CSE505 – Spring 2018

Assignment 3 – Lambda Calculus
Due Date: Weds, April 11 (11:59 pm)

You may work in pairs for this assignment.

Problem 1: Consider an ML binary tree defined with two constructors, node and leaf, as follows:

datatype tree = leaf **of** int | node **of** tree * tree;

(a) Develop a representation for the above ML binary trees in lambda-calculus following the technique outlined in Lecture 14, slides 22-24. Show the representation for the following ML binary tree in the lambda-calculus:

```
node(node(node(leaf(1),
               leaf(2)),
          leaf(3)),
     node(leaf(4),
          leaf(5)))
```

Write your answer in the file defs.txt (posted in the LAMBDA directory on Piazza) by adding a ‘let’ definition of the form:

let tree1 = _______________

(b) Define a function in lambda-calculus that counts the number of leaf nodes in the tree. Write your answer in the file defs.txt by adding a ‘let’ definition of the form:

let leaves = ______________

Test your answer by deriving the normal form of (leaves tree1).

(c) Define a function in lambda-calculus that sums up the numbers in the leaf nodes in the tree. Write your answer in the file defs.txt by adding a ‘let’ definition of the form:

let treesum = ______________

Test your answer by deriving the normal form of (treesum tree1).

Note: Simple definitions can be given for (b) and (c) without recursion.

Problem 2: Define in lambda-calculus an equality testing function, eq, for two numbers represented as Church numerals. Following Lecture 14, slides 29-30, first write a recursive definition for the equality operation and then abstract the name of the recursive function to obtain a function, Eq, defined as:

let Eq = Lf.______________f____________

Finally, the desired equality function is obtained used the fixed-point finding function, Y, as follows:

let eq = (Y Eq)

Write the definitions for Eq and eq in the file defs.txt and test your answer deriving the normal form of expressions ((eq 0) 0), ((eq 1) 2), etc.

Problem 3: **Lila** and **Lola** are two lambda-calculus simulators with two different reduction strategies: **Lila** always chooses *the leftmost-innermost* redex when reducing lambda-terms, but **Lola** always chooses the *leftmost-outermost* redex when reducing lambda-terms.

For each of the following statements, indicate whether it is TRUE or FALSE, giving an example or counter-example where possible.

i. If **Lila** is non-terminating on some term, then so also is **Lola**.
ii. If **Lola** is non-terminating on some term, then so also is **Lila**.
iii. **Lila** will always derive the normal form of a term (when it exists) in fewer steps than **Lola**.
iv. **Lola** will always derive the normal form of a term (when it exists) in fewer steps than **Lila**.

Write your answer in a file called lilalola.pdf.

WHAT TO SUBMIT: Make a directory called A3_UBITId if working solo or A3_UBITId1_UBITId2 if working as a pair (give UBITId's in alphabetic order). Put defs.txt and lilalola.pdf in the directory, compress, and submit using the submit_cse505 command.

End of Assignment 3