

**Ejercicio 5.1.** Definir la relación `divisores(+N,-L)` que se verifica si L es la lista de los divisores propios del número N. Por ejemplo,

```
?- divisores(24,L).
L = [1, 2, 3, 4, 6, 8, 12, 24]
```

**Ejercicio 5.2.** Definir la propiedad `primo(+N)` que se verifica si N es un número primo. Por ejemplo,

```
?- primo(7).    =>  Yes
?- primo(9).    =>  No
```

[Indicación: Hacer distintas definiciones basadas en las siguientes ideas:

1. X es primo si el número de divisores de X es menor que 3.
2. X es primo si no hay ningún número entre 2 y la raíz cuadrada de X que sea un divisor de X.
3. X es primo si no tiene ningún divisor primo entre 2 y su raíz cuadrada.]

**Ejercicio 5.3.** Se considera el siguiente polinomio  $p(X) = X^2 - X + 41$ . Definir la siguiente relación `genera_primo(+X,+Y,-L)` que se verifica si L es el conjunto de los pares (A,p(A)) tales A es un número del intervalo [X,Y] y p(A) es primo. Por ejemplo,

```
?- genera_primo(5,7,L).
L = [5-61, 6-71, 7-83]
```

Comprobar que p(A) es primo para A desde 1 hasta 40, pero no es primo para A=41.

**Ejercicio 5.4.** Se dice que dos números X e Y son amigos cuando X es igual a la suma de los divisores de Y (exceptuando al propio Y) y viceversa. Por ejemplo, 6 es amigo de sí mismo puesto que los divisores de 6 (exceptuando al 6) son 1, 2 y 3 y  $6=1+2+3$ . Igualmente 28 es amigo de sí mismo, puesto que  $28=1+2+4+7+14$ . Se pide definir la relación `amigos(+A,+B,-L)` que se verifica si L es la lista de las parejas de números amigos comprendidos entre A y B. Por ejemplo,

```
?- amigos(1,300,L).
L = [6-6, 28-28, 220-284, 284-220]
```

**Ejercicio 5.5.** Definir la relación `lista_a_conjunto(+L,-C)` que se verifica si C es el conjunto de los elementos de la lista no vacía L. Por ejemplo,

```
?- lista_a_conjunto([b,c,d,a,c,f,a],C).
C = [a, b, c, d, f]
```

**Ejercicio 5.6.** Definir la relación `para_todos(+P,+L)` que se verifica si todos los elementos de la lista L cumplen la propiedad P. Por ejemplo,

```
?- para_todos(number,[1,2,3]).
Yes
?- para_todos(number,[1,2,3,a]).
No
```

**Ejercicio 5.7.** Definir la relación `existe(+P,+L)` que se verifica si existe algún elemento de la lista `L` que cumple la propiedad `P`. Por ejemplo,

```
?- existe(number,[a,b,c]).  
No  
?- existe(number,[a,1,b,c]).  
Yes
```

**Ejercicio 5.8.** Definir la relación `sublista(+P,+L1,?L2)` que se verifica si `L2` es la lista de los elementos de `L1` que verifican la propiedad `P`. Por ejemplo,

```
?- sublista(number,[1,a,2,b,1],L).  
L = [1, 2, 1]
```

(Nota: `sublista/3` corresponde a la relación definida `sublist/3`).

**Ejercicio 5.9.** [Ex. Feb. 2000] Definir la relación `rotaciones(+L1,-L2)` que se verifica si `L2` es la lista cuyos elementos son las listas formadas por las sucesivas rotaciones de los elementos de la lista `L1`. Por ejemplo,

```
?- rotaciones([1,2,3,4],L).  
L = [[1, 2, 3, 4], [2, 3, 4, 1], [3, 4, 1, 2], [4, 1, 2, 3]]
```

**Ejercicio 5.10.** [Ex. Feb. 2000] Definir la relación `capicúas(N,L)` que se verifica si `L` es lista formada por todos los números enteros positivos capicúa menores que `N`. Por ejemplo,

```
?- capicúas(100,L).  
L = [1,2,3,4,5,6,7,8,9,11,22,33,44,55,66,77,88,99]
```

**Ejercicio 5.11.** Definir la relación `invierte_palabra(P1,P2)` que se verifica si `P2` es la palabra formada invirtiendo el orden de las letras de la palabra `P1`. Por ejemplo,

```
?- invierte_palabra(arroz,P).  
P = zorra
```