Alicia Yu

Ling 185A: Assignment 1

a. `let x = 4 + 5 in 3 * x`
`let` ⟹ `3 * (4 + 5)`
arithmetic ⟹ `3 * 9`
arithmetic ⟹ `27`

b. `(\x -> 3 * x) (4 + 5)`
lambda ⟹ `3 * (4 + 5)`
arithmetic ⟹ `3 * 9`
arithmetic ⟹ `27`

c. `((\x -> (\y -> x + (3 * y))) 4) 1`
lambda ⟹ `(\y -> 4 + (3 * y)) 1`
lambda ⟹ `4 + (3 * 1)`
arithmetic ⟹ `4 + 3`
arithmetic ⟹ `7`

d. `let x = 4 in (let y = 1 + x in (x + (3 * y)))`
`let` ⟹ `let y = 1 + 4 in (4 + (3 * y))`
arithmetic ⟹ `let y = 5 in (4 + (3 * y))`
`let` ⟹ `4 + (3 * 5)`
arithmetic ⟹ `4 + 15`
arithmetic ⟹ `19`

e. `(\y -> y + ((\y -> 3 * y) 4)) 5`
lambda ⟹ `5 + ((\y -> 3 * y) 4))`
lambda ⟹ `5 + (3 * 4)`
arithmetic ⟹ `5 + 12`
arithmetic ⟹ `17`

f. `(\y -> ((\y -> 3 * y) 4) + y) 5`
lambda ⟹ `((\y -> 3 * y) 4) + 5`
lambda ⟹ `(3 * 4) + 5`
arithmetic ⟹ `12 + 5`
arithmetic ⟹ `17`

g. `(\x -> x * (let x = 3 * 2 in (x + 7)) + x) 4`
lambda ⟹ `4 * (let x = 3 * 2 in (x + 7)) + 4`
`let` ⟹ `4 * ((3 * 2) + 7) + 4`
arithmetic ⟹ `4 * (6 + 7) + 4`
arithmetic ⟹ `(4 * 13) + 4`
arithmetic ⟹ `52 + 4`
arithmetic ⟹ `56`

h. `let k = (\x -> (let y = 3 in x + y)) in k 4`
`let` ==> `(\x -> (let y = 3 in x + y)) 4`
lambda ==> `let y = 3 in 4 + y`
`let` ==> `4 + 3`
arithmetic ==> `7`

i. `let k = (let y = 3 in \x -> x + y) in k 4`
`let` ==> `let k = (\x -> x + 3) in k 4`
`let` ==> `(\x -> x + 3) 4`
lambda ==> `4 + 3`
arithmetic ==> `7`

j. `f ((\k -> k Rock) (\x -> losesTo x))`
lambda ==> `f ((\x -> losesTo x) Rock)`
lambda ==> `f (losesTo Rock)`
def. of `losesTo` ==> `f ((\s -> case s of {Rock -> Scissors; Paper -> Rock; Scissors -> Paper}) Rock)`
lambda ==> `f (case Rock of {Rock -> Scissors; Paper -> Rock; Scissors -> Paper})`
`case` ==> `f Scissors`
def. of `f` ==> `(\s -> case s of {Rock -> 112; Paper -> 71; Scissors -> 304}) Scissors`
lambda ==> `case Scissors of {Rock -> 112; Paper -> 71; Scissors -> 304}`
`case` ==> `304`

k. `((\f -> (\x -> f (f x))) losesTo) Paper`
lambda ==> `(\x -> losesTo (losesTo x)) Paper`
lambda ==> `losesTo (losesTo Paper)`
def. of `losesTo` ==> `losesTo ((\s -> case s of {Rock -> Scissors; Paper -> Rock; Scissors -> Paper}) Paper)`
lambda ==> `losesTo (case Paper of {Rock -> Scissors; Paper -> Rock; Scissors -> Paper})`
`case` ==> `losesTo Rock`
def. of `losesTo` ==> `(\s -> case s of {Rock -> Scissors; Paper -> Rock; Scissors -> Paper}) Rock)`
lambda ==> `case Rock of {Rock -> Scissors; Paper -> Rock; Scissors -> Paper}`
`case` ==> `Scissors`

l. `losesTo (case Paper of {Rock -> Paper; Paper -> Rock; Scissors -> Scissors})`

`case` ==> `losesTo Rock`
def. of `losesTo` ==> `(\s -> case s of {Rock -> Scissors; Paper -> Rock; Scissors -> Paper}) Rock)`
lambda ==> `case Rock of {Rock -> Scissors; Paper -> Rock; Scissors -> Paper}`
`case` ==> `Scissors`

m. `case MyMove (losesTo Paper) of {YourMove v -> n; MyMove x -> (n + f x)}`

def. of `losesTo` ==> `case MyMove (\s -> case s of {Rock -> Scissors; Paper -> Rock;`

```
Scissors -> Paper}) Paper) of {YourMove v -> n; MyMove x -> (n + f x)}
```

lambda ==> `case MyMove (case Paper of {Rock -> Scissors; Paper -> Rock; Scissors -> Paper}) of {YourMove v -> n; MyMove x -> (n + f x)}`

`case` ==> `case MyMove Rock of {YourMove v -> n; MyMove x -> (n + f x)}`
`case` ==> `n + f Rock`
def. of `f` ==> `n + (\s -> case s of {Rock -> 112; Paper -> 71; Scissors -> 304}) Rock`
`case` ==> `n + 112`
def. of `n` ==> `1 + 112`
arithmetic ==> `113`

n. `(case MyMove Rock of {YourMove v -> losesTo; MyMove z -> (\s -> Scissors)}) Paper`

`case` ==> `(\s -> Scissors) Paper`

lambda ==> `Scissors`

o. `let y = 2 in (case MyMove (losesTo Rock) of {YourMove v -> n; MyMove y -> (n + f y)} + y)`

`case` ==> `let y = 2 in ((n + f (losesTo Rock)) + y)`

def. of `losesTo` ==> `let y = 2 in ((n + f (\s -> case s of {Rock -> Scissors; Paper -> Rock; Scissors -> Paper}) Rock)) + y)`

lambda ==> `let y = 2 in ((n + f (case Rock of {Rock -> Scissors; Paper -> Rock; Scissors -> Paper})) + y)`

`case` ==> `let y = 2 in ((n + f (Scissors)) + y)`

def. of `f` ==> `let y = 2 in ((n + ((\s -> case s of {Rock -> 112; Paper -> 71; Scissors -> 304}) Scissors)) + y)`

lambda ==> `let y = 2 in ((n + (case Scissors of {Rock -> 112; Paper -> 71; Scissors -> 304})) + y)`

`case` ==> `let y = 2 in ((n + 304) + y)`

def. of `n` ==> `let y = 2 in ((1 + 304) + y)`

`let` ==> `(1 + 304) + 2`
arithmetic ==> `305 + 2`
arithmetic ==> `307`