ELFE
Résolution – Récursivité

Compte rendu TP

Nom & prénom:
Djebien Tarik
Groupe : **4**
Date: **Février 2011**

1 Généalogie

Question 1

1.1

généalogie1.pl

```
/* Récursivité                  */
/* genealogie1.pl               */
/* "Prolog, tout de suite !" p 66    */

enfant_de(anne,bridget).
enfant_de(bridget,caroline).
enfant_de(caroline,donna).
enfant_de(donna,emily).

descendant_de(X,Y) :- enfant_de(X,Y).
descendant_de(X,Y) :-
        enfant_de(X,Z),
        descendant_de(Z,Y).
```

1.2

```
[trace]  ?- descendant_de(anne,emily).
   Call: (7) descendant_de(anne, emily) ? creep
   Call: (8) enfant_de(anne, emily) ? creep
   Fail: (8) enfant_de(anne, emily) ? creep
   Redo: (7) descendant_de(anne, emily) ? creep
   Call: (8) enfant_de(anne, _L196) ? creep
   Exit: (8) enfant_de(anne, bridget) ? creep
   Call: (8) descendant_de(bridget, emily) ? creep
   Call: (9) enfant_de(bridget, emily) ? creep
   Fail: (9) enfant_de(bridget, emily) ? creep
   Redo: (8) descendant_de(bridget, emily) ? creep
   Call: (9) enfant_de(bridget, _L207) ? creep
   Exit: (9) enfant_de(bridget, caroline) ? creep
   Call: (9) descendant_de(caroline, emily) ? creep
   Call: (10) enfant_de(caroline, emily) ? creep
   Fail: (10) enfant_de(caroline, emily) ? creep
   Redo: (9) descendant_de(caroline, emily) ? creep
   Call: (10) enfant_de(caroline, _L218) ? creep
   Exit: (10) enfant_de(caroline, donna) ? creep
   Call: (10) descendant_de(donna, emily) ? creep
   Call: (11) enfant_de(donna, emily) ? creep
   Exit: (11) enfant_de(donna, emily) ? creep
   Exit: (10) descendant_de(donna, emily) ? creep
   Exit: (9) descendant_de(caroline, emily) ? creep
   Exit: (8) descendant_de(bridget, emily) ? creep
   Exit: (7) descendant_de(anne, emily) ? creep
true.
```

Il y a donc 25 étapes pour que PROLOG parvient à obtenir une réponse à la requête suivante:
?- descendant_de(anne,emily).

1.3

La requete nous donne :
[debug] ?- descendant_de(X,Y).
X = anne,
Y = bridget ;
X = bridget,
Y = caroline ;
X = caroline,
Y = donna ;
X = donna,
Y = emily ;
X = anne,
Y = caroline ;
X = anne,
Y = donna ;
X = anne,
Y = emily ;
X = bridget,
Y = donna ;
X = bridget,
Y = emily ;
X = caroline,
Y = emily ;
fail.

On obtient donc 10 solutions possibles.

Question 2.

2.1

généalogie2.pl

```
/* Récursivité                  */
/* genealogie2.pl               */
/* "Prolog, tout de suite !" p 66    */

enfant_de(anne,bridget).
enfant_de(bridget,caroline).
enfant_de(caroline,donna).
enfant_de(donna,emily).


descendant_de(X,Y) :-
        enfant_de(X,Z),
        descendant_de(Z,Y).
descendant_de(X,Y) :- enfant_de(X,Y).
```

2.2

```
[trace]  ?- descendant_de(anne,emily).
   Call: (7) descendant_de(anne, emily) ? creep
   Call: (8) enfant_de(anne, _L196) ? creep
   Exit: (8) enfant_de(anne, bridget) ? creep
   Call: (8) descendant_de(bridget, emily) ? creep
   Call: (9) enfant_de(bridget, _L214) ? creep
   Exit: (9) enfant_de(bridget, caroline) ? creep
   Call: (9) descendant_de(caroline, emily) ? creep
   Call: (10) enfant_de(caroline, _L232) ? creep
   Exit: (10) enfant_de(caroline, donna) ? creep
   Call: (10) descendant_de(donna, emily) ? creep
   Call: (11) enfant_de(donna, _L250) ? creep
   Exit: (11) enfant_de(donna, emily) ? creep
   Call: (11) descendant_de(emily, emily) ? creep
   Call: (12) enfant_de(emily, _L268) ? creep
   Fail: (12) enfant_de(emily, _L268) ? creep
   Redo: (11) descendant_de(emily, emily) ? creep
   Call: (12) enfant_de(emily, emily) ? creep
   Fail: (12) enfant_de(emily, emily) ? creep
   Redo: (10) descendant_de(donna, emily) ? creep
   Call: (11) enfant_de(donna, emily) ? creep
   Exit: (11) enfant_de(donna, emily) ? creep
   Exit: (10) descendant_de(donna, emily) ? creep
   Exit: (9) descendant_de(caroline, emily) ? creep
   Exit: (8) descendant_de(bridget, emily) ? creep
   Exit: (7) descendant_de(anne, emily) ? creep
true.
```

Il y a donc 25 étapes pour que PROLOG parvient à obtenir une réponse à la requête suivante:

```
  ?- descendant_de(anne,emily).
```

2.3

```
?- descendant_de(X,Y).
X = anne,
Y = emily ;
X = anne,
Y = donna ;
X = anne,
Y = caroline ;
X = bridget,
Y = emily ;
X = bridget,
Y = donna ;
X = caroline,
Y = emily ;
X = anne,
Y = bridget ;
X = bridget,
Y = caroline ;
X = caroline,
Y = donna ;
X = donna,
Y = emily ;
fail.
```

On obtient donc également 10 solutions possibles.

Question 3.

3.1

généalogie3.pl

```
/* Récursivité                  */
/* genealogie3.pl               */
/* "Prolog, tout de suite !" p 67    */

enfant_de(anne,bridget).
enfant_de(bridget,caroline).
enfant_de(caroline,donna).
enfant_de(donna,emily).


descendant_de(X,Y) :-
        descendant_de(Z,Y),
        enfant_de(X,Z).
descendant_de(X,Y) :- enfant_de(X,Y).
```

3.2

Call: (8) descendant_de(anne, emily) ? creep
Call: (9) descendant_de(_L196, emily) ? creep
Call: (10) descendant_de(_L214, emily) ? creep
Call: (11) descendant_de(_L232, emily) ? creep
Call: (12) descendant_de(_L250, emily) ? creep
Call: (13) descendant_de(_L268, emily) ? creep
Call: (14) descendant_de(_L286, emily) ? creep
Call: (15) descendant_de(_L304, emily) ? creep
Call: (16) descendant_de(_L322, emily) ? creep
Call: (17) descendant_de(_L340, emily) ? creep
Call: (18) descendant_de(_L358, emily) ? creep
Call: (19) descendant_de(_L376, emily) ? creep
Call: (20) descendant_de(_L394, emily) ? Creep

1. .....

   Il y a donc une infinité d'étapes et PROLOG ne parvient pas à obtenir une réponse à la requête suivante:

   ?- descendant_de(anne,emily).

2.

3.3

?- descendant_de(X,Y).
ERROR: Out of local stack
Exception: (232,333) descendant_de(_L4181990, _G184) ? creep
Exception: (232,332) descendant_de(_L4181972, _G184) ? creep
Exception: (232,331) descendant_de(_L4181954, _G184) ? creep
Exception: (232,330) descendant_de(_L4181936, _G184) ? Creep

On obtient donc aucune solutions possibles.

Question 4.

4.1

généalogie4.pl

```
/* Récursivité                  */
/* genealogie4.pl               */
/* "Prolog, tout de suite !" p 68    */

enfant_de(anne,bridget).
enfant_de(bridget,caroline).
enfant_de(caroline,donna).
enfant_de(donna,emily).

descendant_de(X,Y) :- enfant_de(X,Y).
descendant_de(X,Y) :-
        descendant_de(Z,Y),
        enfant_de(X,Z).
```

4.2

```
[trace]  ?- descendant_de(anne,emily).
   Call: (7) descendant_de(anne, emily) ? creep
   Call: (8) enfant_de(anne, emily) ? creep
   Fail: (8) enfant_de(anne, emily) ? creep
   Redo: (7) descendant_de(anne, emily) ? creep
   Call: (8) descendant_de(_L196, emily) ? creep
   Call: (9) enfant_de(_L196, emily) ? creep
   Exit: (9) enfant_de(donna, emily) ? creep
   Exit: (8) descendant_de(donna, emily) ? creep
   Call: (8) enfant_de(anne, donna) ? creep
   Fail: (8) enfant_de(anne, donna) ? creep
   Redo: (8) descendant_de(_L196, emily) ? creep
   Call: (9) descendant_de(_L207, emily) ? creep
   Call: (10) enfant_de(_L207, emily) ? creep
   Exit: (10) enfant_de(donna, emily) ? creep
   Exit: (9) descendant_de(donna, emily) ? creep
   Call: (9) enfant_de(_L196, donna) ? creep
   Exit: (9) enfant_de(caroline, donna) ? creep
   Exit: (8) descendant_de(caroline, emily) ? creep
   Call: (8) enfant_de(anne, caroline) ? creep
   Fail: (8) enfant_de(anne, caroline) ? creep
   Redo: (9) enfant_de(_L196, donna) ? creep
   Redo: (9) descendant_de(_L207, emily) ? creep
   Call: (10) descendant_de(_L218, emily) ? creep
   Call: (11) enfant_de(_L218, emily) ? creep
   Exit: (11) enfant_de(donna, emily) ? creep
   Exit: (10) descendant_de(donna, emily) ? creep
   Call: (10) enfant_de(_L207, donna) ? creep
   Exit: (10) enfant_de(caroline, donna) ? creep
   Exit: (9) descendant_de(caroline, emily) ? creep
   Call: (9) enfant_de(_L196, caroline) ? creep
   Exit: (9) enfant_de(bridget, caroline) ? creep
   Exit: (8) descendant_de(bridget, emily) ? creep
   Call: (8) enfant_de(anne, bridget) ? creep
   Exit: (8) enfant_de(anne, bridget) ? creep
   Exit: (7) descendant_de(anne, emily) ? creep
true.
```

Il y a donc 35 étapes pour que PROLOG parvient à obtenir une réponse à la requête suivante:

```
  ?- descendant_de(anne,emily).
```

4.3.

?- descendant_de(X,Y).
X = anne,
Y = bridget ;
X = bridget,
Y = caroline ;
X = caroline,
Y = donna ;
X = donna,
Y = emily ;
X = anne,
Y = caroline ;
X = bridget,
Y = donna ;
X = caroline,
Y = emily ;
X = anne,
Y = donna ;
X = bridget,
Y = emily ;
X = anne,
Y = emily ;
ERROR: Out of local stack

On obtient donc 10 solutions possibles ensuite PROLOG boucle indéfiniment sur la branche de droite de l'arbre de résolution et déborde la pile d'exécution à cause d'un nombre de constante trop élevé.

Question 5.

	Généalogie1.pl	**Généalogie2.pl**	**Généalogie3.pl**	**Généalogie4.pl**
Nombres d'étapes.	25	25	∞ (infinité)	35
Nombres solutions	10	10	Aucune	10

On remarque donc que la meilleure façon d'exprimer le prédicat descendant_de/2 reste celle utilisée dans généalogie1.pl ou généalogie2.pl

Critères de sélection : Rapidité de résolution et d'unification de la requêtes.

Conclusion :
Parce que les significations déclaratives et procédurales d'un programme de PROLOG peuvent différer, en écrivant des programmes de PROLOG on doit tenir compte des deux aspects. Souvent nous pouvons obtenir l'idée complète de la façon d'écrire le programme en pensant déclarativement, c'est-à-dire en pensant simplement en termes de décrire le problème précisément.