University of Toronto

CSC 324F Programming Languages **Fall 1996**

Homework 2: **Pure and Applied λ-Calculus**

1. Convert the following into β-normal form:
   (a) $(\lambda x.\lambda y.yx)(\lambda x.xx)a$
   (b) $(\lambda y.yx)(\lambda x.xa)a$
   (c) $(\lambda y.yx)(\lambda x.\lambda a.xa)a$
   (d) $\lambda a.(\lambda z.\lambda y.z.(yyz))(\lambda x.\lambda y.xy(yx)a)$

2. (Recursive functions.) Using the applied λ-calculus of Section 14.4 of Sethi, define the following functions:
   (a) Term `power` a b which returns a^b. You may assume that $b \geq 0$. (Hint: use function `times`.)
   (b) Function `fib` n to compute the n'th element of the Fibonacci sequence. For example, `fib` $2 = 2$, `fib` $4 = 5$.

3. (Lists.) A list is an ordered sequence of elements. Notation [a b] stands for a list of 2 elements, a and b. In this problem, you are asked to implement a couple of functions for lists using λ-calculus. [] indicates a list with no elements in it.

   Add the following constants to our λ-calculus: **cons**, **first**, **rest**, **isnil**. **first** is a term that returns the first element of the list. For example, **first** [a b] = a. **rest** returns the list without the first element. For example, **rest** [a b] = [b]; **rest** [a] = []. **cons** takes an element and a list and inserts the element in the beginning of the list. For example, **cons** a [b] = [a b]. Lists can be nested in each other, i.e. an element of a list can be a list. For example, **cons** [a] [b c] = [[a] b c]. Here, the first element is [a], a list.

   The following are some of the δ-rules for these constants:

rest[]	$\stackrel{\delta}{\Rightarrow}$	[]
isnil[]	$\stackrel{\delta}{\Rightarrow}$	**true**
isnil(**cons** M N)	$\stackrel{\delta}{\Rightarrow}$	**false**
cons(**first** M)(**rest** M)	$\stackrel{\delta}{\Rightarrow}$	M
first(**cons** M N)	$\stackrel{\delta}{\Rightarrow}$	M
rest(**cons** M N)	$\stackrel{\delta}{\Rightarrow}$	N

   (a) Write a recursive function `toa` which goes through the list of elements, changing each to a. `toa`[] should return []. `toa`**cons** M N should change M to a and then recursively go through elements of N, changing each to a (you can do this by **cons** a N). `toa` [a b c [d e]] = [a a a a].

(b) Modify `toa` to replace all elements of the list (sublists included) by a. To do this, we will need to define one more constant, **islist**. It returns **true** if its argument is a list and **false** otherwise.

$$\mathbf{islist}(\mathbf{cons}\ \mathrm{M}\ \mathrm{N}) \stackrel{\delta}{\Rightarrow} \mathbf{true}$$
$$\mathbf{islist}[] \stackrel{\delta}{\Rightarrow} \mathbf{false}$$

So, **islist** a = **false** whereas **islist**[a] = **true**. The new function is called `toarec`. To solve this problem, we will need the following definition: `toarec`(**cons** M N) - if M is a list, we have to process M recursively. Else just replace M by a. So, `toarec` [a b c [d e]] = [a a a [a a]].