# Relación 1: Revisión de Prolog

1. Escribir un programa que tenga como cláusulas

   (a) *Todo hombre es mortal*

   (b) *Sócrates es un hombre*

   y utilizarlo para demostrar que *Sócrates es mortal*.

2. Tomando la notación `0, s(0), s(s(0)), ...` para los números naturales, encontrar un programa que encuentre los números impares.

3. Añade la cláusula `nat(a)` *al final* del programa `naturales.pl`

   - ¿Prueba este nuevo programa que `a` es un número natural?
   - Pídele al programa que encuentre un número natural, luego otro y otro, ...
     ¿Descubrirá Prolog que `a` es un número natural?
   - ¿Qué ocurre si en lugar de añadirlo *al final* lo añadimos *al principio*?.

4. ¿Son unificables `X` y `f(X)`?. Compruébalo en Prolog.

5. Considera el siguiente programa `caminos.pl`

   ```
   camino(X,Z) :- arco(X,Y),camino(Y,Z).
   camino(U,U).
   arco(b,c).
   ```

   - Dibuja el árbol de resolución para `?- camino(V,c)`
   - ¿Qué ocurriría si en el programa `caminos.pl` tomáramos la función de selección que elige el último literal de la cláusula?

6. [**Poole-98 pp. 81–86**] Escribir definiciones para los siguientes predicados usando, si se considera oportuno, los predicados ya definidos.

   (a) `pertenece_1(X,L)` si `X` es un elemento de la lista `L` (usando `conc`).

   (b) `pertenece_2(X,L)` si `X` es un elemento de la lista `L` (sin usar `conc`).

   (c) `inversa(L1,L2)` si `L2` es la lista `L1` en orden inverso.

   (d) `ultimo_1(X,L)` si `X` es el último elemento de la lista `L` (usando `conc`).

   (e) `ultimo_2(X,L)` si `X` es el último elemento de la lista `L` (sin usar `conc`).

   (f) `borrado_1(X,L1,L2)` si al borrar *una ocurrencia* del elemento `X` de la lista `L1` obtenemos la `L2`.

   (g) `borrado_2(X,L1,L2)` si al borrar *todas las ocurrencias* del elemento `X` de la lista `L1` obtenemos la lista `L2`.

   (h) `sublista(L1,L2)` si `L1` es una sublista de `L2`.

(i) `palindromo(L)` si `L` es un palíndromo.

7. ¿Son unificables `L` y `[a|L]`?. Compruébalo en Prolog.

8. [**Poole–98 p. 66**] Calcular un unificador de máxima generalidad para cada uno de los siguientes pares de expresiones:

   (a) `p(f(X),g(g(b)))` y `p(Z,g(Y))`.

   (b) `g(f(X),r(X),t)` y `g(W,r(Q),Q)`.

   (c) `igual(suma(Y,b),Z)` e `igual(P,P)`.

9. Pon la nueva cláusula `nat(a)` *al principio* del programa `naturales.pl` y dibuja el árbol de resolución.

10. [**Flach–94 p.47**] Dado el programa

```
lista(nil).
lista(cons(X,Y)) :- lista(Y).
```

dibujar el árbol de resolución SLD correspondiente a la pregunta `?- lista(L).`

11. [**Van Le–93 p. 15**] Considera las siguientes afirmaciones en castellano:

```
Toda madre ama a su hijo si su hijo es bueno.
Toda madre es una mujer
Ana es una mujer
El marido de Ana es bueno
```

Vamos a trasladar ese conocimiento a dos programa Prolog, uno con símbolos de función y otro sin ellos.

```
% Programa 1:
ama(madre(X),X) :- es_bueno(X).
es_mujer(madre(X)).
es_mujer(ana).
es_bueno(marido(ana)).

% Programa 2:
ama(X,Y) :- madre(X,Y), es_bueno(Y).
es_mujer(X) :- madre(X,Y).
es_mujer(ana).
es_bueno(X) :- marido(X,ana).
```

- Da razones de por qué el programa 1 es más expresivo que el programa 2.
- Escribe una cuestión para preguntar si existe alguna mujer que ame al marido de alguien. ¿Cuál es la respuesta de cada uno de los programas?

- Completa el programa 2 para poder dar respuesta a la cuestión del apartado anterior

12. [**Bratko–86 p. 91**](**Algoritmo de Euclides**) Dados dos enteros positivos `X` e `Y`, el máximo común divisor `D` puede obtenerse de la siguiente manera:

    - Si `X` e `Y` son iguales, entonces `D` es igual a `X`
    - Si `X<Y`, entonces `D` es igual al máximo común divisor de `X` y la diferencia `Y-X`.
    - Si `Y<X` entonces hacemos lo mismo que en caso anterior con `X` e `Y` intercambiados.

    Define el predicado `mcd(X,Y,D)` que calcule el máximo común divisor `D` de los enteros positivos `X` e `Y`.

13. [**Bratko–86 p. 128**] Dado el programa

    ```
    p(1).
    p(2) :- !.
    p(3).
    ```

    dibujar los árboles de resolución SLD correspondiente a las preguntas

    - `?- p(X).`
    - `?- p(X), p(Y).`
    - `?- p(X), !, p(Y).`

14. [**Bratko–86 p. 129**] La siguiente relación clasifica los números en tres categorías: positivo, cero y negativo:

    ```
    clase(Numero,positivo) :- Numero > 0.
    clase(0,cero).
    clase(Numero,negativo) :- Numero < 0.
    ```

    Definir este procedimiento de una manera más eficiente usando cortes.

15. [**Flach–94 p. 56**] Dado el programa

    ```
    soltero(X) :- no(casado(X)), hombre(X).
    hombre(federico).
    hombre(pedro).
    casado(pedro).
    ```

    (a) Dibujar los árboles de resolución SLD correspondiente a las preguntas

    - `?- soltero(federico).`
    - `?- soltero(pedro).`
    - `?- soltero(X).`

(b) ¿Qué modificación hay que hacer en el programa para obtener la respuesta correcta a la pregunta `?- soltero(X).`?

16. [**Flach–94 p. 62**] Definir el predicado `cero(A,B,C,X)` de forma que, dados los coeficientes `A`, `B` y `C`, calcule ambos valores de `X` para los cuales `A*X^2+B*X+C = 0`. Por ejemplo,

```
?- cero(1,-5,6,X).
X = 3 ;
X = 2 ;
No
```

[Indicación: `sqrt(X)` es la raíz cuadrada de `X`].

17. [**Flach–94 p. 63**] Dado el programa

```
longitud([],0).
longitud([X|R], N) :-
   longitud(R,M),
   N is M+1.
```

dibujar el árbol de resolución SLD correspondiente a la pregunta `?- longitud([a,b,c],N).`

18. [**Flach–94 p. 63**] Dado el programa

```
longitud(L,N) :-
   longitud_ac(L,0,N).

longitud_ac([],N,N).
longitud_ac([X|R],N0,N) :-
   N1 is N0+1,
   longitud_ac(R,N1,N).
```

dibujar el árbol de resolución SLD correspondiente a la pregunta `?- longitud([a,b,c],N).`

19. [**Flach–94 p. 64**] Dado el programa

```
inversa([],[]).
inversa([X|L1],L2) :-
   inversa(L1,L3),
   conc(L3,[X],L2).

conc([],L,L).
conc([X|L1],L2,[X|L3]) :-
   conc(L1,L2,L3).
```

dibujar el árbol de resolución SLD correspondiente a la pregunta `?- inversa([a,b,c],R).`