Ejercicio 1 (2 punto) *Se define en Haskell por recursión la función*

```
f []     = 0
f (x:xs) = x + f xs
```

1. *Determinar el tipo inferido por Haskell para la función* `f`.
2. *Escribir una definición* no recursiva *de la función* `f`.

Solución:

1. El tipo de `f` es

   ```
   f :: (Num a) => [a] -> a
   ```

2. Una definición no recursiva de `f` es

   ```
   f = sum
   ```

   otra definición no recursiva es

   ```
   f = foldl (+) 0
   ```

Ejercicio 2 (2 punto) *¿Qué responde Haskell al evaluar las siguientes expresiones?*

1. `map (/10) [2,4..10]`
2. `until (>1000) (*2) 1`
3. `filter (\x -> 2*x > 7) [1..5]`
4. `[x+y | [x,y] <- [[1,2],[3,4],[5,6]]]`

Solución:

```
map (/10) [2,4..10]                  ==>  [0.2,0.4,0.6,0.8,1.0]
until (>1000) (*2) 1                 ==>  1024
filter (\x -> 2*x > 7) [1..5]        ==>  [4,5]
[x+y | [x,y] <- [[1,2],[3,4],[5,6]]] ==>  [3,7,11]
```

Ejercicio 3 (2 puntos) *Los árboles binarios cuyos nodos son números enteros se pueden representar por el tipo de datos* `Arbol` *definido por*

```
data Arbol = Vacio
           | Nodo Int Arbol Arbol
           deriving (Eq, Show)
```

Por ejemplo, el árbol

```
Nodo 10 Vacio Vacio
Nodo 17 (Nodo 14 Vacio Vacio) (Nodo 9 Vacio Vacio)
```

representan árboles binarios con nodos enteros. Definir la función

```
ocurrencias :: Arbol -> Int -> Int
```

tal que `(ocurrencias a n)` *es el número de veces que aparece* `n` *como un nodo del árbol* `a`*. Por ejemplo,*

```
Main> ocurrencias (Nodo 17 (Nodo 14 Vacio Vacio) (Nodo 17 Vacio Vacio)) 17
2
```

Solución:

```
ocurrencias :: Arbol -> Int -> Int
ocurrencias Vacio          _ = 0
ocurrencias (Nodo n a1 a2) p
    | n == p                 = 1 + ocurrencias a1 p + ocurrencias a2 p
    | otherwise              =     ocurrencias a1 p + ocurrencias a2 p
```

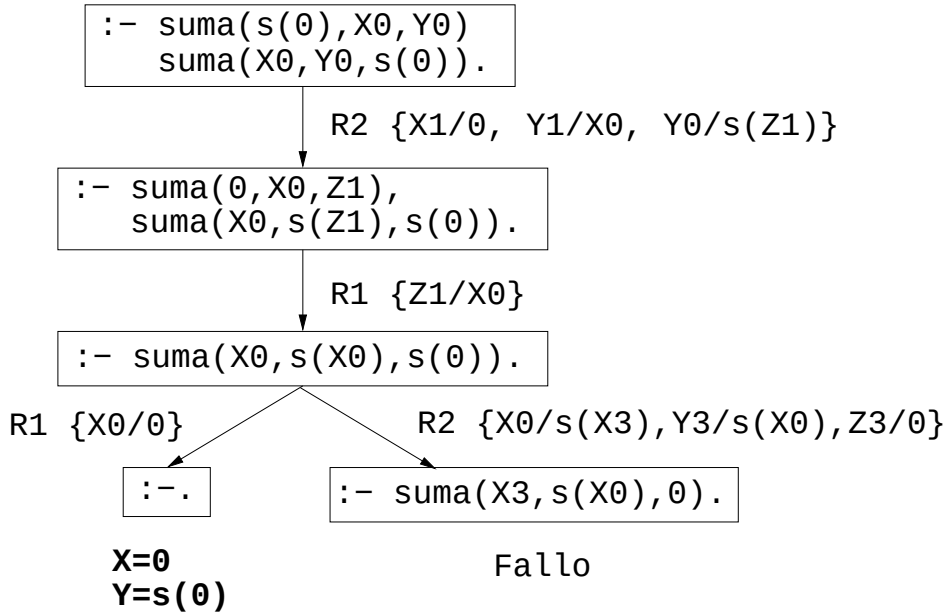
Ejercicio 4 (1 punto) *Se considera el siguiente programa lógico*

```
suma(0,Y,Y).                          % R1
suma(s(X),Y,s(Z)) :- suma(X,Y,Z).     % R2
```

Escribir el árbol de resolución y las respuestas correspondientes al programa y a la pregunta

```
?- suma(s(0),X,Y), suma(X,Y,s(0)).
```

Solución: El árbol de resolución es



La respuesta obtenida es `X=0` e `Y=s(0)`.

Ejercicio 5 (2 puntos) *La relación* `cuenta(+A,+L,-N)` *se verifica si* `N` *es el número de ocurrencias del elemento* `A` *en la lista* `L`*. Por ejemplo,*

```
?- cuenta(a,[a,b,a,a],N).
N = 3.
?- cuenta(a,[a,X,a,a],N).
N = 3.
```

1. *Escribir una definición* recursiva *de la relación* `cuenta`.
2. *Escribir una definición* no recursiva *de la relación* `cuenta`.

Solución: La definición recursiva es

```
cuenta(_,[],0).
cuenta(A,[B|L],N) :-
   A == B, !,
   cuenta(A,L,M),
   N is M+1.
cuenta(A,[B|L],N) :-
   % A \== B,
   cuenta(A,L,N).
```

La definición no recursiva es

```
cuenta(X,L,N) :-
   findall(Y,(member(Y,L),X==Y),L1),
   length(L1,N).
```