Assignment 1

Ethan Poole
LING 185A: Comp. Ling. I
Due: 6 April 2020

Instructions: Please submit your assignment as two separate files. The first file should be typed-up answers to the first question. The second file should be a modified version of `Assignment01.hs`, making sure not to change the file name.

1

26 points

For this question, assume that the type `Shape` and the following names have been defined via the following code in a Haskell file that we have loaded:

(1)
```
data Shape = Rock | Paper | Scissors deriving Show

n = 1
f = \s -> case s of Rock -> 334
                    Paper -> 138
                    Scissors -> 99
g = \z -> z + 4
whatItBeats = \s -> case s of Rock -> Scissors
                              Paper -> Rock
                              Scissors -> Paper
```

Please show how the evaluation of each Haskell expression in (2) proceeds step-by-step, i.e. showing each intermediate expression.

(2) a. `let x = 4 + 5 in (3 * x)`

b. `(\x -> 3 * x) (4 + 5)`

c. `((\x -> (\y -> x + (3 * y))) 4) 1`

d. `let x = 4 in (let y = 1 + x in (x + (3 * y)))`

e. `((\x -> (\y -> y + (3 * y))) 4) 1`

f. `(\y -> y + ((\y -> 3 * y) 4)) 5`

g. `(\y -> ((\y -> 3 * y) 4) + y) 5`

h. `(\x -> x * (let x = 3 * 2 in (x + 7)) + x) 4`

i. `g ((let x = 4 in (\y -> x + y)) 2)`

j. `let x = 5 in (\z -> x * z)`

k. `(\x -> (\z -> x * z)) 5`

l. `f ((\fn -> fn Rock) (\x -> whatItBeats x))`

m. `((\f -> (\x -> f (f x))) whatItBeats) Paper`

For the purposes of this exercise, assume that each of the following constitute one step: a **`let`** reduction step, a lambda reduction step, a **`case`** reduction step, a substitution using a definition from our loaded file, and a single arithmetic addition or multiplication operation.

Here is an example of what your answers should look like:

(3)	`(\x -> (3 + x) * n) 2`	
$\Longrightarrow$	`(3 + 2) * n`	lambda reduction
$\Longrightarrow$	`5 * n`	arithmetic
$\Longrightarrow$	`5 * 1`	substitution from file
$\Longrightarrow$	`5`	arithmetic

You may use `ghci` to check the final result of most (but not all!) of them, but you will be graded on whether you give all of the intermediate steps correct as well.

In many cases, there are multiple routes to the final result, depending on which part of the expression you simplify first. You will get the same result no matter which route you take, but some routes involve more work than others.

$\boxed{\quad\quad\quad\quad 2}$ *14 points*

In lecture, we discussed representing propositional logic in Haskell using the recursive type in (4), and then how to write recursive functions operating over that type. Please write in `Assignment01.hs` a Haskell function named `denotation` (from type `WFF` onto type `Bool`) that implements the written-out function $[\![.]\!]$ in (5).

(4)

```
data WFF = T | F | Neg WFF
         | Conj WFF WFF | Disj WFF WFF deriving Show
```

(5) a. $[\![\mathrm{T}]\!]$ is true

b. $[\![\mathrm{F}]\!]$ is false

c. $[\![\neg\phi]\!]$ is true if $[\![\phi]\!]$ is false; and is false otherwise

d. $[\![\phi \land \psi]\!]$ is true if both $[\![\phi]\!]$ is true and $[\![\psi]\!]$ is true; and is false otherwise

e. $[\![\phi \lor \psi]\!]$ is true if either $[\![\phi]\!]$ is true or $[\![\psi]\!]$ is true; and is false otherwise

In grading your assignment, we will run several test cases against your code, e.g. `denotation $ Disj T F` and `denotation $ Disj (Neg T) F`.