CS314 Spring 2014

Assignment 7

Due Tuesday, April 15, **before** class

Problem 1 – Scheme Programming

1. As we discussed in class (lecture 19), **let** and **let*** do not add anything to the expressiveness of the language, i.e., they are only a convenient shorthand. For instance, `(let ((x v1) (y v2)) e)` can be rewritten as `((lambda (x y) e) v1 v2)`.

   How can you rewrite `(let* ((x v1) (y v2) (z v3)) e)` in terms of λ-abstractions and function applications?

2. Use the `map` and `reduce` functions of lecture 19 (page 5 and page 7) to implement function `maxSquareVal` that determines the maximal square value of a list of integer numbers. Example

```
(define maxSquareVal
  (lambda (l)
    ... ))
...
(maxSquareVal '(-5 3 -7 -10 11 8 7)) --> 121
```

Problem 2 – Lambda Calculus

Use β -reductions to compute the final answer for the following λterms. Note: Use a "fake" reduction step for the "+" operator. Identify each *redex* for β -reduction steps. Your computation ends with a final result if no more reductions can be applied. Does the order in which you apply the β -reduction make a different whether you can compute a final result? Justify your answer.

1. $(((\lambda x.x)\ (\lambda x.1))\ (\lambda y.y))$
2. $((\lambda x.((\lambda z.((\lambda x.(z\ x))\ 3))\ (\lambda y.(+\ x\ y))))\ 1)$
3. $((\lambda z.\ ((\lambda y.z)\ ((\lambda x.(x\ x))(\lambda x.(x\ x)))))\ 5)$

Problem 3 – Programming in Lambda Calculus

Lecture 20, page 8 discusses possible representations of truth values true and false in the lambda calculus, together with lambda calculus implementation of logical operators.

1. Compute the value of ((or true) false) using β -reductions.
2. Define the and operator in lambda calculus. Prove that your definition is correct, i.e., your lambda term for and implements the logical and operation.
3. Define the exor (exclusive or) operator in lambda calculus. Prove that your definition is correct, i.e., your lambda term for exor "implements" the logical exor operation.

Problem 4 – Lambda Calculus and Combinators S & K

In lecture 20, page 16, we introduced the S and K combinators:

- $\mathrm{K} \equiv \lambda \mathrm{xy.x}$
- $\mathrm{S} \equiv \lambda \mathrm{xyz.((xz)(yz))}$

Prove that the identify function $\mathrm{I} \equiv \lambda \mathrm{x.x}$ is equivalent to ((S K) K), i.e.,

$\mathrm{I} \equiv \mathrm{SKK}$