UNIVERSITY OF LONDON
(University College London)

B.Sc. DEGREE DEGREE 1991

COMPUTER SCIENCE B330: FUNCTIONAL PROGRAMMING

Answer THREE Questions.

The Use of Electronic Calculators: is NOT Permitted.

1. (a) What is a recursive function? Give an example of a recursive function definition in SML and explain the three features which prevent your function from looping forever.

[5]

(b) Give examples of a stack-recursive function and an accumulative-recursive function. Explain the difference between the two styles.

[4]

(c) What is a tail-recursive function? What is the relationship between tail-recursive, stack-recursive and accumulative-recursive functions?

[4]

(d) What is pattern-matching? What are the constructs which may appear as patterns in an SML function definition? What is a *constructor*? Give an example of how to use a constructor in an SML function definition.

[8]

(e) What is a recursive type? Give an example of a recursive type.

[4]

[Total 25]

[TURN OVER]

2. (a) What is the most general type of each of the following two SML functions?

```
fun   maxval  f  g  x =   let
                            fun max  ((a:int),b)
                                    =  if  (a>b)
                                         then a
                                         else b
                          in
                            max  (f  x,  g  x)
                          end


fun   newf  nil  y  z =   42.0
|     newf  gs   y  z =   if     ((hd  gs)  y)
                          then   (y  z)
                          else   42.0
```

[7]

(b) The functions *reduce* and *accumulate* are defined by:

```
fun   reduce f def nil                 =   def
|     reduce f def (front :: rest)     =   f  front (reduce f def rest)

fun   accumulate f acc nil             =   acc
|     accumulate f acc (front :: rest) =   accumulate  f  (f acc front) rest
```

What is the most general type of each of these two functions? Explain how these two functions can be used, giving an example in each case.

[4]

(c) Under certain circumstances, reduce and accumulate are interchangeable. State the conditions which must in general be satisfied by *f* and *a* to ensure that

accumulate f a xs = reduce f a xs

[6]

[Total 25]

[CONTINUED]

3. SML has many features to help the programmer to produce code which is modular and has a high degree of abstraction. Discuss these features and their benefits.

[Total 25]

4. Give an SML abstract type definition for a SET of restricted-polymorphic items. The SET should be implemented as a list of unordered items with no duplicates. You should provide definitions for the following functions:

nullset	returns an empty SET
addset	adds an item to a SET
subtractset	removes an item from a SET
intersect	returns the intersection of two SETS
union	returns the union of two SETS
showset	returns all the items of a SET, collected into a list

All of the above functions except showset should return a SET. The function *intersect* should use the higher-order function *filter* (as defined below). The function *union* should use the higher-order function *reduce* (as defined below).

Minor syntactic errors will not be penalised. You may assume the following function definitions:

```
fun   member x nil                =   false
|     member x (front :: rest)    =   (x=front) orelse (member x rest)

fun   C f y x                     =   f x y

fun   filter pred nil             =   nil
|     filter pred (front :: rest) =   if    (pred front)
                                      then  (front :: (filter pred rest))
                                      else  (filter pred rest)

fun   reduce ff def nil           =   def
|     reduce ff def (front :: rest) = ff front (reduce ff def rest)
```

[Total 25]

[TURN OVER]

5. (a) Why is garbage collection necessary in a graph reduction system?

[8]

(b) Describe briefly the operation of the following three garbage-collectors and compare their advantages and disadvantages:

mark-scan
“Baker” copying
reference-count

[10]

(c) Explain how a single-bit reference count can be used for garbage-collection. What is the advantage of holding the reference count in the cell pointers instead of in the cells themselves? Why is it necessary to implement a second type of garbage collector when using single-bit reference counts?

[7]

[Total 25]

[END OF PAPER]