince_edward, prince_andrew) ? creep
   Redo: (26) parent_of(queen_elizabeth, _198) ? creep
   Exit: (26) parent_of(queen_elizabeth, prince_edward) ? creep
   Exit: (25) child_of(prince_edward, queen_elizabeth) ? creep
   Exit: (24) successor_of(prince_edward, queen_elizabeth) ? creep
   Call: (24) older_than(prince_edward, prince_edward) ? creep
   Fail: (24) older_than(prince_edward, prince_edward) ? creep
```

```
   Redo: (23) predecessor_of(prince_edward, _198) ? creep
   Call: (24) successor_of(prince_edward, _24902) ? creep
   Call: (25) child_of(prince_edward, _24902) ? creep
   Call: (26) parent_of(_24902, prince_edward) ? creep
   Exit: (26) parent_of(queen_elizabeth, prince_edward) ? creep
   Exit: (25) child_of(prince_edward, queen_elizabeth) ? creep
   Exit: (24) successor_of(prince_edward, queen_elizabeth) ? creep
   Call: (24) successor_of(_198, queen_elizabeth) ? creep
   Call: (25) child_of(_198, queen_elizabeth) ? creep
   Call: (26) parent_of(queen_elizabeth, _198) ? creep
   Exit: (26) parent_of(queen_elizabeth, prince_charles) ? creep
   Exit: (25) child_of(prince_charles, queen_elizabeth) ? creep
   Exit: (24) successor_of(prince_charles, queen_elizabeth) ? creep
   Call: (24) older_than(prince_edward, prince_charles) ? creep
   Fail: (24) older_than(prince_edward, prince_charles) ? creep
   Redo: (26) parent_of(queen_elizabeth, _198) ? creep
   Exit: (26) parent_of(queen_elizabeth, princess_ann) ? creep
   Exit: (25) child_of(princess_ann, queen_elizabeth) ? creep
   Exit: (24) successor_of(princess_ann, queen_elizabeth) ? creep
   Call: (24) older_than(prince_edward, princess_ann) ? creep
   Fail: (24) older_than(prince_edward, princess_ann) ? creep
   Redo: (26) parent_of(queen_elizabeth, _198) ? creep
   Exit: (26) parent_of(queen_elizabeth, prince_andrew) ? creep
   Exit: (25) child_of(prince_andrew, queen_elizabeth) ? creep
   Exit: (24) successor_of(prince_andrew, queen_elizabeth) ? creep
   Call: (24) older_than(prince_edward, prince_andrew) ? creep
   Fail: (24) older_than(prince_edward, prince_andrew) ? creep
```

```
   Redo: (26) parent_of(queen_elizabeth, _198) ? creep
   Exit: (26) parent_of(queen_elizabeth, prince_edward) ? creep
   Exit: (25) child_of(prince_edward, queen_elizabeth) ? creep
   Exit: (24) successor_of(prince_edward, queen_elizabeth) ? creep
   Call: (24) older_than(prince_edward, prince_edward) ? creep
   Fail: (24) older_than(prince_edward, prince_edward) ? creep
   Fail: (23) predecessor_of(prince_edward, _198) ? creep
^  Exit: (19) findall(prince_edward, user:predecessor_of(prince_edward, _198), [prince_edward, prince_edward]) ? creep
   Call: (19) length([prince_edward, prince_edward], _120) ? creep
   Exit: (19) length([prince_edward, prince_edward], 2) ? creep
   Exit: (18) count_predecessor(prince_edward, 2) ? creep
^  Exit: (13) findall(_118-_120, user:(member(_118, [princess_ann, prince_charles, prince_andrew, prince_edward]), count
_predecessor(_118, _120)), [princess_ann-0, prince_charles-10, prince_andrew-6, prince_edward-2]) ? creep
   Call: (13) sort(2, @>=, [princess_ann-0, prince_charles-10, prince_andrew-6, prince_edward-2], _24532) ? creep
   Exit: (13) sort(2, @>=, [princess_ann-0, prince_charles-10, prince_andrew-6, prince_edward-2], [prince_charles-10, pr
ince_andrew-6, prince_edward-2, princess_ann-0]) ? creep
^  Call: (13) apply:maplist([_118-_26210, _118]>>true, [prince_charles-10, prince_andrew-6, prince_edward-2, princess_an
n-0], _58) ? creep
^  Call: (15) yall: >>([_118-_26210, _118], true, prince_charles-10, _27058) ? creep
   Call: (16) yall:unify_lambda_parameters([_118-_26210, _118], [prince_charles-10, _27058], _27916, user:true, _27918)
? creep
   Call: (17) var([_118-_26210, _118]) ? creep
   Fail: (17) var([_118-_26210, _118]) ? creep
```

```
   Redo: (16) yall:unify_lambda_parameters([_118-_26210, _118], [prince_charles-10, _27058], _27916, user:true, _27918)
? creep
   Call: (17) error:must_be(list, [_116-_212, _116]) ? creep
   Exit: (17) error:must_be(list, [_116-_212, _116]) ? creep
   Call: (17) copy_term_nat([_116-_212, _116]>>(user:true), _1694>>_252) ? creep
   Exit: (17) copy_term_nat([_116-_212, _116]>>(user:true), [_2524-_2526, _2524]>>(user:true)) ? creep
   Call: (17) yall:unify_lambda_parameters_([_2524-_2526, _2524], [prince_charles-10, _228], _250, [_116-_212, _116]>>(u
ser:true)) ? creep
   Call: (18) _2524-_2526=prince_charles-10 ? creep
   Exit: (18) prince_charles-10=prince_charles-10 ? creep
   Call: (18) yall:unify_lambda_parameters_([prince_charles], [_228], _250, [_116-_212, _116]>>(user:true)) ? creep
   Call: (19) prince_charles=_228 ? creep
   Exit: (19) prince_charles=prince_charles ? creep
   Call: (19) yall:unify_lambda_parameters_([], [], _250, [_116-_212, _116]>>(user:true)) ? creep
   Exit: (19) yall:unify_lambda_parameters_([], [], [], [_116-_212, _116]>>(user:true)) ? creep
   Exit: (18) yall:unify_lambda_parameters_([prince_charles], [prince_charles], [], [_116-_212, _116]>>(user:true)) ? cr
eep
   Exit: (17) yall:unify_lambda_parameters_([prince_charles-10, prince_charles], [prince_charles-10, prince_charles], []
, [_116-_212, _116]>>(user:true)) ? creep
   Exit: (16) yall:unify_lambda_parameters([_116-_212, _116], [prince_charles-10, prince_charles], [], user:true, user:t
rue) ? creep
   Call: (16) _12404=..[call, user:true] ? creep
   Exit: (16) call(user:true)=..[call, user:true] ? creep
```

```
^  Call: (16) call(user:true) ? creep
^  Exit: (16) call(user:true) ? creep
^  Exit: (15) yall: >>([_116-_212, _116], user:true, prince_charles-10, prince_charles) ? creep
^  Call: (16) yall: >>([_116-_212, _116], true, prince_andrew-6, _16516) ? creep
   Call: (17) yall:unify_lambda_parameters([_116-_212, _116], [prince_andrew-6, _16516], _17374, user:true, _17376) ? cr
eep
   Call: (18) var([_116-_212, _116]) ? creep
   Fail: (18) var([_116-_212, _116]) ? creep
   Redo: (17) yall:unify_lambda_parameters([_116-_212, _116], [prince_andrew-6, _16516], _17374, user:true, _17376) ? cr
eep
   Call: (18) error:must_be(list, [_116-_212, _116]) ? creep
   Exit: (18) error:must_be(list, [_116-_212, _116]) ? creep
   Call: (18) copy_term_nat([_116-_212, _116]>>(user:true), _22310>>_17376) ? creep
   Exit: (18) copy_term_nat([_116-_212, _116]>>(user:true), [_23140-_23142, _23140]>>(user:true)) ? creep
   Call: (18) yall:unify_lambda_parameters_([_23140-_23142, _23140], [prince_andrew-6, _16516], _17374, [_116-_212, _116
]>>(user:true)) ? creep
   Call: (19) _23140-_23142=prince_andrew-6 ? creep
   Exit: (19) prince_andrew-6=prince_andrew-6 ? creep
   Call: (19) yall:unify_lambda_parameters_([prince_andrew], [_16516], _17374, [_116-_212, _116]>>(user:true)) ? creep
   Call: (20) prince_andrew=_16516 ? creep
   Exit: (20) prince_andrew=prince_andrew ? creep
   Call: (20) yall:unify_lambda_parameters_([], [], _17374, [_116-_212, _116]>>(user:true)) ? creep
   Exit: (20) yall:unify_lambda_parameters_([], [], [], [_116-_212, _116]>>(user:true)) ? creep
   Exit: (19) yall:unify_lambda_parameters_([prince_andrew], [prince_andrew], [], [_116-_212, _116]>>(user:true)) ? cree
p
   Exit: (18) yall:unify_lambda_parameters_([prince_andrew-6, prince_andrew], [prince_andrew-6, prince_andrew], [], [_11
6-_212, _116]>>(user:true)) ? creep
```

```
   Exit: (17) yall:unify_lambda_parameters([_116-_212, _116], [prince_andrew-6, prince_andrew], [], user:true, user:true
) ? creep
   Call: (17) _1832=..[call, user:true] ? creep
   Exit: (17) call(user:true)=..[call, user:true] ? creep
^  Call: (17) call(user:true) ? creep
^  Exit: (17) call(user:true) ? creep
^  Exit: (16) yall: >>([_116-_212, _116], user:true, prince_andrew-6, prince_andrew) ? creep
^  Call: (17) yall: >>([_116-_212, _116], true, prince_edward-2, _5944) ? creep
   Call: (18) yall:unify_lambda_parameters([_116-_212, _116], [prince_edward-2, _5944], _6802, user:true, _6804) ? creep
   Call: (19) var([_116-_212, _116]) ? creep
   Fail: (19) var([_116-_212, _116]) ? creep
   Redo: (18) yall:unify_lambda_parameters([_116-_212, _116], [prince_edward-2, _5944], _6802, user:true, _6804) ? creep
   Call: (19) error:must_be(list, [_116-_212, _116]) ? creep
   Exit: (19) error:must_be(list, [_116-_212, _116]) ? creep
   Call: (19) copy_term_nat([_116-_212, _116]>>(user:true), _11738>>_6804) ? creep
   Exit: (19) copy_term_nat([_116-_212, _116]>>(user:true), [_12568-_12570, _12568]>>(user:true)) ? creep
   Call: (19) yall:unify_lambda_parameters_([_12568-_12570, _12568], [prince_edward-2, _5944], _6802, [_116-_212, _116]>
>(user:true)) ? creep
   Call: (20) _12568-_12570=prince_edward-2 ? creep
   Exit: (20) prince_edward-2=prince_edward-2 ? creep
   Call: (20) yall:unify_lambda_parameters_([prince_edward], [_5944], _6802, [_116-_212, _116]>>(user:true)) ? creep
   Call: (21) prince_edward=_5944 ? creep
   Exit: (21) prince_edward=prince_edward ? creep
   Call: (21) yall:unify_lambda_parameters_([], [], _6802, [_116-_212, _116]>>(user:true)) ? creep
   Exit: (21) yall:unify_lambda_parameters_([], [], [], [_116-_212, _116]>>(user:true)) ? creep
   Exit: (20) yall:unify_lambda_parameters_([prince_edward], [prince_edward], [], [_116-_212, _116]>>(user:true)) ? cree
p
   Exit: (19) yall:unify_lambda_parameters_([prince_edward-2, prince_edward], [prince_edward-2, prince_edward], [], [_11
6-_212, _116]>>(user:true)) ? creep
```

```
   Exit: (18) yall:unify_lambda_parameters([_116-_212, _116], [prince_edward-2, prince_edward], [], user:true, user:true
) ? creep
   Call: (18) _22448=..[call, user:true] ? creep
   Exit: (18) call(user:true)=..[call, user:true] ? creep
^  Call: (18) call(user:true) ? creep
^  Exit: (18) call(user:true) ? creep
^  Exit: (17) yall: >>([_116-_212, _116], user:true, prince_edward-2, prince_edward) ? creep
^  Call: (18) yall: >>([_116-_212, _116], true, princess_ann-0, _26560) ? creep
   Call: (19) yall:unify_lambda_parameters([_116-_212, _116], [princess_ann-0, _26560], _27418, user:true, _27420) ? cre
ep
   Call: (20) var([_116-_212, _116]) ? creep
   Fail: (20) var([_116-_212, _116]) ? creep
   Redo: (19) yall:unify_lambda_parameters([_116-_212, _116], [princess_ann-0, _26560], _27418, user:true, _27420) ? cre
ep
   Call: (20) error:must_be(list, [_116-_212, _116]) ? creep
   Exit: (20) error:must_be(list, [_116-_212, _116]) ? creep
   Call: (20) copy_term_nat([_116-_212, _116]>>(user:true), _1074>>_270) ? creep
   Exit: (20) copy_term_nat([_116-_212, _116]>>(user:true), [_1904-_1906, _1904]>>(user:true)) ? creep
   Call: (20) yall:unify_lambda_parameters_([_1904-_1906, _1904], [princess_ann-0, _246], _268, [_116-_212, _116]>>(user
:true)) ? creep
   Call: (21) _1904-_1906=princess_ann-0 ? creep
   Exit: (21) princess_ann-0=princess_ann-0 ? creep
   Call: (21) yall:unify_lambda_parameters_([princess_ann], [_246], _268, [_116-_212, _116]>>(user:true)) ? creep
   Call: (22) princess_ann=_246 ? creep
   Exit: (22) princess_ann=princess_ann ? creep
```

```
   Call: (22) yall:unify_lambda_parameters_([], [], _268, [_116-_212, _116]>>(user:true)) ? creep
   Exit: (22) yall:unify_lambda_parameters_([], [], [], [_116-_212, _116]>>(user:true)) ? creep
   Exit: (21) yall:unify_lambda_parameters_([princess_ann], [princess_ann], [], [_116-_212, _116]>>(user:true)) ? creep
   Exit: (20) yall:unify_lambda_parameters_([princess_ann-0, princess_ann], [princess_ann-0, princess_ann], [], [_116-_2
12, _116]>>(user:true)) ? creep
   Exit: (19) yall:unify_lambda_parameters([_116-_212, _116], [princess_ann-0, princess_ann], [], user:true, user:true)
? creep
   Call: (19) _11784=..[call, user:true] ? creep
   Exit: (19) call(user:true)=..[call, user:true] ? creep
^  Call: (19) call(user:true) ? creep
^  Exit: (19) call(user:true) ? creep
^  Exit: (18) yall: >>([_116-_212, _116], user:true, princess_ann-0, princess_ann) ? creep
^  Exit: (13) apply:maplist(user:[_116-_212, _116]>>true, [prince_charles-10, prince_andrew-6, prince_edward-2, princess
_ann-0], [prince_charles, prince_andrew, prince_edward, princess_ann]) ? creep
   Exit: (12) line_of_succession(queen_elizabeth, [prince_charles, prince_andrew, prince_edward, princess_ann]) ? creep
X = [prince_charles, prince_andrew, prince_edward, princess_ann].
```

Result : [prince charles, prince andrew, prince edward, princess ann]

b. Line of Succession as per new rule -

Assumptions -

1. The order of birth of the royals is - Prince Charles, Princess Ann, Prince Andrew, Prince Edward.
2. The old Royal succession rule states that the throne is passed down according to the order of birth irrespective of gender.

A. <u>Rules in Knowledge Base -</u>

The new rule of succession has been modified to pass down the throne according to birth order irrespective of gender. The rules defined in prolog according to the old Royal succession rule -

1. If X is older than Y then X is the predecessor of Y.

The prolog code uses these rules to determine the line of succession to the British throne after monarch Queen Elizabeth according to the new succession rule. The prolog code needs to be updated to use the new Predcessor_of rule instead of the old one. The existing successor_of and line_of_succession can remain the same.

<u>Change From Old succession rule -</u>

Predecessor of rule changes. Now, irrespective of gender, the older sibling precedes the younger sibling

1. Translation of **given natural language** statements into **First Order Logic (FOL) Predicates:**

S. No.	Natural Language	First Order Logic
1.	Queen_elizabeth, the monarch of United Kingdom	female(queen_elizabeth)
2.	Has four offsprings; namely:- prince charles, princess ann, prince andrew and prince edward	female(princess_ann). male(prince_charles). male(prince_andrew). male(prince_edward). parent_of(queen_elizabeth, prince_charles). parent_of(queen_elizabeth, prince_andrew). parent_of(queen_elizabeth, prince_edward). parent_of(queen_elizabeth, princess_ann).
3.	Sibling relationship in order of	older_than(prince_charles, princess_ann).

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

2. Rule Predicates

S. No.	Rule	First Order Logic Rule Predicate
1.	Throne is now passed down according to the order of birth irrespective of gender.	$(\forall X)(\forall Y)$predecessor_of(X,Y)$\Leftrightarrow$older_than(X,Y))

3. Functions

S. No.	FOL Function Code Snippet	Explanation

1.	`count_predecessor(X, Count):-`
	`findall(X,`
	`predecessor_of(X, _),`
	`PredecessorList),`
	`length(PredecessorList,`
	`Count).`
		● `findall` finds the list of predecessors of X (`PredecessorList`) using `predecessor_of(X,_)` predicate.
		
		● `length (PredecessorList, Count)` finds the length of PredecessorList for each X and stores it into Count. It finds the number of monarchs who X preceded.
		
		● For example,
		
		`count_predecessor(prince_charles, X)` gives X = 3 as X precedes 3 royals..
		`count_predecessor(princess_ann, X)`. gives X = 2 because ann precedes 2 royals.

2.	`line_of_succession(Monarch, SuccessorList):-`
	
	`3. findall(X-Count,`
	`(member(X,`
		Line_of_succession function is used to find the successor list of monarch.
		
		3. X-Count represents
		X: Royal Name

	[princess_ann, prince_charles, prince_andrew, prince_edward]),count_predecessor(X, Count)), Monarch_Count_Pair), 4. sort(2, @>=,Monarch_Count_Pair, SortedList), maplist([X-_,X]>>true, SortedList, SuccessorList).	Count: Number of royals X precedes Findall function takes the X-Count, from the list of royals and returns the Monarch_Count_Pair as [(prince_charles,3), (princess_ann,2), (prince_andrew,1), (prince_edward, 0)] 4. Sort the Monarch_Count_Pair according to the second element in the pair (i.e. the count part). @>= indicates descending order. Result stored in SortedList. Therefore, line_of_succession(queen_elizabeth,X) gives output - X = [prince_charles, princess_ann, prince_andrew, prince_edward]

4. FOL Statements to Prolog clauses (Code)

```
% Define the royal family members
female(queen_elizabeth).
female(princess_ann).
male(prince_charles).
male(prince_andrew).
male(prince_edward).
male(X):- not(female(X)).

% Define parent child relationships
parent_of(queen_elizabeth, prince_charles).
parent_of(queen_elizabeth, princess_ann).
parent_of(queen_elizabeth, prince_andrew).
parent_of(queen_elizabeth, prince_edward).
child_of(X,Y):- parent_of(Y,X).
```

```
% Define sibling relationships
older_than(prince_charles, princess_ann).
older_than(prince_charles, prince_andrew).
older_than(prince_charles, prince_edward).
older_than(princess_ann, prince_andrew).
older_than(princess_ann, prince_edward).
older_than(prince_andrew, prince_edward).

% X is successor of Y when X is child of Y.
successor_of(X,Y) :- child_of(X,Y).
```

```
% Order of precedence according to new rule.
% Define rules
predecessor_of(X,Y) :-
    % If X and Y are siblings and X is older than Y then X precedes Y.
    successor_of(X,W), successor_of(Y,W), older_than(X,Y).

% Count nuumber of Predecessors of X
count_predecessor(X, Count):- findall(X, predecessor_of(X, _), PredecessorList), length

% Find the line of Succession of the Monarch
line_of_succession(Monarch, SuccessorList):-
    findall(X-Count, (member(X, [princess_ann, prince_charles, prince_andrew, prince_ed
    sort(2, @>=,Monarch_Count_Pair, SortedList),
    maplist([X-_,X]>>true, SortedList, SuccessorList).
```

5. **Simple Output**

```
[debug] 6 ?- line_of_succession(queen_elizabeth,X).
X = [prince_charles, princess_ann, prince_andrew, prince_edward].
```

6. **Trace**

   A trace of the Prolog program will show the order of execution and how the succession line is computed based on the old Royal succession rule.

```
   Call: (12) line_of_succession(queen_elizabeth, _276) ? creep
^  Call: (13) findall(_1582-_1584, (member(_1582, [princess_ann, prince_charles, prince_andrew, prince_edward]), count_p
redecessor(_1582, _1584)), _1628) ? creep
   Call: (18) lists:member(_1582, [princess_ann, prince_charles, prince_andrew, prince_edward]) ? creep
   Exit: (18) lists:member(princess_ann, [princess_ann, prince_charles, prince_andrew, prince_edward]) ? creep
   Call: (18) count_predecessor(princess_ann, _1584) ? creep
^  Call: (19) findall(princess_ann, predecessor_of(princess_ann, _4942), _4944) ? creep
   Call: (23) predecessor_of(princess_ann, _4942) ? creep
   Call: (24) successor_of(princess_ann, _6610) ? creep
   Call: (25) child_of(princess_ann, _6610) ? creep
   Call: (26) parent_of(_6610, princess_ann) ? creep
   Exit: (26) parent_of(queen_elizabeth, princess_ann) ? creep
   Exit: (25) child_of(princess_ann, queen_elizabeth) ? creep
   Exit: (24) successor_of(princess_ann, queen_elizabeth) ? creep
   Call: (24) successor_of(_4942, queen_elizabeth) ? creep
   Call: (25) child_of(_4942, queen_elizabeth) ? creep
   Call: (26) parent_of(queen_elizabeth, _4942) ? creep
   Exit: (26) parent_of(queen_elizabeth, prince_charles) ? creep
   Exit: (25) child_of(prince_charles, queen_elizabeth) ? creep
   Exit: (24) successor_of(prince_charles, queen_elizabeth) ? creep
   Call: (24) older_than(princess_ann, prince_charles) ? creep
   Fail: (24) older_than(princess_ann, prince_charles) ? creep
```

```
   Redo: (26) parent_of(queen_elizabeth, _4942) ? creep
   Exit: (26) parent_of(queen_elizabeth, princess_ann) ? creep
   Exit: (25) child_of(princess_ann, queen_elizabeth) ? creep
   Exit: (24) successor_of(princess_ann, queen_elizabeth) ? creep
   Call: (24) older_than(princess_ann, princess_ann) ? creep
   Fail: (24) older_than(princess_ann, princess_ann) ? creep
   Redo: (26) parent_of(queen_elizabeth, _4942) ? creep
   Exit: (26) parent_of(queen_elizabeth, prince_andrew) ? creep
   Exit: (25) child_of(prince_andrew, queen_elizabeth) ? creep
   Exit: (24) successor_of(prince_andrew, queen_elizabeth) ? creep
   Call: (24) older_than(princess_ann, prince_andrew) ? creep
   Exit: (24) older_than(princess_ann, prince_andrew) ? creep
   Exit: (23) predecessor_of(princess_ann, prince_andrew) ? creep
   Redo: (24) older_than(princess_ann, prince_andrew) ? creep
   Fail: (24) older_than(princess_ann, prince_andrew) ? creep
   Redo: (26) parent_of(queen_elizabeth, _4942) ? creep
   Exit: (26) parent_of(queen_elizabeth, prince_edward) ? creep
   Exit: (25) child_of(prince_edward, queen_elizabeth) ? creep
   Exit: (24) successor_of(prince_edward, queen_elizabeth) ? creep
   Call: (24) older_than(princess_ann, prince_edward) ? creep
   Exit: (24) older_than(princess_ann, prince_edward) ? creep
   Exit: (23) predecessor_of(princess_ann, prince_edward) ? creep
```

```
^  Exit: (19) findall(princess_ann, user:predecessor_of(princess_ann, _198), [princess_ann, princess_ann]) ? creep
   Call: (19) length([princess_ann, princess_ann], _120) ? creep
   Exit: (19) length([princess_ann, princess_ann], 2) ? creep
   Exit: (18) count_predecessor(princess_ann, 2) ? creep
   Redo: (18) lists:member(_118, [princess_ann, prince_charles, prince_andrew, prince_edward]) ? creep
   Exit: (18) lists:member(prince_charles, [princess_ann, prince_charles, prince_andrew, prince_edward]) ? creep
   Call: (18) count_predecessor(prince_charles, _120) ? creep
^  Call: (19) findall(prince_charles, predecessor_of(prince_charles, _10014), _10016) ? creep
   Call: (23) predecessor_of(prince_charles, _10014) ? creep
   Call: (24) successor_of(prince_charles, _11682) ? creep
   Call: (25) child_of(prince_charles, _11682) ? creep
   Call: (26) parent_of(_11682, prince_charles) ? creep
   Exit: (26) parent_of(queen_elizabeth, prince_charles) ? creep
   Exit: (25) child_of(prince_charles, queen_elizabeth) ? creep
   Exit: (24) successor_of(prince_charles, queen_elizabeth) ? creep
   Call: (24) successor_of(_10014, queen_elizabeth) ? creep
   Call: (25) child_of(_10014, queen_elizabeth) ? creep
   Call: (26) parent_of(queen_elizabeth, _10014) ? creep
   Exit: (26) parent_of(queen_elizabeth, prince_charles) ? creep
   Exit: (25) child_of(prince_charles, queen_elizabeth) ? creep
   Exit: (24) successor_of(prince_charles, queen_elizabeth) ? creep
   Call: (24) older_than(prince_charles, prince_charles) ? creep
   Fail: (24) older_than(prince_charles, prince_charles) ? creep
```

```
   Redo: (26) parent_of(queen_elizabeth, _10014) ? creep
   Exit: (26) parent_of(queen_elizabeth, princess_ann) ? creep
   Exit: (25) child_of(princess_ann, queen_elizabeth) ? creep
   Exit: (24) successor_of(princess_ann, queen_elizabeth) ? creep
   Call: (24) older_than(prince_charles, princess_ann) ? creep
   Exit: (24) older_than(prince_charles, princess_ann) ? creep
   Exit: (23) predecessor_of(prince_charles, princess_ann) ? creep
   Redo: (24) older_than(prince_charles, princess_ann) ? creep
   Fail: (24) older_than(prince_charles, princess_ann) ? creep
   Redo: (26) parent_of(queen_elizabeth, _10014) ? creep
   Exit: (26) parent_of(queen_elizabeth, prince_andrew) ? creep
   Exit: (25) child_of(prince_andrew, queen_elizabeth) ? creep
   Exit: (24) successor_of(prince_andrew, queen_elizabeth) ? creep
   Call: (24) older_than(prince_charles, prince_andrew) ? creep
   Exit: (24) older_than(prince_charles, prince_andrew) ? creep
   Exit: (23) predecessor_of(prince_charles, prince_andrew) ? creep
   Redo: (24) older_than(prince_charles, prince_andrew) ? creep
   Fail: (24) older_than(prince_charles, prince_andrew) ? creep
   Redo: (26) parent_of(queen_elizabeth, _198) ? creep
   Exit: (26) parent_of(queen_elizabeth, prince_edward) ? creep
   Exit: (25) child_of(prince_edward, queen_elizabeth) ? creep
   Exit: (24) successor_of(prince_edward, queen_elizabeth) ? creep
   Call: (24) older_than(prince_charles, prince_edward) ? creep
   Exit: (24) older_than(prince_charles, prince_edward) ? creep
   Exit: (23) predecessor_of(prince_charles, prince_edward) ? creep
```

```
^  Exit: (19) findall(prince_charles, user:predecessor_of(prince_charles, _198), [prince_charles, prince_charles, prince
_charles]) ? creep
   Call: (19) length([prince_charles, prince_charles, prince_charles], _120) ? creep
   Exit: (19) length([prince_charles, prince_charles, prince_charles], 3) ? creep
   Exit: (18) count_predecessor(prince_charles, 3) ? creep
   Redo: (18) lists:member(_118, [princess_ann, prince_charles, prince_andrew, prince_edward]) ? creep
   Exit: (18) lists:member(prince_andrew, [princess_ann, prince_charles, prince_andrew, prince_edward]) ? creep
   Call: (18) count_predecessor(prince_andrew, _120) ? creep
^  Call: (19) findall(prince_andrew, predecessor_of(prince_andrew, _17846), _17848) ? creep
   Call: (23) predecessor_of(prince_andrew, _17846) ? creep
   Call: (24) successor_of(prince_andrew, _19514) ? creep
   Call: (25) child_of(prince_andrew, _19514) ? creep
   Call: (26) parent_of(_19514, prince_andrew) ? creep
   Exit: (26) parent_of(queen_elizabeth, prince_andrew) ? creep
   Exit: (25) child_of(prince_andrew, queen_elizabeth) ? creep
   Exit: (24) successor_of(prince_andrew, queen_elizabeth) ? creep
   Call: (24) successor_of(_17846, queen_elizabeth) ? creep
   Call: (25) child_of(_17846, queen_elizabeth) ? creep
   Call: (26) parent_of(queen_elizabeth, _17846) ? creep
   Exit: (26) parent_of(queen_elizabeth, prince_charles) ? creep
   Exit: (25) child_of(prince_charles, queen_elizabeth) ? creep
   Exit: (24) successor_of(prince_charles, queen_elizabeth) ? creep
   Call: (24) older_than(prince_andrew, prince_charles) ? creep
   Fail: (24) older_than(prince_andrew, prince_charles) ? creep
```

```
   Redo: (26) parent_of(queen_elizabeth, _17846) ? creep
   Exit: (26) parent_of(queen_elizabeth, princess_ann) ? creep
   Exit: (25) child_of(princess_ann, queen_elizabeth) ? creep
   Exit: (24) successor_of(princess_ann, queen_elizabeth) ? creep
   Call: (24) older_than(prince_andrew, princess_ann) ? creep
   Fail: (24) older_than(prince_andrew, princess_ann) ? creep
   Redo: (26) parent_of(queen_elizabeth, _198) ? creep
   Exit: (26) parent_of(queen_elizabeth, prince_andrew) ? creep
   Exit: (25) child_of(prince_andrew, queen_elizabeth) ? creep
   Exit: (24) successor_of(prince_andrew, queen_elizabeth) ? creep
   Call: (24) older_than(prince_andrew, prince_andrew) ? creep
   Fail: (24) older_than(prince_andrew, prince_andrew) ? creep
   Redo: (26) parent_of(queen_elizabeth, _198) ? creep
   Exit: (26) parent_of(queen_elizabeth, prince_edward) ? creep
   Exit: (25) child_of(prince_edward, queen_elizabeth) ? creep
   Exit: (24) successor_of(prince_edward, queen_elizabeth) ? creep
   Call: (24) older_than(prince_andrew, prince_edward) ? creep
   Exit: (24) older_than(prince_andrew, prince_edward) ? creep
   Exit: (23) predecessor_of(prince_andrew, prince_edward) ? creep
^  Exit: (19) findall(prince_andrew, user:predecessor_of(prince_andrew, _198), [prince_andrew]) ? creep
   Call: (19) length([prince_andrew], _120) ? creep
   Exit: (19) length([prince_andrew], 1) ? creep
   Exit: (18) count_predecessor(prince_andrew, 1) ? creep
```

```
   Redo: (18) lists:member(_118, [princess_ann, prince_charles, prince_andrew, prince_edward]) ? creep
   Exit: (18) lists:member(prince_edward, [princess_ann, prince_charles, prince_andrew, prince_edward]) ? creep
   Call: (18) count_predecessor(prince_edward, _120) ? creep
^  Call: (19) findall(prince_edward, predecessor_of(prince_edward, _20538), _20540) ? creep
   Call: (23) predecessor_of(prince_edward, _20538) ? creep
   Call: (24) successor_of(prince_edward, _22206) ? creep
   Call: (25) child_of(prince_edward, _22206) ? creep
   Call: (26) parent_of(_22206, prince_edward) ? creep
   Exit: (26) parent_of(queen_elizabeth, prince_edward) ? creep
   Exit: (25) child_of(prince_edward, queen_elizabeth) ? creep
   Exit: (24) successor_of(prince_edward, queen_elizabeth) ? creep
   Call: (24) successor_of(_20538, queen_elizabeth) ? creep
   Call: (25) child_of(_20538, queen_elizabeth) ? creep
   Call: (26) parent_of(queen_elizabeth, _20538) ? creep
   Exit: (26) parent_of(queen_elizabeth, prince_charles) ? creep
   Exit: (25) child_of(prince_charles, queen_elizabeth) ? creep
   Exit: (24) successor_of(prince_charles, queen_elizabeth) ? creep
   Call: (24) older_than(prince_edward, prince_charles) ? creep
   Fail: (24) older_than(prince_edward, prince_charles) ? creep
   Redo: (26) parent_of(queen_elizabeth, _198) ? creep
   Exit: (26) parent_of(queen_elizabeth, princess_ann) ? creep
   Exit: (25) child_of(princess_ann, queen_elizabeth) ? creep
   Exit: (24) successor_of(princess_ann, queen_elizabeth) ? creep
   Call: (24) older_than(prince_edward, princess_ann) ? creep
   Fail: (24) older_than(prince_edward, princess_ann) ? creep
```

```
   Redo: (26) parent_of(queen_elizabeth, _198) ? creep
   Exit: (26) parent_of(queen_elizabeth, prince_andrew) ? creep
   Exit: (25) child_of(prince_andrew, queen_elizabeth) ? creep
   Exit: (24) successor_of(prince_andrew, queen_elizabeth) ? creep
   Call: (24) older_than(prince_edward, prince_andrew) ? creep
   Fail: (24) older_than(prince_edward, prince_andrew) ? creep
   Redo: (26) parent_of(queen_elizabeth, _198) ? creep
   Exit: (26) parent_of(queen_elizabeth, prince_edward) ? creep
   Exit: (25) child_of(prince_edward, queen_elizabeth) ? creep
   Exit: (24) successor_of(prince_edward, queen_elizabeth) ? creep
   Call: (24) older_than(prince_edward, prince_edward) ? creep
   Fail: (24) older_than(prince_edward, prince_edward) ? creep
   Fail: (23) predecessor_of(prince_edward, _198) ? creep
^  Exit: (19) findall(prince_edward, user:predecessor_of(prince_edward, _198), []) ? creep
   Call: (19) length([], _120) ? creep
   Exit: (19) length([], 0) ? creep
   Exit: (18) count_predecessor(prince_edward, 0) ? creep
^  Exit: (13) findall(_118-_120, user:(member(_118, [princess_ann, prince_charles, prince_andrew, prince_edward]), count
_predecessor(_118, _120)), [princess_ann-2, prince_charles-3, prince_andrew-1, prince_edward-0]) ? creep
   Call: (13) sort(2, @>=, [princess_ann-2, prince_charles-3, prince_andrew-1, prince_edward-0], _21914) ? creep
   Exit: (13) sort(2, @>=, [princess_ann-2, prince_charles-3, prince_andrew-1, prince_edward-0], [prince_charles-3, prin
cess_ann-2, prince_andrew-1, prince_edward-0]) ? creep
^  Call: (13) apply:maplist('__aux_yall_78cbea4df0621fc54d3badcf2a338283de5ff8c7', [prince_charles-3, princess_ann-2, pr
ince_andrew-1, prince_edward-0], _58) ? creep
```

```
   Call: (15) '__aux_yall_78cbea4df0621fc54d3badcf2a338283de5ff8c7'(prince_charles-3, _24416) ? creep
   Exit: (15) '__aux_yall_78cbea4df0621fc54d3badcf2a338283de5ff8c7'(prince_charles-3, prince_charles) ? creep
   Call: (16) '__aux_yall_78cbea4df0621fc54d3badcf2a338283de5ff8c7'(princess_ann-2, _26042) ? creep
   Exit: (16) '__aux_yall_78cbea4df0621fc54d3badcf2a338283de5ff8c7'(princess_ann-2, princess_ann) ? creep
   Call: (17) '__aux_yall_78cbea4df0621fc54d3badcf2a338283de5ff8c7'(prince_andrew-1, _27668) ? creep
   Exit: (17) '__aux_yall_78cbea4df0621fc54d3badcf2a338283de5ff8c7'(prince_andrew-1, prince_andrew) ? creep
   Call: (18) '__aux_yall_78cbea4df0621fc54d3badcf2a338283de5ff8c7'(prince_edward-0, _29294) ? creep
   Exit: (18) '__aux_yall_78cbea4df0621fc54d3badcf2a338283de5ff8c7'(prince_edward-0, prince_edward) ? creep
^  Exit: (13) apply:maplist(user:'__aux_yall_78cbea4df0621fc54d3badcf2a338283de5ff8c7', [prince_charles-3, princess_ann-
2, prince_andrew-1, prince_edward-0], [prince_charles, princess_ann, prince_andrew, prince_edward]) ? creep
   Exit: (12) line_of_succession(queen_elizabeth, [prince_charles, princess_ann, prince_andrew, prince_edward]) ? creep
X = [prince_charles, princess_ann, prince_andrew, prince_edward].
```

Result : [prince charles, princess ann, prince andrew, prince edward]