UNIVERSITY COLLEGE LONDON

University of London

EXAMINATION FOR INTERNAL STUDENTS

For The Following Qualification:-

M.Eng.

Comp Sci B330: Functional Programming

COURSE CODE	:	**COMPB330**
UNIT VALUE	:	**0.50**
DATE	:	**19-MAY-03**
TIME	:	**14.30**
TIME ALLOWED	:	**2 Hours 30 Minutes**

TURN OVER

Answer any THREE questions.

1. Consider the following λ-calculus expression:

$$(\lambda y.((\lambda f.(\lambda x.(f\ x)))\ (\lambda g.(g\ y)))\ 37)\ (\text{if True } (+\ 1)\ (+\ (37\ /\ 0)))$$

(a) In the sub-expression $\lambda g.(g\ y)$ explain which identifiers are free and which are bound.

[2 marks]

(b) Explain the expected operation of the sub-expression $\lambda g.(g\ y)$ when applied to an argument. Give an example application of this sub-expression (you may assume any value for y - for example, you might assume that y is 37) and show the evaluation steps of your example.

[4 marks]

(c) Give the evaluation steps for the Normal Order Reduction of the above expression. *(Assume that operators + and / may appear in either prefix or infix position)*

[8 marks]

(d) Give the evaluation steps for the above expression using some other reduction order, and use your example to demonstrate how evaluation could produce a different outcome. Is this the only different outcome that can occur? Why?

[10 marks]

(e) The above λ-calculus expression is very slightly altered by changing the bound names as follows:

$$(\lambda a.((\lambda f.(\lambda a.(f\ a)))\ (\lambda f.(f\ a)))\ 37)\ (\text{if True } (+\ 1)\ (+\ (37\ /\ 0)))$$

First give the evaluation steps to perform a simple beta-reduction of the following sub-expression:

$$\lambda f.(\lambda a.(f\ a))\ (\lambda f.(f\ a))$$

In the full expression, replace the above sub-expression with its beta-reduced result. Then give the evaluation steps for any remaining redexes (in Normal Order). When an error occurs, explain what the error is and why it occurred.

Hint: in the above sub-expression, which identifiers (names) are free before the simple beta-reduction, and which (if any) are free after the simple beta-reduction? If a free identifier has become bound, then simple beta-reduction has failed - but what has caused the simple beta-reduction to behave like this?

[9 marks]
[Total 33 marks]

[CONTINUED]

2. (a) Explain what the following Miranda algebraic type represents:

```
fred * ::= Empty | Node * [fred *]
```

[4 marks]

(b) Consider the following Miranda code (the `!` operator provides an index into a list counting from zero, for example `[1,2,3]!0` returns the value `1`):

```
t_graph * ::= Emptygraph | Node * [t_graph *]

glist :: [t_graph char]
glist = [  Node 'A' [glist!2, glist!1],
           Node 'B' [glist!3],
           Node 'C' [glist!0, glist!3],
           Node 'D' []
        ]

graph :: t_graph char
graph = hd glist
```

Give a diagram to illustrate the data structures `glist` and `graph`.

[6 marks]

(c) The predefined Miranda function `member` is defined as:

```
member :: [*] -> * -> bool
member [] x = False
member (x:xs) y = (x=y) || member xs y
```

Using the function `member`, give the definition for a Miranda function *printgraph* which takes an argument of type `t_graph char` and returns a list of characters representing the data structure. The word "empty" should be used to signify an empty graph node. Your program must **not** loop forever; if a Node has been encountered previously, the word "seen" should be printed instead of its list of successor nodes.

For example, the application `(printgraph graph)` should return the following result (NB a node that is shared and appears in different branches may appear multiple times in the output):

```
Node A [ Node C [ Node A seen, Node D empty], Node B [Node D empty] ]
```

[23 marks]

[Total 33 marks]

[TURN OVER]

3. You are given the following Miranda function definitions:

```
nil f = f  (error "head of nil")  (error "tail of nil")  True

cons a b f = f a b False

head x = x h
         where
         h a b c = a

tail x = x t
         where
         t a b c = b

isnil x = x g
          where
          g a b c = c
```

(a) Use hand evaluation to demonstrate the following two equalities:

```
head (cons a b) = a

tail (cons a b) = b
```

[4 marks]

(b) Use hand evaluation to demonstrate the following equality:

```
head (tail (cons a (cons b c))) = b
```

[8 marks]

(c) Explain why the following function definition is wrong:

```
newmap f         nil = nil
newmap f (cons a b) = cons (f a) (newmap f b)
```

[6 marks]

(d) Suggest a correct definition for the function **newmap** that maps a function over the elements of the "cons" as defined in the introduction to this question. Do **not** attempt to give the type for **newmap**.

[15 marks]

[Total 33 marks]

[CONTINUED]

4. (a) Given a heap memory with the following characteristics:

- heap size = 2Mbytes
- block header size = 4 bytes
- minimum allocatable block size = 128 bytes
- maximum allocatable block size = 16Kbytes
- free-list allocator using LIFO first-fit allocation

(b) What is the maximum number of live blocks that can exist in this heap?

[2 marks]

(c) What is the minimum information that needs to be stored in each block header?

[2 marks]

(d) How would the performance of the allocator be affected if address-ordered first-fit allocation were used instead of LIFO first-fit allocation?

[2 marks]

(e) What is the maximum number of live blocks that can exist in this heap if 8-byte alignment of blocks is required?

[2 marks]

(f) What problem is solved by Knuth's boundary tags mechanism? Give a detailed explanation of how boundary tags could be implemented in the above heap to provide a comprehensive solution to the stated problem. Specific attention should be paid to the contents of the block headers, the procedures involved, and the performance implications.

[15 marks]

(g) If the maximum block size were to be increased to 127Kbytes, how would this affect the block header (assume coalescing is supported)? Suggest two solutions and state the factors that would determine which solution would be preferable in a given context.

[10 marks]

[Total 33 marks]

[TURN OVER]

5. (a) Explain how a tree of binary application nodes can be used to represent a functional program, how this representation naturally supports Curried function definitions, and how it can be extended to support recursive functions.

[6 marks]

(b) Illustrate with diagrams how a beta-reduction is performed in a graph reduction system, and use your illustration to demonstrate how garbage may be produced.

[10 marks]

(c) Give a detailed description of the contents of a cell in a graph reduction system that uses binary application cells, and explain how the cell could be augmented to accommodate a reference-counting garbage collector.

[8 marks]

(d) Give three limitations of reference-counting garbage collection, in terms of its functionality and its performance, and suggest possible solutions for two of these limitations.

[9 marks]

[Total 33 marks]

[END OF PAPER]