MÁS EJERCICIOS DE PROLOG CON LISTAS

1. Escribir un programa en PROLOG que, dada una lista numérica, devuelva otra cuyo resultado sea la suma de los elementos de la original de 2 en 2. Si solo queda un elemento por sumar, el resultado de esta suma será el propio elemento.

```
 ?- sumar2en2([5,3,4,2,1],L).
L = [8, 6, 1] ;
false.

?- sumar2en2([5,3,4,2],L).
L = [8, 6].

?- sumar2en2([5],L).
L = [5] ;
false.

?- sumar2en2([],L).
L = [].
```

2. Transformar un número decimal a binario en forma de lista.

```
?- dec_bin(0,L).
L = [] ;
false.

?- dec_bin(1,L).
L = [1] ;
false.

?- dec_bin(22,L).
L = [1, 0, 1, 1, 0] ;
false.
```

3. El predicado *familia* tiene como argumentos los miembros de una familia. El primero es el padre, el segundo la madre y el tercero es la lista de hijos. Cada componente de la familia se expresa mediante la función *persona(Nombre,Apellido1,Edad)*.

```
familia(persona(juan,perez,50),
        persona(maria,alonso,45),
        [persona(carlos,perez,20),
        persona(andres,perez,18),
        persona(elena,perez,12)]).

familia(persona(pedro,lopez,40),
        persona(carmen,ruiz,39),
        [persona(carlos,lopez,19),
        persona(teresa,lopez,8)]).

familia(persona(carlos,martinez,25),
        persona(lola,garcia,22),
        []).

edad(persona(_,_,E),E).
```

Añadir a este programa las líneas necesarias para:

a) Obtener **el hijo menor** de la familia, sabiendo que los hijos están ordenados por edad de mayor a menor.

```
?- ultimo(X,Y,Z).
X = persona(juan, perez, 50),
Y = persona(maria, alonso, 45),
Z = persona(elena, perez, 12) ;
X = persona(pedro, lopez, 40),
Y = persona(carmen, ruiz, 39),
Z = persona(teresa, lopez, 8) ;
false.
```

b) Obtener **las familias** que tienen un **número de hijos par o impar** (Cero se considera par).

```
?- numhijospar(X,Y,H,Num).
X = persona(pedro, lopez, 40),
Y = persona(carmen, ruiz, 39),
H = [persona(carlos, lopez, 19), persona(teresa, lopez, 8)],
Num = 2 ;
X = persona(carlos, martinez, 25),
Y = persona(lola, garcia, 22),
H = [],
Num = 0.

?- numhijosimpar(X,Y,H,Num).
X = persona(juan, perez, 50),
Y = persona(maria, alonso, 45),
H = [persona(carlos, perez, 20), persona(andres, perez, 18), persona(elena, perez, 12)],
Num = 3 ;
false.
```

c) Calcular **la media de edad** de la familia en decimal.
d) Calcular **la suma total de edad** de la familia en binario.

```
?- media(X,Y,MediaDec,T,TotalBin).
X = persona(juan, perez, 50),
Y = persona(maria, alonso, 45),
MediaDec = 29,
T = 145,
TotalBin = [1, 0, 0, 1, 0, 0, 0, 1] ;
X = persona(pedro, lopez, 40),
Y = persona(carmen, ruiz, 39),
MediaDec = 26.5,
T = 106,
TotalBin = [1, 1, 0, 1, 0, 1, 0] ;
X = persona(carlos, martinez, 25),
Y = persona(lola, garcia, 22),
MediaDec = 23.5,
T = 47,
TotalBin = [1, 0, 1, 1, 1, 1] ;
false.
```

NOTA 1: Tener en cuenta que el programa debe permitir hacer consultas que obtengan el resultado para una familia, para varias familias que cumplan un determinado patrón o para toda la base de datos.

NOTA 2: No se permiten usar predicados predefinidos.

NOTA 3: La base de datos puede tener más familias con distinto número de hijos que el de los ejemplos y el programa debería funcionar.

NOTA 4: Las consultas ejemplo pueden ser distintas. Por ejemplo, los apartados c y d se pueden hacer de forma independiente.

4. Decir si una lista tiene todos sus elementos iguales.

   ?-todos_iguales([5,3,4,2,1]).
   false.

   ?- todos_iguales([5,5,5]).
   true ;
   false.

5. Transformar un número natural en una lista de sus cifras como elementos.

   ?- cifras_lista(1234,R).
   R = [1, 2, 3, 4] ;
   false.

   ?- cifras_lista(1034,R).
   R = [1, 0, 3, 4] ;
   false.

   ?- cifras_lista(1,R).
   R = [1] ;
   false.

   ?- cifras_lista(0,R).
   R = [0] ;
   false.

6. Rotar una lista una posición hacia la derecha

   ?- rotar_derecha([1,2,3,4,5],R).
   R = [5, 1, 2, 3, 4] ;
   false.

   ?- rotar_derecha([],R).
   R = [] ;
   false.

   ?- rotar_derecha([1],R).
   R = [1] ;
   false.

7. Rotar una lista N posiciones hacia la derecha.

```
?- rotar_derecha_posiciones([],3,R).
R = [] ;
false.

?- rotar_derecha_posiciones([1],3,R).
R = [1] ;
false.

?- rotar_derecha_posiciones([1,2,3],3,R).
R = [1, 2, 3] ;
false.

?- rotar_derecha_posiciones([1,2,3,4,5],3,R).
R = [3, 4, 5, 1, 2] ;
false.
```

8. Construir una lista con un elemento repetido N veces.

```
?- repetir(5,7,R).
R = [5, 5, 5, 5, 5, 5, 5] ;
false.

?- repetir(a,8,R).
R = [a, a, a, a, a, a, a, a] ;
false.

?- repetir(b,0,R).
R = [] ;
false.

?- repetir(c,1,R).
R = [c] ;
false.
```

9. Contar las veces que aparece un elemento en una lista.

```
?- contar(a,[a,b,c,a,d,a],R).
R = 3 ;
false.

?- contar(a,[1,2,3,4],R).
R = 0 ;
false.

?- contar(a,[],R).
R = 0 ;
false.

?- contar(a,[a],R).
R = 1 ;
false.
```

10. Decir si una lista tiene todos sus elementos distintos.

```
?- distintos([1,2,3,4,5,6]).
true ;
false.

?- distintos([1,2,3,2,3,1]).
false.

?- distintos([1,a,3,2,7,a]).
false.

?- distintos([]).
true.

?- distintos([1]).
true ;
false.
```