Exercises
Chapter 4.2 - The Lambda Calculus

Exercise 1:

Keeping in mind alpha equivalence, choose an answer that is equivalent to the listed lambda term.

1. $\lambda xy.xz$
   (a) $\lambda xz.xz$
   (b) $\lambda mn.mz$
   (c) $\lambda z(\lambda.x.xz)$
2. $\lambda xy.xxy$
   (a) $\lambda mn.mnp$
   (b) $\lambda x.(\lambda y.xy)$
   (c) $\lambda a(\lambda b.aab)$
3. $\lambda xyz.zx$
   (a) $\lambda x.(\lambda y.(\lambda z))$
   (b) $\lambda tos.st$
   (c) $\lambda mnp.mn$

Solution 1:

1. $\lambda xy.xz$
   (b) $\lambda mn.mz$
2. $\lambda xy.xxy$
   (c) $\lambda a(\lambda b.aab)$
3. $\lambda xyz.zx$
   (b) $\lambda tos.st$

Exercise 2:

Which (two or more) of the following are equivalent?

1.
```
mth x y z = x * y * z
```

2.
```
mth x y  = \z -> x * y * z
```

3.
```
mth x   = \y -> \z -> x * y * z
```

4.
```
mth    = \x -> \y -> \z -> x * y * z
```

Solution 2:

All are equivalent to each other

Exercise 3:

The type of **mth** (above) is

```
mth :: Num a=> a-> a-> a-> a
```

Write down the type of

```
mth 3
```

Solution 3:

```
mth 3 :: Num a => a-> a-> a
```

Exercise 4:

Rewrite, using Haskell and evaluate the following:

1. $(\lambda x.x)2$
2. $(\lambda x.(x * 2))4$
3. $(\lambda x.(\lambda y.x * y))3\ 4$
4. $(\lambda x.\lambda y.(if\ x < y\ then\ -1\ else\ if\ x\ ==\ y\ then\ 0\ else\ 1))\ 3\ 4$
   (***Note:*** Use of if inside the lambda expression.)

Solution 4:

1. 2
2. 8
3. 12
4. -1

Exercise 5:

Rewrite the *f* function in the *where* clause using anonymous lambda syntax

```
addOneIfOdd n = case odd n of
    True -> f n
    False -> n
    where f n = n + 1
```

Solution 5:

```
addOneIfOdd n = case odd n of
    True -> (\x->x+1) n
    False -> n
```

Exercise 6:

Rewrite the following to use anonymous lambda syntax

```
  addFive x y = (if x > y then x else y) + 5
```

Solution 6:

```
   ( \x y-> if x > y then x+5 else y+5) 3 4 --applying it to 3 4
```

Exercise 7:

Write a lambda version of the following functions:

1. ***abs:*** which takes an Integer and returns the non-negative value.
   e.g. abs -1 = 1, abs 4 = 4.
2. ***mymax:*** which takes two numbers and returns the larger of the two
3. ***mymin:*** which takes two numbers and returns the smaller of the two

Solution 7:

1. ***abs:***

```
(\x -> if x <0 then (-x) else x) (-4) --applying it to (-4)
```

2. ***mymax:***

```
(\x y -> if x>y then x else y) 14 5 --applying it to the
    arguments 14, 5
```

3. ***mymin:***

```
(\x y -> if x<y then x else y) 14 5 --applying it to the
    arguments 14, 5
```

Exercise 8:

Using the techniques seen in class, encode the following using lambda calculus:

1. AND
2. OR

Solution 8:

1. AND
   $\lambda a.\lambda b.\ a\ b\ FALSE$
2. OR
   $\lambda a.\lambda b.\ a\ TRUE\ b$