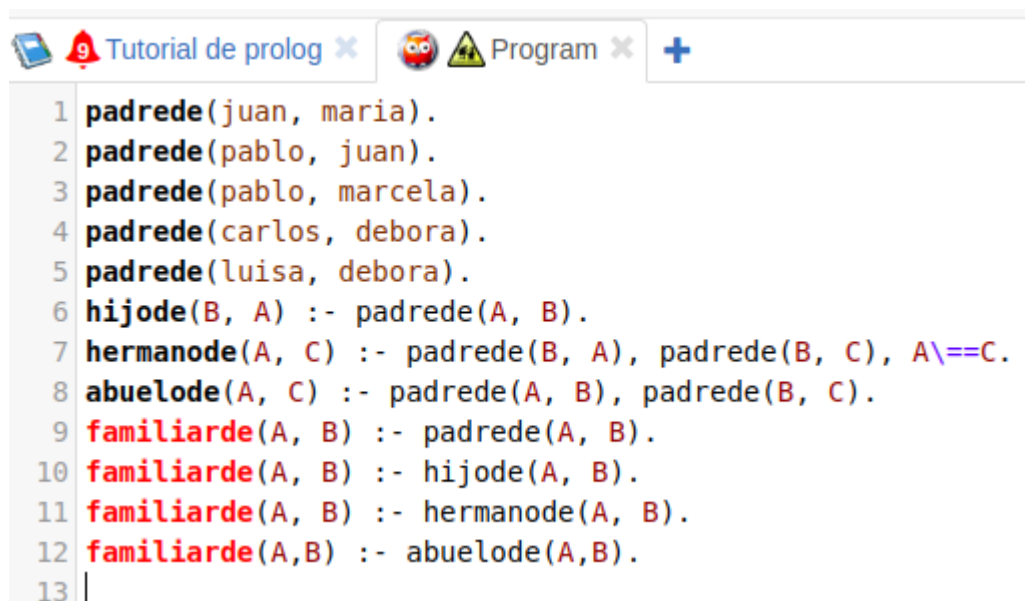


**Ejercicio 1** Edita el siguiente programa Prolog, almacenándolo en el fichero familia.pl

```
padrede(juan, maria).
padrede(pablo, juan).
padrede(pablo, marcela).
padrede(carlos, debora).
padrede(luisa, debora).
hijode(B, A) :- padrede(A, B).
hermanode(A, C) :- padrede(B, A), padrede(B, C), A\==C.
abuelode(A, C) :- padrede(A, B), padrede(B, C).
familiarde(A, B) :- padrede(A, B).
familiarde(A, B) :- hijode(A, B).
familiarde(A, B) :- hermanode(A, B).
familiarde(A,B) :- abuelode(A,B).
```

1

Carga el programa guardado en el fichero familia.pl haciendo uso de alguna de las instrucciones de carga anteriores. NO olvides que todas las consultas finalizan con un punto.



```
1 padrede(juan, maria).
2 padrede(pablo, juan).
3 padrede(pablo, marcela).
4 padrede(carlos, debora).
5 padrede(luisa, debora).
6 hijode(B, A) :- padrede(A, B).
7 hermanode(A, C) :- padrede(B, A), padrede(B, C), A\==C.
8 abuelode(A, C) :- padrede(A, B), padrede(B, C).
9 familiarde(A, B) :- padrede(A, B).
10 familiarde(A, B) :- hijode(A, B).
11 familiarde(A, B) :- hermanode(A, B).
12 familiarde(A,B) :- abuelode(A,B).
13 |
```

He cargado el programa en el editor online (SWISH)

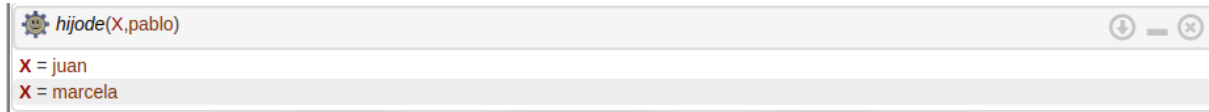
**Ejercicio 2** *¿Qué ocurre si preguntas quiénes son los hijos de Pablo? Para ello reescribe la consulta del ejemplo anterior cambiando la variable X por el término pablo y el término maria por una variable (por ejemplo X). ¿Qué ocurre cuando pulsas la tecla r después de la primera respuesta?*

Al preguntar por los hijos de pablo aparece esto:



A screenshot of a Prolog query window titled 'hijode(X,pablo)'. The window has a title bar with standard OS controls. Below the title bar, the text 'X = juan' is displayed. At the bottom of the window, there is a row of buttons: 'Next', '10', '100', '1,000', and 'Stop'.

Si le doy al Next, me aparece marcela (y ya serían todos los hijos)



A screenshot of a Prolog query window titled 'hijode(X,pablo)'. The window has a title bar with standard OS controls. Below the title bar, the text 'X = juan' is displayed on the first line, and 'X = marcela' is displayed on the second line.

### Ejercicio 3 Haciendo uso de familia.pl:

*Consultar quién es abuelo de María.*

*Consultar todos los familiares de Pablo conocidos.*

*Consultar todas las personas que son familia de Débora.*

*Usando un editor de texto, añadir al fichero familia.pl una regla que defina la relación es nieto de similar a es hijo de pero usando el predicado abuelode en lugar de padrede. Guarda los cambios y recarga el fichero con el predicado reconsult(familia). Averigua de quién es nieta María usando el predicado nietode.*

*Lanza la consulta familiarde(A,B). Añade al programa la regla familiarde(A,B) :- nietode(A,B) y reconsultalo. ¿En qué difieren las dos respuestas computadas?*

El abuelo de María:

```
abuelode(X, maria).  
X = pablo  
false
```

Todos los familiares de Pablo:

```
familiarde(pablo, X).  
X = juan  
X = marcela  
X = maria  
false
```

Todas las personas que son familia de Débora:

```
familiarde(X, debora).  
X = carlos  
X = luisa  
false
```

Regla que define es nieto de:

```
nietode(A, B) :- abuelode(B, A).
```

De quién es nieta María:

```
nietode(maria, X).  
X = pablo  
false
```

Lanzar consulta familiarde(A,B):

```
familiarde(A, B).  
A = juan,  
B = maria  
A = pablo,  
B = juan  
A = pablo,  
B = marcela  
A = carlos,  
B = debora  
A = luisa,  
B = debora  
A = maria,  
B = juan  
A = juan,  
B = pablo  
A = marcela,  
B = pablo  
A = debora,  
B = carlos  
A = debora,  
B = luisa  
A = juan,  
B = marcela  
A = marcela,  
B = juan  
A = pablo,  
B = maria  
false
```

Añado a regla familiarde(A, B) :- nietode(A, B)

```
familiarde(A, B) :- nietode(A, B).
```

Vuelvo a consolutar y el resultado cambia:

```
familiarde(A, B).  
A = juan,  
B = maria  
A = pablo,  
B = juan  
A = pablo,  
B = marcela  
A = carlos,  
B = debora  
A = luisa,  
B = debora  
A = maria,  
B = juan  
A = juan,  
B = pablo  
A = marcela,  
B = pablo  
A = debora,  
B = carlos  
A = debora,  
B = luisa  
A = juan,  
B = marcela  
A = marcela,  
B = juan  
A = pablo,  
B = maria  
A = maria,  
B = pablo  
false
```

Ahora hay un resultado más, el último, A = maria y B = juan. Esto es porque ahora ser nieta de también significa ser familiar de.

El programa al final del ejercicio:

```
padrede(juan, maria).
padrede(pablo, juan).
padrede(pablo, marcela).
padrede(carlos, debora).
padrede(luisa, debora).
hijode(B, A) :- padrede(A, B).
hermanode(A, C) :- padrede(B, A), padrede(B, C), A\==C.
abuelode(A, C) :- padrede(A, B), padrede(B, C).
nietode(A, B):- abuelode(A, B).
familiarde(A, B) :- padrede(A, B).
familiarde(A, B) :- hijode(A, B).
familiarde(A, B) :- hermanode(A, B).
familiarde(A,B) :- abuelode(A,B).
familiarde(A, B) :- nietode(A, B).
```

**Ejercicio 4** La unificación es el mecanismo que permite dar valor a las variables en los objetivos. Para que dos predicados unifiquen han de tener el mismo nombre, la misma aridad y deben unificar cada uno de los términos de sus parámetros. En Prolog el operador = representa la unificación. Comprueba si unifican o no los siguientes términos o átomos usando el intérprete de Prolog:

```
padrede(X,debor) y padrede(carlos,debor)
X y fecha(10,nov,Y)
fecha(10,oct,2013) y fecha(X,nov,2013)
momento(fecha(10,nov,2013),Y) y momento(X,hora(13,05))
```

```
padrede(X,debor) = padrede(carlos,debor)
X = carlos
```

Sí unifica

```
X = fecha(10,nov,Y)
X = fecha(10,nov,Y)
```

Sí unifica

```
fecha(10,oct,2013) = fecha(X,nov,2013)
false
```

No unifica

```
momento(fecha(10,nov,2013),Y) = momento(X,hora(13,05))
X = fecha(10,nov,2013),
Y = hora(13,5)
```

Sí unifica

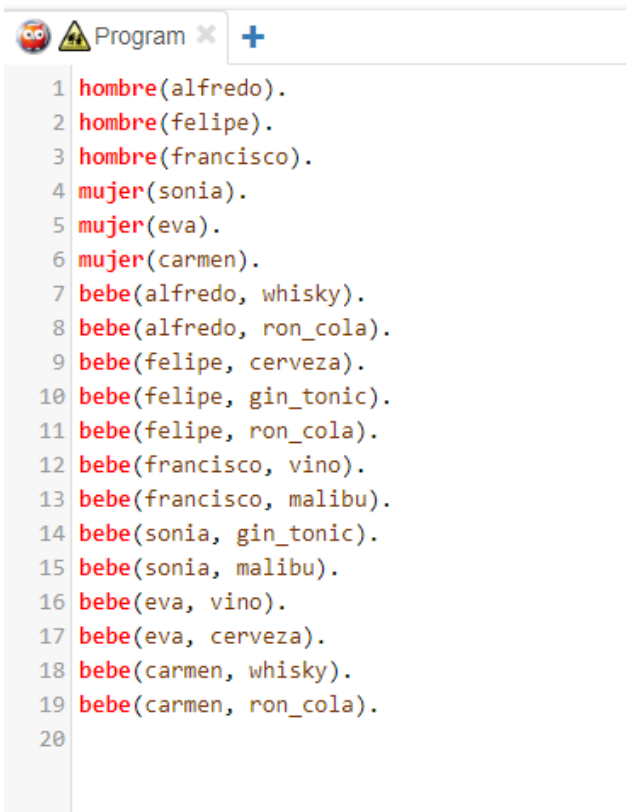
**Ejercicio 5** Edita el siguiente programa Prolog y almacenálo en el fichero `fiesta.pl`

```
hombre(alfredo).  
hombre(felipe).  
hombre(francisco).  
mujer(sonia).  
mujer(eva).  
mujer(carmen).  
bebe(alfredo, whisky).
```

3

```
bebe(alfredo, ron_cola).  
bebe(felipe, cerveza).  
bebe(felipe, gin_tonic).  
bebe(felipe, ron_cola).  
bebe(francisco, vino).  
bebe(francisco, malibu).  
bebe(sonia, gin_tonic).  
bebe(sonia, malibu).  
bebe(eva, vino).  
bebe(eva, cerveza).  
bebe(carmen, whisky).  
bebe(carmen, ron_cola).
```




el fichero cargado en el swish:



```
1 hombre(alfredo).  
2 hombre(felipe).  
3 hombre(francisco).  
4 mujer(sonia).  
5 mujer(eva).  
6 mujer(carmen).  
7 bebe(alfredo, whisky).  
8 bebe(alfredo, ron_cola).  
9 bebe(felipe, cerveza).  
10 bebe(felipe, gin_tonic).  
11 bebe(felipe, ron_cola).  
12 bebe(francisco, vino).  
13 bebe(francisco, malibu).  
14 bebe(sonia, gin_tonic).  
15 bebe(sonia, malibu).  
16 bebe(eva, vino).  
17 bebe(eva, cerveza).  
18 bebe(carmen, whisky).  
19 bebe(carmen, ron_cola).  
20
```



- a) Define un predicado `pareja(X, Y)` que tenga éxito cuando `X` es un hombre e `Y` una mujer y tengan al menos una bebida favorita en común. Averigua qué parejas tienen Alfredo y Francisco.

```
pareja(X, Y) :- hombre(X), mujer(Y), bebe(X, Bebida), bebe(Y, Bebida).
```

 `pareja(alfredo, X).`   

`X = carmen`

`X = carmen`

 `pareja(francisco, X).`   

`X = sonia`

`X = eva`

`false`



b) Modifica el programa para reflejar los siguientes hábitos de bebida:

Pepe bebe cualquier cosa que beba Alfredo.

Elena bebe cualquier cosa que beban Sonia o Felipe.

```
bebe(pepe, X) :- bebe(alfredo, X).
```

bebe(pepe, X).

X = whisky

X = ron\_cola

```
bebe(elena, X) :- bebe(sonia, X).
```

```
bebe(elena, X) :- bebe(felipe, X).
```

bebe(elena, X).

X = gin\_tonic

X = malibu

X = cerveza

X = gin\_tonic

X = ron\_cola

c) Averigua qué parejas tienen Pepe y Elena.



Programa al final del ejercicio:

hombre(alfredo).

hombre(felipe).

hombre(francisco).

hombre(pepe).

mujer(sonia).

mujer(eva).

mujer(carmen).

mujer(elena).

bebe(alfredo, whisky).

bebe(alfredo, ron\_cola).

bebe(felipe, cerveza).

bebe(felipe, gin\_tonic).

bebe(felipe, ron\_cola).

bebe(francisco, vino).

bebe(francisco, malibu).

bebe(sonia, gin\_tonic).

bebe(sonia, malibu).

bebe(eva, vino).

bebe(eva, cerveza).

bebe(carmen, whisky).

bebe(carmen, ron\_cola).

bebe(pepe, X) :- bebe(alfredo, X).

bebe(elena, X) :- bebe(sonia, X).

bebe(elena, X) :- bebe(felipe, X).

pareja(X, Y) :- hombre(X), mujer(Y), bebe(X, Bebida), bebe(Y, Bebida).

### Ejercicio 6 Sea el siguiente enunciado:

Bertoldo y Bartolo son rufianes.  
Romeo y Bertoldo, como su nombre indica, son nobles.  
Bartolo es un plebeyo.  
Gertrudis y Julieta son damas.  
Julieta es hermosa.  
Los plebeyos desean a cualquier dama, mientras que los nobles solo a aquellas que son hermosas.  
Los rufianes, para satisfacer sus instintos, raptan a las personas a las que desean.

Representa el enunciado como un programa Prolog. Responde a las siguientes preguntas:

```
Program +
1 rufian(bertoldo).
2 rufian(bartolo).
3 noble(romeo).
4 noble(bertoldo).
5 plebeyo(bartolo).
6 dama(gertrudis).
7 dama(julieta).
8 hermosa(julieta).
9
10 desea(X,Y) :- plebeyo(X), dama(X).
11 desea(X,Y) :- noble(X), dama(X), hermosa(X).
12
13 rapta(X,Y) :- rufian(X), desea(X,Y).
```

¿Qué noble es un rufián?

```
rufian(X), noble(X)
X = bertoldo
false
```

¿A quién podría raptar Romeo?

```
rapta(romeo,X).
false
```

no podría raptar a nadie porque Romeo no es un rufián

¿Quién puede raptar a Julieta?

```
⚙️ rapta(X,julieta).  
X = bertoldo  
X = bartolo  
false
```

¿Quién rapta a quién?

```
⚙️ rapta(X,Y).  
X = bertoldo,  
Y = julieta  
X = bartolo,  
Y = gertrudis  
X = bartolo,  
Y = julieta  
false
```

¿A quién desea Bartolo?

```
⚙️ desea(bartolo,Y).  
Y = gertrudis  
Y = julieta  
false
```

¿Y Romeo?

```
⚙️ desea(romeo,Y).  
Y = julieta
```

¿Qué hermosa dama es deseada por Bartolo?

```
⚙️ desea(bartolo,Y), dama(Y), hermosa(Y).  
Y = julieta  
false
```

Programa al final del ejercicio:

rufian(bertoldo).

rufian(bartolo).

noble(romeo).

noble(bertoldo).

plebeyo(bartolo).

dama(gertrudis).

dama(julieta).

hermosa(julieta).

desea(X,Y) :- plebeyo(X), dama(Y).

desea(X,Y) :- noble(X), dama(Y), hermosa(Y).

rapta(X,Y) :- rufian(X), desea(X,Y).

**Ejercicio 7** Edita el siguiente programa Prolog y almacénalo en el fichero `menu.pl`

```
primero(ensalada).  
primero(sopa).  
primero(revuelto).  
segundo(chuleta).  
segundo(lubina).  
segundo(pollo).  
postre(natillas).  
postre(flan).
```

4

```
postre(cafe).  
comida(X, Y, Z):- primero(X), segundo(Y), postre(Z).
```

Este programa describe la información con la que un restaurante confecciona el menú del día, el cual está formado por un primero, un segundo y postre. Utiliza el predicado `comida/3` para determinar las distintas comidas que puede servir el restaurante. ¿Puedes predecir el orden en que se obtienen estas comidas? ¿De qué depende este orden?

comida(X,Y,Z)

X = ensalada,  
Y = chuleta,  
Z = natillas

X = ensalada,  
Y = chuleta,  
Z = flan

X = ensalada,  
Y = chuleta,  
Z = cafe

X = ensalada,  
Y = lubina,  
Z = natillas

X = ensalada,  
Y = lubina,  
Z = flan

Next 10 100 1,000 Stop

Sí se puede predecir el orden en que se obtienen las comidas.

Primero va cambiando los valores del 3er argumento, luego del segundo y finalmente del primero. Y los valores cambian dependiendo del orden en el que están en la base de conocimiento.