Homework 1

```
-- problem 1
largest:: String->String->String
largest a b
    | length a >= length b = a
    | otherwise = b

-- problem 2
-- Haskell is left associative so reflect num is called recursively without the +1 or -1 computed onto the num parameter. We must add

reflect:: Integer-> Integer
reflect 0 = 0
reflect num
    | num < 0 = (-1) + reflect (num+1)
    | num > 0 = 1 + reflect (num-1)

--- problem 3a
all_factors:: Integer-> [Integer]
all_factors num = [factors | factors <- [1..num], num `mod` factors == 0]

-- problem 3b
perfect_numbers:: [Integer]
perfect_numbers = [nums | nums <- [1..], sum (init (all_factors nums)) == nums]

-- problem 4

-- using if statements

is_odd:: Integer->Bool
is_odd num =
    if num == 0 then False
    else is_even (num-1)

is_even:: Integer->Bool
is_even num =
    if num == 0 then True
    else is_odd (num-1)

-- using guards

is_odd:: Integer->Bool
is_odd num
    | num == 0 = False
    | otherwise = is_even (num-1)

is_even:: Integer->Bool
is_even num
    | num == 0 = True
    | otherwise = is_odd (num-1)

-- using pattern-matching
is_odd:: Integer->Bool
is_odd 0 = False
is_odd num = is_even (num-1)

is_even:: Integer->Bool
is_even 0 = True
is_even num = is_odd (num-1)
```