CECS 424
Assignment 5
Total: 40 Points

General Instruction

- I recommend that you type your answers to exercise questions by using a word processor (Microsoft word, LibreOffice writer, LaTeX, etc.).
- Submit a `PDF` file (not a zip file) via BeachBoard (Not email or in class).

---

1. (10 points) Evaluate the following λ expressions:
   (a) (2 points) `((λx.λy.(y x) λp.λq.p) λi.i)`
   (b) (2 points) `(((λx.λy.λz.((x y) z) λf.λa.(f a)) λi.i) λj.j)`
   (c) (2 points) `(λh.((λa.λf.(f a) h) h) λf.(f f))`
   (d) (2 points) `((λp.λq.(p q) (λx.x λa.λb.a)) λk.k)`
   (e) (2 points) `(((λf.λg.λx.(f (g x)) λs.(s s)) λa.λb.b) λx.λy.x)`

2. (5 points) Define a function:

   ```
   def make_triplet = ...
   ```

   which is like `make_pair` but constructs a triplet from a sequence of three arguments so that any one of the arguments may be selected by the subsequent application of a triplet to a selector function.
   Define selector functions:

   ```
   def triplet_first = ...
   def triplet_second = ...
   def triplet_third = ...
   ```

   which will select the first, second or third item from a triplet respectively.

   ```
   make_triplet <item1> <item2> <item3> triplet_first => ... => <item1>
   make_triplet <item1> <item2> <item3> triplet_second => ... => <item2>
   make_triplet <item1> <item2> <item3> triplet_third => ... => <item3>
   ```

   for the arbitrary arguments: `<item1> <item2> <item3>`

3. (10 points) Use α conversion to ensure unique names in the expressions in each of the following λ expressions:
   (a) (2 points) `λx.λy.(λx.y λy.x)`

(b) (2 points) `λx.(x (λy.(λx.x y) x))`

(c) (2 points) `λa.(λb.a λb.(λa.a b))`

(d) (2 points) `(λfree.bound λbound.(λfree.free bound))`

(e) (2 points) `λp.λq.(λr.(p (λq.(λp.(r q)))) (q p))`

4. (5 points) The boolean operation `implication` is defined by the following truth table:

X	Y	X IMPLIES Y
F	F	T
F	T	T
T	F	F
T	T	T

Define a λ calculus representation for `implication`:

```
def implies = λx.λy...
```

5. (5 points) The boolean operation `equivalence` is defined by the following truth table:

X	Y	X EQUIV Y
F	F	T
F	T	F
T	F	F
T	T	T

Define a λ calculus representation for `equivalence`:

```
def equiv = λx.λy...
```

6. (5 points) Write a function that finds the product of the numbers between `n` and `one`:

```
def prod1 f n = ...
def prod = recursive prod1
```

so that:

```
prod n
```

in λ calculus is equivalent to:

```
n * n-1 * n-2 * ... * 1
```

in normal arithmetic.