



Instituto Tecnológico de Mexicali

Practica 3

Carrera:

Ingeniería en Sistemas Computacionales

Nombre del Alumno(a):

Martínez Yebra Beatriz Andrea
#13490929

Nombre del profesor(a):

Tafoya Diaz Luis Aram

Materia:

Programación Lógica y Funcional

Horario:

5:00 p.m. – 6:00 p.m.

Mexicali Baja California, Miércoles 27 de Septiembre 2017.

PART 1

```
mama( ana , hector ) .  
mama( ana , maria ) .  
mama( janet , marcos ) .  
mama( janet , francisco ) .  
mama( anabel , julian ) .  
mama( maria , benito ) .  
mama( jazmin , lorena ) .  
mama( lucia , rosa ) .  
mama( lorena , carolina ) .  
mama( carolina , david ) .  
mama( carolina , rosa ) .  
papa( hector , julian ) .  
papa( juan , maria ) .  
papa( francisco , teodulfo ) .  
papa( francisco , lorena ) .  
papa( juan , marcos ) .  
papa( gabriel , lucia ) .  
papa( juan , francisco ) .
```

```
progenitor( victoria , george ) .  
progenitor( victoria , eduardo ) .  
progenitor(X , Y ):- mama(X , Y ) .  
progenitor(X , Y ):- papa(X , Y ) .  
progenitor( elizabeth , carlos ) .  
progenitor( elizabeth , andres ) .
```

```
ancestro(X , Y ) :- progenitor(X , Y ) .  
ancestro(X , Y ) :- progenitor(X , Z ) , ancestro(Z , Y ) .
```

```
hijo_de(A , B):- progenitor(B , A).  
abuelo_de(A , B):- progenitor(A , C) , progenitor(C,B).
```

Extender el programa para definir lo siguiente:

1.-

hijo_de(A,B)

2.-

abuelo_de(A,B)

3.-

Abuela_de(A,B)

4.-

bisabuelo_de(A,B)

RESPUESTAS

1.-

hijo_de(A,B).

```
3 mama( janet , marcos ) .
4 mama( janet , francisco ) .
5 mama( anabel , julian ) .
6 mama( maria , benito ) .
7 mama( jazmin , lorena ) .
8 mama( lucia , rosa ) .
9 mama( lorena , carolina ) .
10 mama( carolina , david ) .
11 mama( carolina , rosa ) .
12 papa( hector , julian ) .
13 papa( juan , maria ) .
14 papa( francisco , teodulfo ) .
15 papa( francisco , lorena ) .
16 papa( juan , marcos ) .
17 papa( gabriel , lucia ) .
18 papa( juan , francisco ) .
19
20 progenitor( victoria , george ) .
21 progenitor( victoria , eduardo ) .
22 progenitor(X , Y ):- mama(X , Y ) .
23 progenitor(X , Y ):- papa(X , Y ) .
24 progenitor( elizabeth , carlos ) .
25 progenitor( elizabeth , andres ) .
26
27 ancestro(X , Y ):- progenitor(X , Y ) .
28 ancestro(X , Y ):- progenitor(X , Z ) , ancestro(Z , Y ) .
29 hijo_de(A , B):- progenitor(B , A) .
30 abuelo_de(A , B):- progenitor(A , C) , progenitor(C , B) .
```

Execution results:

```
false
abuelo_de(elizabeth, carlos).
false
abuelo_de(X,Y).
X = ana,
Y = julian
X = ana,
Y = benito
X = janet,
Y = teodulfo
X = janet,
Y = lorena
hijo_de(julian, anabel).
true
Next 10 100 1,000 Stop
?- hijo_de(julian, anabel).
```

2.-

abuelo_de(A,B).

```
6 mama( maria , benito ) .
7 mama( jazmin , lorena ) .
8 mama( lucia , rosa ) .
9 mama( lorena , carolina ) .
10 mama( carolina , david ) .
11 mama( carolina , rosa ) .
12 papa( hector , julian ) .
13 papa( juan , maria ) .
14 papa( francisco , teodulfo ) .
15 papa( francisco , lorena ) .
16 papa( juan , marcos ) .
17 papa( gabriel , lucia ) .
18 papa( juan , francisco ) .
19
20 progenitor( victoria , george ) .
21 progenitor( victoria , eduardo ) .
22 progenitor(X , Y ):- mama(X , Y ) .
23 progenitor(X , Y ):- papa(X , Y ) .
24 progenitor( elizabeth , carlos ) .
25 progenitor( elizabeth , andres ) .
26
27 ancestro(X , Y ):- progenitor(X , Y ) .
28 ancestro(X , Y ):- progenitor(X , Z ) , ancestro(Z , Y ) .
29 hijo_de(A , B):- progenitor(B , A) .
30 abuelo_de(A , B):- progenitor(A , C) , progenitor(C , B) , papa(X , _).
31 abuela_de(A , B):- progenitor(A , C) , progenitor(C , B) , mama(X , _).
```

Execution results:

```
A = ana,
B = benito
A = janet,
B = teodulfo
A = janet,
B = lorena
A = jazmin,
B = carolina
A = lorena,
B = david
A = lorena,
B = rosa
A = juan,
B = benito
A = francisco,
B = carolina
A = gabriel,
B = rosa
A = juan,
B = teodulfo
Next 10 100 1,000 Stop
?- abuelo_de(A,B).
```

3.-

Abuela_de(A,B).

The screenshot shows a Prolog IDE with a program defining family relationships. The program includes predicates for `mama`, `papa`, `progenitor`, `ancestro`, `hijo_de`, `abuelo_de`, `abuela_de`, and `abuelo_de`. The execution results on the right show a series of bindings for `A` and `B` that satisfy the `abuela_de(A,B)` query.

```

6 mama( maria , benito ) .
7 mama( jazmin , lorena ) .
8 mama( lucia , rosa ) .
9 mama( lorena , carolina ) .
10 mama( carolina , david ) .
11 mama( carolina , rosa ) .
12 papa( hector , julian ) .
13 papa( juan , maria ) .
14 papa( francisco , teodulfo ) .
15 papa( francisco , lorena ) .
16 papa( juan , marcos ) .
17 papa( gabriel , lucia ) .
18 papa( juan , francisco ) .
19
20 progenitor( victoria , george ) .
21 progenitor( victoria , eduardo ) .
22 progenitor(X , Y):- mama(X , Y ) .
23 progenitor(X , Y):- papa(X , Y ) .
24 progenitor( elizabeth , carlos ) .
25 progenitor( elizabeth , andres ) .
26
27 ancestro(X , Y ) :- progenitor(X , Y ) .
28 ancestro(X , Y ) :- progenitor(X , Z ) , ancestro(Z , Y ) .
29
30 hijo_de(A , B):- progenitor(B , A).
31 abuelo_de(A , B):- progenitor(A , C), progenitor(C,B), papa(A,_).
32 abuela_de(A , B):- progenitor(A,C), progenitor(C,B), mama(A,_).
33

```

Execution results for `abuela_de(A,B)`:

```

A = ana,
B = julian
A = ana,
B = julian
A = ana,
B = benito
A = ana,
B = benito
A = janet,
B = teodulfo
A = janet,
B = teodulfo
A = janet,
B = lorena
A = janet,
B = lorena

```

4.-

bisabuelo_de(A,B).

The screenshot shows the same Prolog IDE with the program from the previous example. The execution results on the right show the results of the `bisabuelo_de(M,N)` query. The results are mostly `false`, but one result is highlighted with a red box, showing bindings for `A` and `C`.

```

7 mama( jazmin , lorena ) .
8 mama( lucia , rosa ) .
9 mama( lorena , carolina ) .
10 mama( carolina , david ) .
11 mama( carolina , rosa ) .
12 papa( hector , julian ) .
13 papa( juan , maria ) .
14 papa( francisco , teodulfo ) .
15 papa( francisco , lorena ) .
16 papa( juan , marcos ) .
17 papa( gabriel , lucia ) .
18 papa( juan , francisco ) .
19
20 progenitor( victoria , george ) .
21 progenitor( victoria , eduardo ) .
22 progenitor(X , Y):- mama(X , Y ) .
23 progenitor(X , Y):- papa(X , Y ) .
24 progenitor( elizabeth , carlos ) .
25 progenitor( elizabeth , andres ) .
26
27 ancestro(X , Y ) :- progenitor(X , Y ) .
28 ancestro(X , Y ) :- progenitor(X , Z ) , ancestro(Z , Y ) .
29
30 hijo_de(A , B):- progenitor(B , A).
31 abuelo_de(A , B):- progenitor(A , C), progenitor(C,B), papa(A,_).
32 abuela_de(A , B):- progenitor(A,C), progenitor(C,B), mama(A,_).
33 bisabuelo_de(A , B):- abuelo_de(C,B), papa(A,C).
34

```

Execution results for `bisabuelo_de(M,N)`:

```

Singleton variables: [A,B]
false
bisabuelo_de(A,B).
false
bisabuelo_de(A,B).
false
bisabuelo_de(A,B).
false
bisabuelo_de(A,C).
A = juan,
C = carolina
A = juan,
C = carolina
false
?- bisabuelo_de(A,C).

```

