Taller VPL prolog.

Jaider Castañeda Villa
Mónica Sofía Restrepo León
Grupo 1

1. Crear una rutina recursiva en Prolog que permita invertir un número dado. Ejemplo: 123456789 -> 987654321

```
%se hace un contador de la longitud del numero
count(0,0).
%arriba el caso base del contador

% método principal con recursión
count(N,R):-
        N>0,    %para el problema se debe comprobar que sea mayor a 0
        N1 is N//10,            %división entera sobre 10
        count(N1,R1),   %va alimentando la regla count hasta llegar al caso base
        R is R1+1.      %se agrega de a uno al R final por cada vez que se hizo la recursion
%caso base inversion
invertir(0,0).

invertir(X,Y):-
        X>0,
        X1 is X//10,
        Z is X mod 10, %este nos da el digito de las unidades del actual numero
        count(X1,C),
        invertir(X1,Y1),
        Y is Z*10**C+Y1. %aqui multiplicamos el el digito de las unidades por un %factor de 10 elevado a la cantidad de digitos que hay en el numero para que %sucede la inversion y se le suma el siguiente hasta que se invierte el numero
```

2. Implementar el ejercicio anterior utilizando recursión de cola (Tail Recursion)

```
invierte(N,R):-invertir_tail(N,0,R).

invertir_tail(0,R,R).
%se aplica una logica similar al anterior pero con el tail recursion
invertir_tail(N,Acc,R):- N>0,
        N1 is div(N, 10),
        UltimoDigito is mod(N, 10),
        Acc1 is Acc * 10 + UltimoDigito,
        invertir_tail(N1, Acc1, R).
```

3. **IMPLEMENTAR: Recursivo que quita todas las ocurrencias de un elemento en una lista**
   ejemplo 1: quitar(2, [3,5,8,6,2,3,2,3,2], N) el resultado es N = [3,5,8,6,3,3]
   ejemplo 2: quitar(g, [d,a,d,g,g,s,d,f], N) el resultado es N = [d,a,d,s,d,f]

```
quitar(_, [], []).  %caso base

quitar(X, [X|T], L) :-
        eliminar(X, T, L). %si la cabeza fuera igual a X no pasaria nada

quitar(X, [H|T], [H|L]) :-
        quitar(X, T, L),
        X \= H. %se hace una comprobacion de abajo para arriba y si X \= H se agrega a la lista de la derecha y la del centro, pero si fuera igual solo se agrega a la del centro
```

4. Encuentra la representación recursiva de cola (Tail recursion) para los siguientes casos:

a. fibonacci(X,Y), Y es el X-simo número de Fibonacci.

```
fibonacci(X,Y):-fibonacci_tail(X,0,1,Y).

fibonacci_tail(0,_,Y,Y).

fibonacci_tail(X,Penultimo,Ultimo,Y):-
        X>0,
        Siguiente is Penultimo+Ultimo,
        X1 is X-1,
        fibonacci_tail(X1,Ultimo,Siguiente,Y).
%se va acumulando el ultimo y anterior hasta que lleguemos al caso base
```

b. producto(X,Y,Z), Z es el producto de X por Y.

```
factorial_tail(0,Y,Y).

factorial_tail(X,Acc,Y):-
        X>0,
        Acc1 is X*Acc,
        X1 is X-1,
        factorial_tail(X1,Acc1,Y).

producto(X,Y,Z):-
        8factorial_tail(X,1,V),
        factorial_tail(Y,1,T),
        Z is V*T.
```

5. resolver un sudoku

% HECHOS

```
num(1). num(2). num(3). num(4).

% REGLAS
unicos(P,Q,R,S) :- num(P), num(Q), num(R), num(S),
                   \+ P=Q, \+ P=R, \+ P=S, \+ Q=R, \+ Q=S, \+ R=S.

% IDEA ESTRUCTURA DE LA SOLUCIÓN
sudoku(R11,R12,R13,R14,
       R21,R22,R23,R24,
       R31,R32,R33,R34,
       R41,R42,R43,R44) :-
    /*filas*/
    unicos(R11,R12,R13,R14),
    unicos(R21,R22,R23,R24),
    unicos(R31,R32,R33,R34),
    unicos(R41,R42,R43,R44),
    /*columnas*/
    unicos(R11,R21,R31,R41),
    unicos(R12,R22,R32,R42),
    unicos(R13,R23,R33,R43),
    unicos(R14,R24,R34,R44),
    /*cuadros*/
    unicos(R11,R12,R21,R22),
    unicos(R13,R14,R23,R24),
    unicos(R31,R32,R41,R42),
    unicos(R33,R34,R43,R44).
```

Soluciones:

R11 = R24, **R24** = R33, **R33** = R42, **R42** = 2,
R12 = R23, **R23** = R34, **R34** = R41, **R41** = 3,
R13 = R22, **R22** = R31, **R31** = R44, **R44** = 1,
R14 = R21, **R21** = R32, **R32** = R43, **R43** = 4
R11 = R23, **R23** = R34, **R34** = R42, **R42** = 3,
R12 = R24, **R24** = R33, **R33** = R41, **R41** = 2,
R13 = R22, **R22** = R31, **R31** = R44, **R44** = 1,
R14 = R21, **R21** = R32, **R32** = R43, **R43** = 4

R11	R12	R13	R14	R21	R22	R23	R24	R31	R32	R33	R34	R41	R42	R43	R44	
2	3	1	4	4	1	3	2	1	4	2	3	3	2	4	1	*1*
3	2	1	4	4	1	3	2	1	4	2	3	2	3	4	1	*2*

6. crear un programa para viajeros y conexiones:

%conexiones disponibles en cierta agencia de viajes

```
conecta(medellin, cartagena, avion, 200000).
conecta(cartagena, islas_del_rosario, lancha, 100000).
conecta(medellin, bogota, avion, 150000).
conecta(bogota, cartagena, avion, 100000).
conecta(cartagena, santa_marta, bus, 80000).
conecta(santa_marta, cali, didi, 100000).
conecta(cali, popayan, bus, 25000).
conecta(popayan, guarne, didi, 400000).
conecta(bogota, berlin, avion, 1000000).
conecta(guarne, popayan, bus, 15000).
conecta(popayan, bogota, bus, 20000).
conecta(medellin,bogota, avion, 100000).

%regla para encontrar un camino entre dos ciudades

enrutar(Origen, Destino, Camino, Precio_por_trayecto, Precio_total, Tipos_transporte) :-
   enrutar_aux(Origen, Destino, [Origen], Camino, [], Precio_por_trayecto, 0, Precio_total,
[], Tipos_transporte).

enrutar_aux(Destino, Destino, Camino, Camino, Precios, Precios, Precio_total,
Precio_total, Tipos_transporte, Tipos_transporte).
enrutar_aux(Origen, Destino, Visitados, Camino, Precios_ant, Precios, Precio_total_ant,
Precio_total, Tipos_transporte_ant, Tipos_transporte) :-
   conecta(Origen, Siguiente, Transporte, Precio),
   no_miembro(Siguiente, Visitados),
   unir(Visitados, [Siguiente], Nuevos_visitados),
   unir(Precios_ant, [Precio], Nuevos_precios),
   Precio_total_nuevo is Precio_total_ant + Precio,
   unir(Tipos_transporte_ant, [Transporte], Nuevos_tipos_transporte),
   enrutar_aux(Siguiente, Destino, Nuevos_visitados, Camino, Nuevos_precios, Precios,
Precio_total_nuevo, Precio_total, Nuevos_tipos_transporte, Tipos_transporte).

%regla para unir dos listas
unir([], L, L).
unir([X|L1], L2, [X|L3]) :- unir(L1, L2, L3).

%Simular el member recursivamente sin usar member porque es azucar
no_miembro(_, []).
no_miembro(X, [Y|L]) :- dif(X,Y), no_miembro(X,L).
```

enrutar(medellin,islas_del_rosario, X,Y,Z,M)

M = [avion, lancha],
X = [medellin, cartagena, islas_del_rosario],
Y = [200000, 100000],
Z = 300000

enrutar(guarne, berlin, X,Y,Z,M)

M = [bus, bus, avion],
X = [guarne, popayan, bogota, berlin],
Y = [15000, 20000, 1000000],
Z = 1035000