Programación Funcional - Práctica 3

Inducción y Recursión - parte 2

1. Decir cuáles de las siguientes expresiones son válidas y dar un tipo válido para ellas.

   - []
   - [3, 1.0, 2]
   - [2 :: Int, 1.0 , 2]
   - [True, [False]]

2. Implementar recursivamente las siguientes funciones.

   - sumaLista :: [Int] -> Int que calcula la suma de todos los elementos de la lista.
   - prodLista :: [Int] -> Int que calcula el producto de todos los elementos de la lista.
   - todosPares :: [Int] -> Bool que devuelve True si y solo si todos los elementos de la lista son pares.
   - todosIguales :: [Int] -> Bool que devuelve True si y solo si todos los elementos de la lista son iguales.
   - pertenece :: Eq a => a -> [a] -> Bool que toma un elemento x y una lista xs y devuelve True si y solo si x pertenece a xs.

3. Probar por inducción estructural que las funciones implementadas en el ejercicio 2 son correctas.

4. Dadas las siguientes definiciones recursivas probar por inducción la sigiuentes propiedades.

```
length [] = 0                    (L1)
length (x:xs) = 1 + length xs    (L2)

[] ++ ys = ys                    (C1)
(x:xs) ++ ys = x: (xs ++ ys)     (C2)

reverse [] = []                  (R1)
reverse (x:xs) = reverse xs ++ [x]  (R2)
```

- P(xs): para toda lista ys y para toda lista zs, (xs ++ ys) ++ zs == xs ++ (yz ++ zs)
- P(xs): para toda lista ys, length (xs ++ ys) == length xs + length ys
- P(xs): length xs == length (reverse xs)