Las definiciones son equivalentes:

```
prop_equivalencia :: [Int] -> Bool
prop_equivalencia xs =
    n_map_1 (*2) xs == map (*2) xs &&
    n_map_2 (*2) xs == map (*2) xs
```

Comprobación

```
Main> quickCheck prop_equivalencia
OK, passed 100 tests.
```

2.9. Filtrado mediante una propiedad

Ejercicio 2.9. *Redefinir la función* `filter` *tal que* `filter p l` *es la lista de los elementos de* `l` *que cumplen la propiedad* `p`*. Por ejemplo,*

```
filter even [1,3,5,4,2,6,1] ⇝ [4,2,6]
filter (>3) [1,3,5,4,2,6,1] ⇝ [5,4,6]
```

Solución: Presentamos distintas definiciones:

1. Definición por recursión:

```
n_filter_1 :: (a -> Bool) -> [a] -> [a]
n_filter_1 p []                = []
n_filter_1 p (x:xs) | p x      = x : n_filter_1 p xs
                    | otherwise = n_filter_1 p xs
```

2. Definición con listas intensionales:

```
n_filter_2 :: (a -> Bool) -> [a] -> [a]
n_filter_2 p xs = [ x | x <- xs, p x ]
```

Las definiciones son equivalentes cuando la propiedad es `even`:

```
prop_equivalencia :: [Int] -> Bool
prop_equivalencia xs =
    n_filter_1 even xs == filter even xs &&
    n_filter_2 even xs == filter even xs
```

Comprobación

```
Main> quickCheck prop_equivalencia
OK, passed 100 tests.
```