Department of Computer Science
University College London

Cover Sheet for Examination Paper to be sat in May 2007

COMP3011: Functional Programming

Time allowed 2.5 hours

Calculators are allowed

Answer THREE questions

Checked by First Examiner: Date:

Approved by External Examiner: Date:

Answer any THREE questions

Marks for each part of each question are indicated in square brackets.

Calculators are permitted

1. (a) Give the syntax for the untyped lambda calculus without constants.
[4 marks]

(b) Give the Miranda code for an algebraic type called `exp` that defines the legal values for an expression in the untyped lambda calculus without constants.
[4 marks]

(c) Give the Miranda code for a function called `nsub` that takes three arguments (an expression of type `exp` and two variable names) and returns an expression of type `exp`. The returned expression should be the same as the input expression but with all free occurrences of the first variable name replaced by the second variable name. [10 marks]

(d) Give the Miranda code for a function called `esub` that takes three arguments (an expression of type `exp`, a variable name, and another expression of type `exp`) and returns an expression of type `exp`. The returned result should be the same as the first input expression but with all free occurrences of the variable name replaced by the second input expression. [10 marks]

(e) Give the Miranda code for a function called `beta` that takes an expression of type `exp` as input and returns an expression of type `exp`. The input expression must be an application of a lambda definition to an argument, and the returned expression should be the result of performing a simple beta reduction on that application. Do not attempt to cater for free variable capture.
[5 marks]

[Total 33 Marks]

2. Consider the following Miranda function definitions:

```
rc v g i = g (v:i)

rn x = x

rh g = hd (g [])

f [] y = y
f (x:xs) y = f xs (rc x y)

g [] y = y
g (x:xs) y = g xs (x:y)
```

(a) What are the types of the functions `rc`, `rn`, `rh`, `f` and `g`?

[12 marks]

(b) Evaluate the following expression by hand, giving all the intermediate steps:

```
rh (rc 1 (rc 2 (rc 3 rn)))
```

[9 marks]

(c) What values do the following two expressions return?

```
(f [1,2,3] rn) []

g [1,2,3] []
```

[6 marks]

(d) Explain what the functions `rc`, `rn` and `rh` do, by comparing them with some other common Miranda functions or constructors.

[6 marks]

[Total 33 marks]

3. You are given the following definitions:

```
truth * == (* -> * -> *)

true :: truth *
true  = cancel

false :: truth *
false = (swap cancel)

swap f x y = f y x

cancel x y = x
```

(a) Consider the functions `not`, `and`, and `or`. These are intended to mimic the standard Boolean operators of the same name, but they should work on values of type `truth` instead of values of type `bool`. Give the Miranda types for the functions `not`, `and`, and `or.`

[6 marks]

(b) Provide Miranda definitions for the functions `not,` `and,` and `or,` and demonstrate their correct working with example hand evaluations.

[18 marks]

(c) The Boolean function `xor` (exclusive or) returns true if either of its two arguments are true, but it returns false if either both arguments are false or both arguments are true. Provide a Miranda definition (including the type) for the function `xor`, and demonstrate its correct working with an example hand evaluation. [9 marks]

[Total 33 marks]

4. (a) Given the following Miranda combinator definitions:

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

b f g x = f (g x)

w h x = h x x

e = s (b w b) (b w b)
```

and given that the Y fixpoint combinator is defined axiomatically as:

```
Y f = f (Y f)
```

Show that the combinator `e` is equivalent to `Y` by illustrating that successive transformations of the graph representation of `(e f)`, for any `f`, produces a graph equivalent to `(f (e f))`. At each stage, indicate which of the nodes in the graph are the same as the previous stage; which have been updated; and which are new. [12 marks]

(b) In fact, the combinator `e` given above is not a valid Miranda definition because Miranda cannot infer a valid type for `e`. Explain why Miranda might have difficulty understanding the type of `e`. [7 marks]

(c) Give the Miranda types for the following Miranda function definitions:

```
fun1 (a:b) f x = a ((x f b) . f)

fun2 []     j      acc = acc
fun2 f      []     acc = acc
fun2 (x:xs) (y:ys) acc = y (fun2 xs ys (x y acc))

fun3 [] y z = z
fun3 (x:xs) y z = (x z) & fun3 xs y ((y . x . y) z)
```

[14 marks]

[Total 33 marks]

5. You are told that a novel memory allocator preferentially allocates large blocks at the top end of available memory and small blocks at the bottom end of available memory. In all other respects the allocator is conventional. The allocator uses a single free list in a heap memory of size ***H*** bytes (See Figure 1)

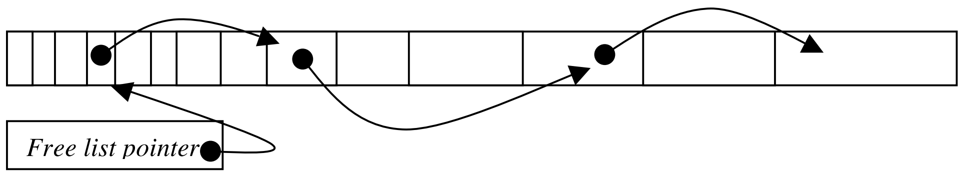

Figure 1

(a) How many blocks of memory (and of what size) are there on the free list when the program starts (before any allocation)? [5 marks]

(b) Give the details of how the very first allocation could be achieved if the requested block size is (i) small, and (ii) large. Give a diagram to illustrate the heap and the free list both before and after allocation in each case. [9 marks]

(c) If the free list is doubly-linked (see Figure 2), and each block's header contains its size, what is the size of a block header and what is the minimum size of an allocatable block? [5 marks]

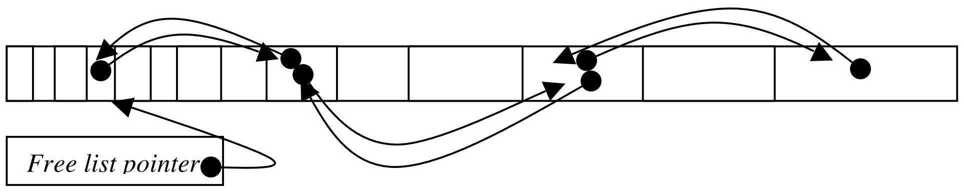

Figure 2

(d) You are told that the allocator maintains two pointers – one to the start of the double-linked free list and one to the end of the double-linked free list. Give an allocation algorithm using these two pointers which would, for multiple successive block allocation requests, place small blocks at the bottom end and large blocks at the top end of available memory. State any assumptions you make. [7 marks]

(e) Give a sequence of allocation requests and requests to free previously-allocated blocks that could defeat the intentions of this allocator, thereby causing small and large blocks to be intermingled. [7 marks]

[Total 33 Marks]

[END OF PAPER]