UNIVERSITY OF LONDON
(University College London)

M.Sc. DEGREE 1998

COMPUTER SCIENCE D16: FUNCTIONAL PROGRAMMING

Answer THREE Questions.

The Use of Electronic Calculators: is NOT Permitted.

Answer Question 1 and any TWO other questions.

1. (a) Explain the terms “lazy evaluation” and “list comprehension”. [4]

   (b) What value does the following expression compute?

```
last [(x ^ 2) + 1 | x <- [1..]; (x ^ 2) < 5000000]
where
last []     = error "last of empty list"
last [x]    = x
last (x:xs) = last xs
```
[6]

   (c) What is a type? [2]

   (d) Why will Miranda not allow the following function definition?

```
dumb x = (x x)
```
[6]

   (e) Given the following function definitions, where `number` is the name of a type synonym:

```
one :: number
one f x = f x

two :: number
two f x = f (f x)

three :: number
three f x = f (f (f x))
```

— Give the most general type synonym definition for `number`. [4]

— What is the most general type of the following function?

```
operator a b = h
               where
               h c x = a c (b c x)
```
[6]

— Explain the operation of the function `operator` (in the context of the other definitions given above) by giving the evaluation steps of a simple application. Then explain in general terms what the function `operator` does (for example, could it be given a more descriptive name?). [6]

[Total 34]

[TURN OVER]

2. (a) What are algebraic data types? Give examples of the different kinds of algebraic type and how they might be used. [5]

(b) Define a type structure to represent binary trees in which the nodes of the tree hold number values and the leaves also hold number values. [4]

(c) Define a function to determine the height of a tree represented using your type, where the height of a tree is the number of nodes along the longest branch from the root to a leaf. [9]

(d) Consider the following function defined for lists:

```
> map_on_tails f [] = []
> map_on_tails f xs = (f xs) : (map_on_tails f (tl xs)
```

Define an analogous function on the trees represented by your type, where a function is applied to every sub-tree within a tree. [11]

(e) Define a function which will take a tree and return a tree containing at each node the height of the corresponding sub-tree in the input tree. [4]

[Total 33]

3. (a) Briefly explain, with examples, what is meant by the following terms:

partial application
case analysis
structural induction

[10]

(b) Briefly explain the advantages of using Higher Order Functions. Illustrate your answer by giving the code (and type) for a higher-order function (call it `gsort`) which will sort a list of any type of object in any user-provided ordering.

[10]

(c) Discuss, with examples, the role played by *recursion* in functional programming. Your answer should address (amongst other things) the following points:

— Recursive function definitions.

— Recursive types (both built-in and user-defined).

— Stack, accumulative and mutual recursion. [13]

[Total 33]

[CONTINUED]

4. (a) Provide definitions, including types, for the two functions (from the Miranda Standard Environment) called `foldr` and `foldl`. [12]

(b) What values do the following five expressions compute?

```
foldr (:) [] [1,2,3]

hd (foldr (:) [] [1..])

foldl (:) [] [1,2,3]

foldl (swap (:)) [] [1,2,3]
where
swap f x y = f y x

foldl (swap (:)) [] [1..]
where
swap f x y = f y x
```

[5]

(c) Under what circumstances are the functions `foldr` and `foldl` interchangeable? [8]

(d) What does the following function do and what is its type?

```
f x = foldr rcons id x []
      where
      rcons a f b = f (a:b)
      id x = x
```

Demonstrate how f works by giving the intermediate evaluation steps of f applied to an argument. [8]

[Total 33]

[TURN OVER]

5. (a) Give the syntax definition for a lambda-calculus expression (with constants but without types) and explain the operation of the three primary evaluation (reduction) mechanisms. Explain what a delta-rule is and how it is used. [7]

(b) What is the Y combinator and why is it important? Give a definition for Y. [6]

(c) Provide two evaluations of the following lambda-calculus expression, first using Normal Order evaluation and then using Applicative Order evaluation and comment on the results.

```
(λx.((λx.(+ x 1)) 3)) (/ 2 0)
```
[6]

(d) What are the advantages of using graph reduction instead of string reduction or tree reduction? [4]

(e) Provide a graph-reduction diagrammatic rule for evaluating the graph representing an application of the Y combinator in the most space-efficient manner. [4]

(f) Briefly explain the main principles of a parallel graph reduction machine. [6]

[Total 33]

[END OF PAPER]