Filter

Nos exemplos abaixo temos duas funções semelhantes que possuem a seguinte característica: recebem uma função (Int -> Bool) e uma lista e fazem uma filtragem nessa lista. A função recebida como parâmetro é aplicada em cada elemento da lista. Quando o resultado é False, o elemento é removido da lista.

```
menorque7 x = x < 7

filtrando f b = if taVazia b
                then []
                else if f (head b)
                    then (head b):(filtrando f (tail b))
                    else (filtrando f (tail b))
        where taVazia x = length x == 0


filtrando' :: (Int -> Bool) -> [Int] -> [Int]
filtrando' _ [] = []
filtrando' f (b:bs) | f b = b:(filtrando' f bs)
                    | otherwise = (filtrando' f bs)


λ > filtrando' odd [1..9]
[1,3,5,7,9]
λ > filtrando' menorque7  [1..9]
[1,2,3,4,5,6]
λ > filtrando odd [1..9]
[1,3,5,7,9]
```

E é pra isso que a função filter serve. A diferença é que é uma função mais "poderosa" do que as implementadas acima, já que não funcionam somente para listas de inteiros, mas para listas de qualquer tipo

```
λ > filter odd [1..9]
[1,3,5,7,9]
λ > filter menorque7 [1..9]
[1,2,3,4,5,6]
```