MID-SEMESTER EXAMINATION

Design and Implementation of Functional Programming Languages

DATE: 30th September 2003 **Maximum Marks: 120**

1. Consider the usual definition of ++:

```
(++) :: [a] -> [a] -> [a]
[ ]    ++ ys = ys
(x:xs) ++ ys = x : (xs ++ ys)
```

   (a) Assuming that as_1 and as_2 are already evaluated, what is complexity of evaluating as_1 ++ as_2 in terms of the lengths of as_1 and as_2.

   (b) Assume that there are a number of *evaluated* lists as_i, each of length n, and different instances of the following information presented in a tabular form.

      i. An *unevaluated* list valued expression e made of these lists

      ii. A position *pos* in e

   For each instance, write the complexity ($O(1)$, $O(n)$ etc.) of selecting the pos^{th} element from e, if the evaluation strategy is lazy. (25 Marks)

e	*pos*	*Complexity*
as_1 ++ as_2	first	
as_1 ++ as_2	last	
((..((as_1 ++ as_2) ++ as_3) .. ++ as_k)	first	
((..((as_1 ++ as_2) ++ as_3) .. ++ as_k)	last	
as_1 ++ (as_2 ++ (as_3 .. ++ as_k)..)	first	
as_1 ++ (as_2 ++ (as_3 .. ++ as_k)..)	last	

2. Fill in the blanks: (20 Marks)

   (a)
```
filter p = concat . map _____________
```

   (b)
```
zip ([ ], [ ]) = [ ]
zip ((x:xs), (y:ys)) = (x,y) : zip (xs,ys)

cross (f,g) (x,y) = (f x, g y)

map (cross(f,g)) . zip = zip . _______________
```

3. (a) Show that Y_{funny} defined below is a fixed point yielding function.

$T = \lambda abcdefghijklmnopqstuvwxyzr.r(thisisafixedpointcombinator)$
$Y_{funny} = TTTTTTTTTTTTTTTTTTTTTTTTTT$

(b) Write a function *max* in lambda calculus which takes two Church numerals and returns the greater numeral. Do not use the Y combinator. (25 Marks)

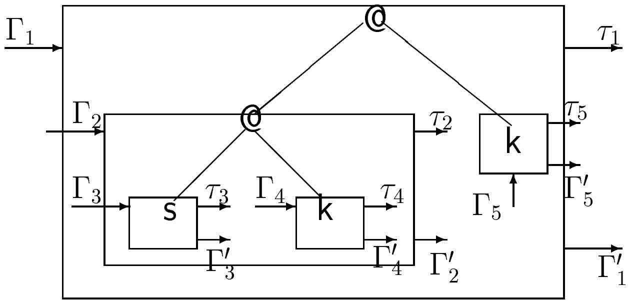
4. What is the type of the function `scan` shown below?

   (a)
```
scan f q xs = q : case xs of
                            [ ] -> [ ]
                            x:xs -> scan f (f q x) xs
```

   (b) Consider inferencing the type of the term shown below using the Hindley-Milner algorithm.

```
let
  s = \x \y \z -> x z (y z)
  k = \x \y -> x
in s k k
```

   Show the type inferencing of the body of the let expression by specifying the type environments Γ_i and Γ'_i, and the types τ_i in the figure below. (30 Marks)



5. For the question below assume that the only numeric type is `Integer`. Find the types of `mylength`, `mylength'`, `a`, `b`, `c`, and `d` in the program shown below. Also rewrite the program to resolve overloading. (20 Marks)

```
mylength l = if l==[ ] then 0
             else 1 + (mylength (tail l))

mylength' [ ] = 0
mylength' (x:xs) = 1 + (mylength' xs)

a = mylength [1,2,3,4,5]
b = mylength [ ]

c = mylength' [1,2,3,4,5]
d = mylength' [ ]
```