# Homework 3 — assigned Thursday 19 October, due Tuesday 7 November

## General instructions

Total number of points available: 150 pts.

Prepare a single file (preferably PDF) containing the entire home work submission, with all code clearly labeled and presented in a logical order, with all test logs and any other information provided in additional files as appropriate. **Make sure that the first page of the PDF file contains your name.**

If you wish, you may choose to prepare your submisstion as a literate Haskell script, using the `lhs2TeX` tool to prepare typeset Haskell output suitable for processing using LaTeX. This would allow you to use the literate programming facilities in `lhs2TeX` to present commentary on your code, your design decisions, your test results, etc. You must provide adequate comments and you must test the code. The course UNM Learn page hosts a guide to using `lhs2TeX`. For non-programming questions, typeset the answers in the PDF file with everything else. Use 11 pt font for main text; Times and Palatino font families are preferred.

If you don't wish to use `lhs2TeX`, prepare your Haskell script with relevant comments to separate the code into questions as appropriate. For non-programming questions, you can just write your answer inside a Haskell multi-line comment. You can either include your test output as a separate file or paste it into comments in your Haskell source file. Converting your main Haskell source file to PDF is preferable for submission.

Your final PDF file should be named `homework3.pdf`, and you should also include any Haskell source files and any other relevant files, but those are for reference only—only the PDF file will be marked.

## 3.1 An interpreter for the untyped lambda-calculus (150 pts)

This exercise concerns the untyped, call-by-value lambda calculus from Chapter 5 of our textbook, *Types and Programming Languages* by Benjamin Pierce. **You may base your answer on the code discussed in our lecture on Metaprogramming in Haskell.** You may also consult the OCaml implementations provided by the author (mentioned in the footnotes at the beginning of each chapter and online at `http://www.cis.upenn.edu/~bcpierce/tapl/`), and the Haskell TAPL project, `http://code.google.com/p/tapl-haskell/`. **However, you must write your implementation from scratch.**

In Haskell, **implement an evaluator based on the small-step evaluation relation for the untyped, call-by-value lambda calculus.** The abstract syntax and evaluation rules for this language are given in Appendix A at the end of this assignment, for reference.

To complete this assignment you will need to complete the following tasks.

(a) (30 pts) Define a Haskell type `Term` to represent the terms of the untyped lambda calculus (see Appendix A).

*Hint: in the cases involving variables and abstractions, you will need some way to store the name of the variable.*

(b) (40 pts) Define a Haskell function `subst`, such that `subst x t1 t2` implements the capture-avoiding substitution operation $[x \mapsto t_1]t_2$.

(c) (10 pts) Define a Haskell function `isValue`, such that `isValue t` returns `True` if and only if the lambda-term represented by `t` (which should have type `Term`) is a **value**.

*Hint: see Appendix A for the definition of values in the untyped lambda-calculus.*

(d) (40 pts) Using your `subst` and `isValue` functions, define a Haskell function `eval1` that implements **single-step reduction** of lambda calculus terms (encoded as values of type `Term`), according to the evaluation rules presented in Appendix A.

*Hint: you will need some way to indicate if the term cannot be reduced, e.g., using a `Maybe` type.*

(e) (10 pts) Define a Haskell function `eval` that recursively calls `eval1` to evaluate a lambda calculus term (encoded as a value of type `Term`) **as many times as possible**, according to the evaluation rules presented in Appendix A.

(f) (20 pts) Demonstrate the use of your `eval` function to reduce the following untyped lambda-term to its normal form:

$$(\lambda x.\, x\; x)\; (\lambda y.\, y)$$

## How to turn in

All required homework files are to be submitted via the homework submission procedure on the UNM Learn page for this course.

## A Reference: untyped call-by-value lambda-calculus

**Syntax**

| | | | |
|---|---|---|---|
| Terms, $t$ | $::=$ | $x$ | *variable* |
| | $\mid$ | $\lambda x.\, t$ | *abstraction* |
| | $\mid$ | $t\ t$ | *application* |
| Values, $v$ | $::=$ | $\lambda x.\, t$ | *abstraction value* |

**Evaluation Rules**

$$\frac{t_1 \longrightarrow t_1'}{t_1\ t_2 \longrightarrow t_1'\ t_2}\ \text{(E-App1)}$$

$$\frac{t_2 \longrightarrow t_2'}{v_1\ t_2 \longrightarrow v_1\ t_2'}\ \text{(E-App2)}$$

$$\frac{}{(\lambda x.\, t_{12})\ v_2 \longrightarrow [x \mapsto v_2]t_{12}}\ \text{(E-AppAbs)}$$