UNIVERSITY OF LONDON

IMPERIAL COLLEGE OF SCIENCE, TECHNOLOGY AND MEDICINE

EXAMINATIONS 1996

BEng Honours Degree in Computing Part II
MEng Honours Degrees in Computing Part II
BSc Honours Degree in Mathematics and Computer Science Part II
MSci Honours Degree in Mathematics and Computer Science Part II
for Internal Students of the Imperial College of Science, Technology and Medicine

This paper is also taken for the relevant examinations for the
Associateship of the Royal College of Science
Associateship of the City and Guilds of London Institute

PAPER 2.3 / MC2.3

COMPILERS

Tuesday, April 30th 1996, 2.00 - 3.30

Answer THREE questions

For admin. only: paper contains
4 questions
4 pages (excluding cover page)

1 Consider the following fragment of C/C++:

```
struct node {
  struct node *left, *right;
  int val;
} ;

struct node *BuildTree(int N)
{
  struct node *p, *L, *R;

  if (N < 2) {
    return NULL;
  }
  else
  {
    L = BuildTree(N-1);
    R = BuildTree(N-2);
    p = malloc(sizeof(struct node));
    p->left = L;
    p->right = R;
    p->val = N;
    return p;
  }
}
```

a Suppose the following function call is executed:

```
tree = BuildTree(3);
```

Draw a sequence of diagrams showing the state of the stack as it appears each time `BuildTree` is entered. Show all variables, parameters, results, return addresses and links (it is not necessary to show the heap). What does the program do?

b Explain how garbage collection can be achieved using reference counting.

c Suppose that the variables `p` and `q` are pointers to objects of type `node` as above. Explain the reference counting operations involved in the following assignment statement:

```
p = tree->left;
p->right = tree;
```

d What problem arises with reference counting garbage collection in examples like this? Explain.

(The four parts carry, respectively, 50%, 20%, 20% and 10% of the marks).

2 Consider the following grammar for a fragment of the English language:

S	→	NP Verb {PP}
NP	→	Noun {PP}
PP	→	'on' NP
Verb	→	'sleep' | 'appear'
Noun	→	'ideas' | 'boats' | 'Tuesdays' | 'stilts'

(Recall that the notation "{PP}" denotes the repetition of PP zero or more times).

a Check, by drawing the parse tree, that the following two sentences are members of the language generated by the grammar:

```
ideas appear on Tuesdays

ideas sleep on stilts
```

b Modify the grammar to eliminate the "$\{\alpha\}$" notation for repetition, to yield a definition which is suitable for recursive-descent parsing.

c Assume that lexical tokens are represented by the following Miranda data type:

```
token ::= On | Sleep | Appear | Ideas | Boats | Tuesdays | Stilts
```

Using Miranda or another convenient programming language, sketch a recursive descent parser for this language. Your parser need not construct an abstract syntax tree, but it should generate an error message if a syntax error is found.

d Show, by computing appropriate FIRST and FOLLOW sets, that this grammar has a context clash.

e Show that this grammar is ambiguous by giving an example.

(The five parts carry, respectively, 10%, 10%, 35%, 35% and 10% of the marks).

Turn over ...

3 In a compiler for a simple programming language, Boolean expressions are represented using the following abstract syntax:

```
bexp ::= True | False | Or bexp bexp | And bexp bexp |
         Var name | Not bexp
```

For simplicity the language has Boolean variables, but no comparison operations.

a Using Miranda, write a function which generates code which tests the value of a supplied Boolean expression, and branches to one of two labels (supplied as parameters of type `name`) as soon as the expression is known to be `True`, and to the other as soon as it is known to be `False`.

Boolean variables are represented at run time using integers with the value 0 for false and 1 for true. Your machine has registers and instructions `BEQZ` r_n `lab` (branch if register r_n is zero) and `BNEQZ` r_n `lab` (branch if register r_n is non-zero) for testing their values.

You may find it useful to assume the existence of a function `newlabel`, which generates a fresh label not used elsewhere.

b Give an example of a Boolean expression in which short-circuiting results in reduced execution time compared with the more straightforward implementation using machine instructions `AND` r_n r_m, `OR` r_n r_m and `NOT` r_n with the usual meanings. Show that under certain circumstances, the short-circuiting version will execute more slowly.

c On some computers, branches take three times as long as other instructions to execute. Would the code generation scheme given in your answer to part (a) be suitable? What changes should be made and why?

(The three parts carry, respectively, 50%, 25% and 25% of the marks).

4 Suppose that expressions are represented using the following Miranda abstract syntax tree:

```
exp ::= Plus exp exp | Var name |
        Const num | Index name exp
```

For example, the expression `x+1` is represented as

```
Plus (Var "x") (Const 1)
```

The `Index` constructor is used to encode references to one-dimensional arrays. For example the expression `A[x+1]` is represented as

```
Index "A" (Plus (Var "x") (Const 1))
```

a Use Miranda to sketch the design of a simple code generator for expressions represented using this data type. The output from your code generator should be a list of instructions for a simple machine with plenty of registers. Use register targetting and the Sethi-Ullman weights algorithm to minimise the number of registers used. Take care to state any assumptions you make about the target instruction set.

b Now consider the same expression language augmented with function calls taking one parameter. The modified abstract syntax tree is as follows:

```
exp ::= Plus exp exp | Var name |
        Const num | Index name exp | Call name exp
```

Suppose that the function requires the parameter to be passed in register zero, and returns its result in register zero, but use any other registers.
Show how your code generator would be extended to handle this.

(The two parts carry, respectively, 50% and 50% of the marks).

End of Paper