

HECHOS

```
1 % HECHOS
2 progenitor(clara, jose).
3 progenitor(tomas, jose).
4 progenitor(tomas, isabel).
5 progenitor(jose, ana).
6 progenitor(jose, patricia).
7 progenitor(patricia, jaime).
8
9
10 hombre(tomas).
11 hombre(jose).
12 hombre(jaime).
13 mujer(clara).
14 mujer(isabel).
15 mujer(ana).
16 mujer(patricia).
17
```

REGLAS

```
17
18 % diferencia
19 dif(X, Y) :- X \= Y.
20
21 % Reglas
22 es_madre(X) :-
23     mujer(X),
24     progenitor(X,_).
25
26 es_padre(X) :-
27     hombre(X),
28     progenitor(X,_).
29
30 es_hijo(X) :-
31     hombre(X),
32     progenitor(_,X).
33
34 hermana_de(X,Y) :-
35     mujer(X),
36     progenitor(P,X),
37     progenitor(P,Y),
38     dif(X,Y).
39
40 abuelo_de(X,Y) :-
41     hombre(X),
42     progenitor(X,Z),
43     progenitor(Z,Y).
```

```
44
45 abuela_de(X,Y) :-
46     mujer(X),
47     progenitor(X,Z),
48     progenitor(Z,Y).
49
50 hermanos(X,Y) :-
51     progenitor(P,X),
52     progenitor(P,Y),
53     dif(X,Y).
54
55 tia(X,Y) :-
56     mujer(X),
57     progenitor(P,Y),
58     hermanos(X,P).
59
```

DEMOSTRACIONES

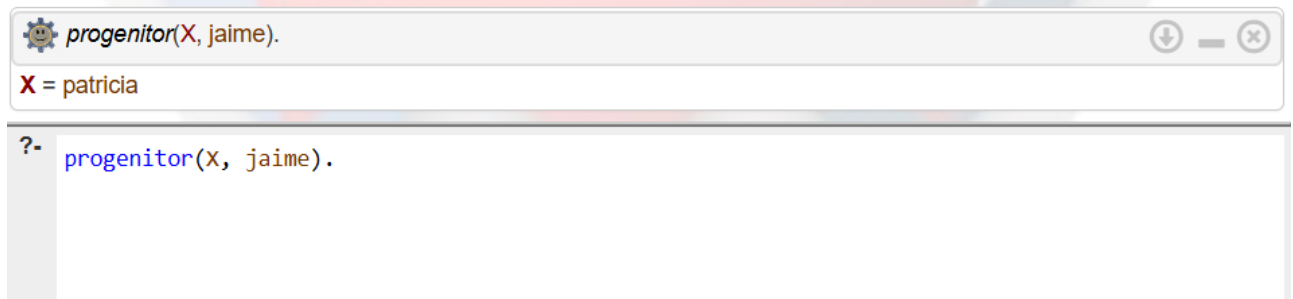
Ejercicio 1.1

a) *?- progenitor(Jaime, X).*



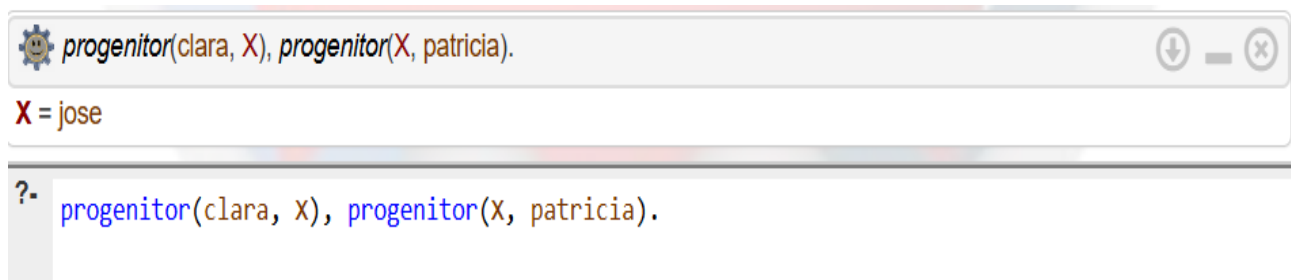
```
progenitor(jaime,X)  
false  
?- progenitor(jaime,X)
```

b) *?- progenitor(X, jaime).*



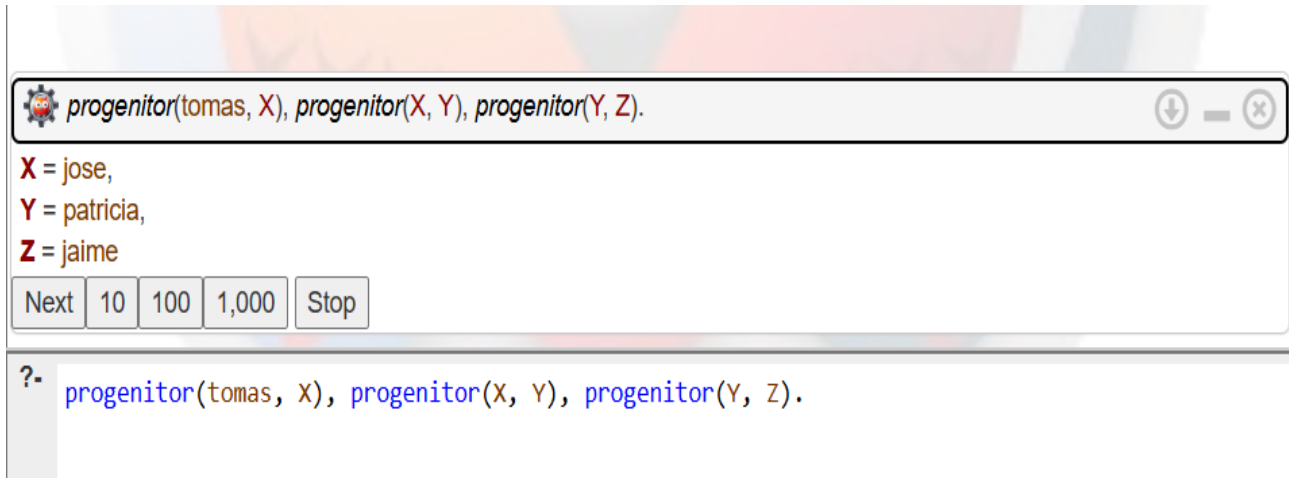
```
progenitor(X, jaime).  
X = patricia  
?- progenitor(X, jaime).
```

c) *?- progenitor(clara, X), progenitor(X, patricia).*



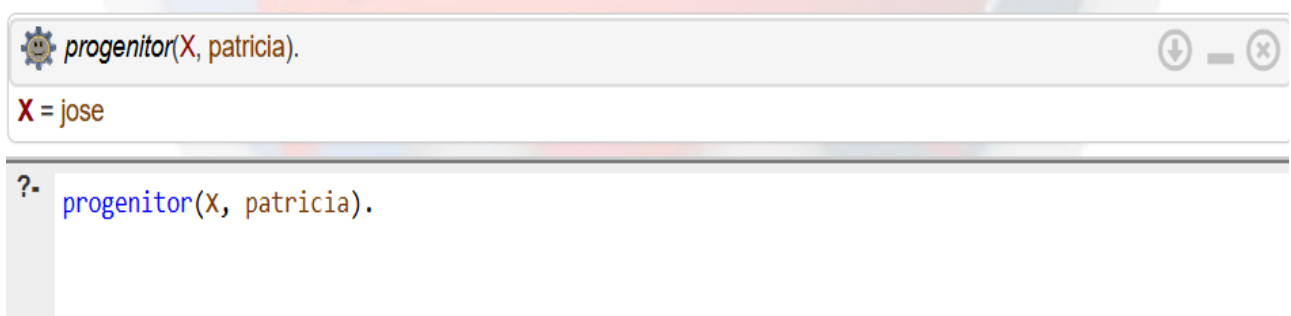
```
progenitor(clara, X), progenitor(X, patricia).  
X = jose  
?- progenitor(clara, X), progenitor(X, patricia).
```

***d) ?- progenitor(tomas, X), progenitor(X, Y),
progenitor(Y, Z).***



Ejercicio 1.2

a) ¿Quién es el progenitor de Patricia?



b) ¿Tiene Isabel un hijo o una hija?



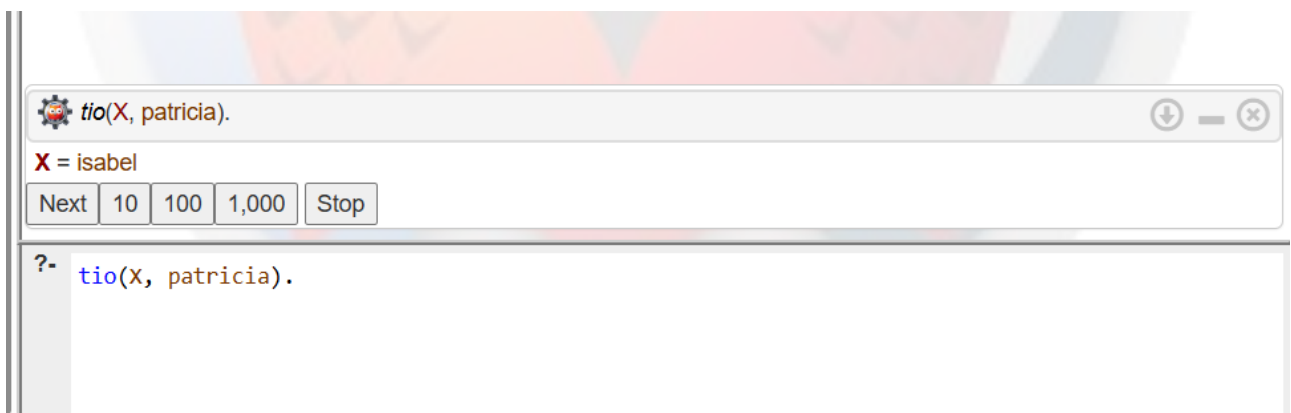
```
progenitor(isabel, X).  
false  
?- progenitor(isabel, X).
```

c) ¿Quién es el abuelo de Isabel?



```
progenitor(X, isabel), progenitor(Y, X).  
false  
?- progenitor(X, isabel), progenitor(Y, X).
```

d) ¿Cuáles son los tíos de Patricia?



```
tio(X, patricia).  
X = isabel  
Next 10 100 1,000 Stop  
?- tio(X, patricia).
```

Ejercicio 1.3

a) *es_madre(X).*

```
X = clara
X = patricia
?- es_madre(x).
```

b) *es_padre(X).*

```
X = tomas
X = tomas
X = jose
X = jose
Next 10 100 1,000 Stop
?- es_padre(x).
```

c) *es_hijo(X).*

```
X = jose
X = jose
X = jaime
?- es_hijo(x).
```

d) *hermana_de(X,Y).*

```
hermana_de(X,Y).  
X = isabel,  
Y = jose  
X = ana,  
Y = patricia  
X = patricia,  
Y = ana  
false  
?- hermana_de(X,Y).
```

e) *abuelo_de(X,Y)* y *abuela_de(X,Y)*.

```
abuelo_de(X,Y)  
X = tomas,  
Y = ana  
X = tomas,  
Y = patricia  
X = jose,  
Y = jaime  
false  
?- abuelo_de(X,Y)
```

```
abuela_de(X,Y)  
X = clara,  
Y = ana  
X = clara,  
Y = patricia  
false  
?- abuela_de(X,Y)
```


f) hermanos(X,Y).

```
hermanos(X,Y).  
X = jose,  
Y = isabel  
X = isabel,  
Y = jose  
X = ana,  
Y = patricia  
X = patricia,  
Y = ana  
false  
?- hermanos(X,Y).
```

g) tia(X,Y).

```
tia(X,Y)  
X = isabel,  
Y = ana  
X = isabel,  
Y = patricia  
X = ana,  
Y = jaime  
false  
?- tia(X,Y)
```

Conclusiones:

El ejercicio permitió comprender cómo una base de datos en Prolog puede modelar relaciones familiares simples mediante hechos y reglas lógicas. Esto muestra la potencia de la programación lógica para representar conocimiento estructurado.

A partir de un conjunto reducido de hechos, Prolog es capaz de inferir relaciones más complejas como abuelo, abuela, hermanos, tía, madre o padre. Esto refleja cómo el motor de inferencia de Prolog aplica reglas de manera recursiva y sistemática para deducir nueva información.

Las consultas permiten obtener respuestas tanto positivas como negativas, demostrando la capacidad de Prolog no solo para verificar información explícita en la base de datos, sino también para identificar relaciones implícitas

**ProgIIIIG101-Act02-
Santiago Henao
-Johany Ballesteros
-Juan Guillermo Galindo**

<https://github.com/johany-ballesteros/ProgIIIIG101-Act01-Santiago-Henao-Johany-Ballesteros-Juan-Guillermo-Galindo>