Write the function `oddList'` using a list comprehension.

```haskell
oddList' n =
  if n <= 0
  then []
  else if (n `mod` 2) == 1
    then oddList' (n-1) ++ [n]
    else oddList' (n-1)
```

What are the types of these functions?

1. `addTriplet (x, y, z) = x + y + z`

2. `addTriplet' x y z = x + y + z`

3. `weird a b = [ if x*y > 3 then [a] else [b] | x <- [1..3], y <- [1..3]]`

4. A function `pythagoras` that takes a tuple of integers `(a, b, c)` and returns `True` if and only if \( a^2 + b^2 = c^2 \).
maxInt :: [Integer] -> Integer

maxInt [x] = x

maxInt (x:xs) = max x (maxInt xs)

Use pattern matching to write a function that returns the last element of a list (i.e. the tail function, but without using tail)

Use pattern matching to write a function that returns the second-to-last element of a list