

Spring 2009, CS255/455 Homework 2  
(Due: 6:00pm April 24th. Drop to Bin Bao's mailbox or his office)

1. Lambda calculus (70 points)

Draw the syntax tree for the following lambda expression. For call-by-value and call-by-name semantics, give the complete content of the environment at the time when sub-expressions  $(x\ a)$  and  $(+ a\ a)$  are evaluated. If an environment has more than one binding, indicate the order in which they are searched for a particular name. Give the result of the evaluation.

```
((λ (a)
  ((λ (x)
    ((λ (a) (x a)) (+ 2 1)))
   (λ(y) (+ a a))))
 (+ 1 1))
```

Answer continues:

2. Recursion construction using lambda calculus (30 points)(CSC455 only)

By recursion we mean a function is defined in terms of itself. Unfortunately lambda calculus does not allow this. However, a function can call some special function and then regenerate itself. By this means, we can achieve recursion using lambda calculus. An instance of this powerful function is defined as

$$Y := (\lambda (y) ((\lambda (x) (y x x)) (\lambda (x) (y x x))))$$

It's easy to verify that for any function  $F$ ,  $YF = F(YF)$ . Now Suppose we want to define a recursive function, something like

$$F(n) = 1, \text{ if } n == 0; \text{ else } n * F(n - 1)$$

Since we can't use  $F$  in defining  $F$ , we use  $f$  instead as a place-holder argument for the function to be passed to itself. Therefore the lambda calculus representation of the function is

$$F := (\lambda (f n) \\ \text{ (if (zero n) } \\ \text{ 1} \\ \text{ (* n (f (- n 1))))))$$

Apply  $Y$  to  $F$ , then we get  $(YF)n = F(YF)n = F(F(YF))n = \dots$  By expanding  $YF$  to  $F(YF)$  level by level, the function gets evaluated in a recursive way.

Now you need to define a recursive function that counts the number of elements from a list. You may use if conditional test as the following syntax

$$\text{(if } < \text{test\_expr} > < \text{true\_expr} > < \text{false\_expr} > \text{)}$$

which means if  $\text{test\_expr}$  is evaluated true then evaluate  $\text{true\_expr}$ , otherwise evaluate  $\text{false\_expr}$ . You may also use  $\text{empty}$  as a predicate testing whether a list is empty or not, and  $\text{rest}$  returning the list with the first element popped off.

To answer this question, you need to put down your  $F$  function, then show how  $(YF)(\text{list})$  expands.

Answer continues:

3. (Extra Credit for both CSC255 and CSC455, 20 points)

Complete the call-by-value interpreter in Scheme. The partial interpreter can be downloaded at [http://www.cs.rochester.edu/~cding/ur\\_only/classes/255S09/66pctEval.scm](http://www.cs.rochester.edu/~cding/ur_only/classes/255S09/66pctEval.scm). The complete interpreter should generate correct result for the test expression defined at the last line of the downloaded file. You are encouraged to craft your program and make it as short and clear as possible. Send a copy of the answer as an email to [cding@cs.rochester.edu](mailto:cding@cs.rochester.edu).