CSE505 – Fall 2012

Assignment 2

Advanced Control and Lambda Calculus

Assigned Mon, Sep 24

Due Weds, Oct 8 (5pm)

1 [35%] Consider a binary tree defined as:

class Tree {int val; Tree left; Tree right;}

Define an **iterator** bf_elements(t) that yields the values of a Tree t in **breadth-first order**.

2 [15%] Just as iterator constructs can be compiled using procedure parameters, one might wonder whether **coroutine** constructs could also be translated in a similar manner. Explain briefly why such a translation is not feasible, by highlighting what aspect of the use of coroutines would pose the greatest difficulty for translation.

3 [50%] Assuming that a **stack** of n elements e_1 e_2 … e_n is represented by the following lambda-term, where e_1 is at the top of the stack and e_n is at the bottom of the stack:

$$\lambda f.\ \lambda x.((f\ e_1)\ ((f\ e_2)\ \ldots((f\ e_n)\ x)\ \ldots)).$$

Show *non-recursive* lambda-calculus definitions for the following operations on a stack. Assume that the empty stack is represented as: $\lambda f.\ \lambda x.x$

(i) **(top stk)**: return the top element of the stack **stk**;

(ii) **(nonempty stk)**: return a boolean indicating whether **stk** is not empty;

(iii**)** **(size stk)**: return a Church numeral indicating the number of elements in **stk**;

(iv) **((push e) stk)**: given an element **e** and a stack **stk**, return a new stack by placing element **e** on top of the stack **stk**.

Test your answers using the Lambda Calculus simulator located at:

http://www.cse.buffalo.edu/LRG/CSE505/Lambda

Start with the 'readme' file in that directory.

4 [10% extra credit, only if question 3 is fully correct] Explain why defining **(pop stk)** is not as simple as defining **push** or **top**. Which arithmetic operation would **pop** most closely parallel? Develop a non-recursive lambda-calculus definition for **pop** along the same lines.

End of Assignment 2