Homework 1

Tejas Kamtam

305749402

CS 131 - Fall 2023

Haskell Warmup

Problem 1 - Solution

```haskell
-- Problem 1
largest :: String → String → String
largest a b = c where
    c = if (length a) ⩾ (length b) then a else b
```

Problem 2 - Solution

Barry's original code:

```haskell
-- Problem 2
reflect :: Integer → Integer
reflect 0 = 0
reflect num
  | num < 0 = (-1) + reflect num+1
  | num > 0 = 1 + reflect num-1
```

Barry's code is causing a stack overflow because it is actually recursively calling the `reflect` function endlessly during the assignment after the guards.
Barry should've simply written `num` instead of `reflect num` in the assignment.

The fixed version (still including the redundant +/- 1) should look like this instead:

```
-- Problem 2
reflect :: Integer → Integer
reflect 0 = 0
reflect num
    | num < 0 = (-1) + num+1
    | num > 0 = 1 + num-1
```

Problem 3 - Solution

Part a

```
-- a
all_factors :: Int → [Int]
all_factors n = [x | x ← [1..n], n `mod` x == 0]
```

Part b

Note: it is slow but runs

```
-- b
perfect_numbers :: [Int]
perfect_numbers = [x | x ← [1..], sum (all_factors x) - x ==
x]
```

Problem 4 - Solution

Using If Statements

```
-- conditional statements
is_odd :: Int → Bool
is_odd n = if n == 0 then False else is_even (n-1)
```

```haskell
is_even :: Int → Bool
is_even n = if n == 0 then True else is_odd (n-1)
```

Using Guards

```haskell
-- guards
is_odd :: Int → Bool
is_odd n
    | n == 0 = False
    | n /= 0 = is_even (n-1)

is_even :: Int → Bool
is_even n
    | n == 0 = True
    | n /= 0 = is_odd (n-1)
```

Using Pattern Matching

```haskell
-- pattern matching
is_odd :: Int → Bool
is_odd 0 = False
is_odd n = is_even (n-1)

is_even :: Int → Bool
is_even 0 = True
is_even n = is_odd (n-1)
```