Date: 8th August 2003 | QUIZ 1 | Total Marks: 25

1. Evaluate the following Haskell expressions (5 Marks)

   (a) `tail [(True, False)]`

   (b) `head [("1","2"), ('3','4')]`

   (c) `concat [[ ], [True], [ ]]`

   (d) `["a", "b"] ++ [[ ]]`

   (e) `"con" : ["cons"]`

2. What would be the result of evaluating `x` in (3 Marks)

```
x = 0 : y
y = map (1+) x
```

3. Give an example where the application of `foldr` on a list will terminate in spite of the list being infinitely long. (2 Marks)

4. Define a recursive function `leftmost` which takes a non-negative integer `x` and returns the leftmost decimal digit of the given integer. For example, `leftmost 8` should evaluate to 8, and `leftmost 625371` to 6. (5 Marks)

5. Give types of the functions `reptree`, `compose` and `s` [1] defined below: (5 Marks)

   (a) `reptree f p = Node p (map (reptree f) (f p))`, assuming the usual definition of `Tree a`.

   (b) `compose f g x = f (g x)`

   (c) `s f g x = f x (g x)`

6. The function `takeWhile` is defined as: (5 Marks)

```
takeWhile           :: (a -> Bool) -> [a] -> [a]

takeWhile p [ ]       = [ ]
takeWhile p (x:xs) = if (p x) then  x : takeWhile p xs else [ ]
```

   Define `takeWhile` in terms of `foldr`. That is, give definitions of `f?` and `id?` such that

```
takewhile p xs  = foldr f? id? xs
```

[1] The function `s` along with the function `k` defined as `k x y = x` is enough for computing. It can be shown that *any* computable function can be written in terms of `s` and `k`!