COMPUTER SCIENCE TRIPOS Part Ib

Wednesday 7 June 1995 1.30 to 4.30

Paper 5

Answer **five** *questions.*
No more than **two** *questions from any one section are to be answered.*
Submit the answers in five **separate** *bundles each with its own cover sheet.*
Write on **one** *side of the paper only.*

SECTION A

1 Processor Architecture

A classical RISC five-stage pipeline is depicted below:

instruction fetch	register fetch	execute	memory access	register write back

Using the above pipeline as a basis for discussion, explain the following:

(*a*) What are *data bypasses* (sometimes called feed-forward paths)? [5 marks]

(*b*) The above pipeline is likely to have two bypasses. Between which stages are the bypasses required and why? [5 marks]

(*c*) Why do load delay slots arise? [5 marks]

(*d*) Which of the following code segments will execute more quickly on the above pipeline and why (you may assume that there are no cache misses)?

Code segment 1

```
load r1,4(sp)
load r2,8(sp)
load r3,12(sp)
add r1,r2,r4 # r4=r1+r2
add r3,r4,r4 # r4=r3+r4
```

Code segment 2

```
load r1,4(sp)
load r2,8(sp)
load r3,12(sp)
add r2,r3,r4 # r4=r2+r3
add r1,r4,r4 # r4=r1+r4
```

[5 marks]

2 Computer Architecture

Give a simplified schematic of a PC showing the main CPU, its support chips, and bus interface. [10 marks]

Describe how interrupts and bus arbitration are handled in this design. [10 marks]

3 Digital Communication I

Compare the functions of a *MAC level bridge* with an *IP router*. In what circumstances is it more appropriate to use one than the other? [6 marks]

Discuss the tables inside both bridges and routers used to control the acceptance and forwarding of packets. Indicate both how these tables are used and how information is put in the tables. How quickly can the tables be searched in each case? [10 marks]

What features might be added to a router or bridge to improve some aspects of network security? [4 marks]

4 Graphics

Why are matrix representations used to describe point transformations in computer graphics? [6 marks]

Describe how to represent three different 2D transformations as matrices. [6 marks]

Explain how to derive a sequence of transformations to achieve the overall effect of performing a 2D rotation about an arbitrary point. [8 marks]

SECTION B

5 Programming in C and C++

Students have been set a programming exercise where they are expected to write a subroutine to perform a certain operation, and they have been told that they should write it in the language C. One of the offerings submitted for assessment is the following. The student involved, who had read a book by Dijkstra but not understood it, proudly proclaims that of course this program has not actually been tried on a computer.

Explain what the program is (probably) supposed to do, and identify as many problems with it as you can, given that in this examination questions are expected to be completed in around 30 minutes.

```
typedef unsigned long int thing;

#define swap(p,q) v = p; p = q; q = v;

void fast(thing a[], int left, int write)
{                    /**********************
    int i, j;         * Declare variobles! *
    thing v;          **********************/
    if (write-left > 1)
    {   v = a[write];
        i = left, j = write;
        for (;;)
        {   while (a[++i] < v};
            while (a[--j] >= v);
            if (i < j) swap(a[i], a[j]);
            else break;
        }
        fast(a, left, i-1);  // re-curse here.
        fast(a, i, write)
    }
}
```

[20 marks]

6 Compiler Construction

Give a BNF syntax for unsigned floating point constants that would allow constants such as

```
1.2          3E4          0.23E-8
```

but not

```
.123         20.          21.E4
```

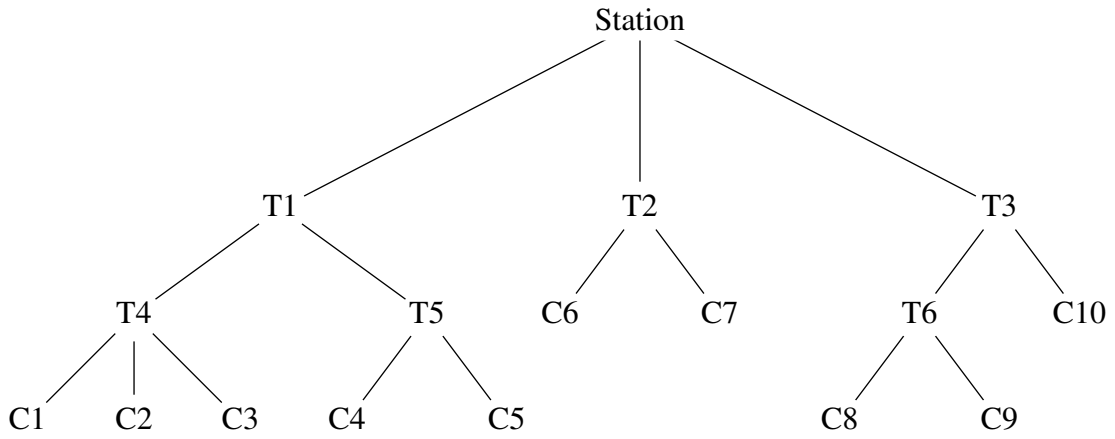
[6 marks]

Construct a finite state machine that could be used in a lexical analyser to read in such floating point numbers. [7 marks]

Show how this finite state machine can be extended to evaluate the floating point number it reads. [7 marks]

7 Prolog for Artificial Intelligence

Electricity consumers are supplied with electricity from an electricity generating station. Electricity is distributed from the station to the various consumers through a network of transformers as shown in this diagram:



Nodes C1 to C10 are consumers; nodes T1 to T6 are transformers. Each consumer has a direct connection to only one transformer. Sometimes a transformer may malfunction or need to be taken out of service temporarily.

Devise a data structure in Prolog to represent networks like this. [5 marks]

Write a Prolog procedure to define predicate `supplies` such that goal `supplies(X,Y)` succeeds when there is an electricity supply from node `X` to node `Y` in the network. [10 marks]

Explain how to enhance your representation to permit multiple connections which can be used to make a supply around a transformer which has been taken out of service. [5 marks]

Note: The `supplies` predicate may contain arguments in addition to `X` and `Y` at your discretion.

8 Databases

What forms of error do database normal forms seek to render less probable? [3 marks]

Define Second Normal Form. [3 marks]

Define Fourth Normal Form. [4 marks]

Give an example of a database schema that is in 2NF form but *not* in 4NF, showing how it is possible for mutually contradictory data to be recorded. Explain your assumptions about the semantics underlying the database schema. [7 marks]

Are there applications where normal forms are unnecessary or even unhelpful? [3 marks]

SECTION C

9 Logic and Proof

Describe the main features and applications of ordered binary-decision diagrams (OBDDs). [3 marks]

Describe an algorithm for computing OBDDs efficiently. Be careful to distinguish optimisations from essential features of the data structure. [6 marks]

Give an example to demonstrate how the variable ordering can affect the size of the OBDD. [3 marks]

Using the alphabetic ordering on variables, construct OBDDs for

$$[(P \wedge Q) \wedge R] \rightarrow [(R \wedge Q) \wedge P]$$

and

$$(P \vee Q) \wedge (Q \rightarrow P)$$

[4 + 4 marks]

10 Foundations of Functional Programming

$$
\begin{array}{lrcl}
\text{Let} & A & \equiv & \lambda x\ y.\ y\ (x\ x\ y) \\
 & \Theta & \equiv & A\ A \\
 & suc & \equiv & \lambda n\ f\ x.\ f\ (n\ f\ x) \\
 & true & \equiv & \lambda x\ y.\ x \\
 & false & \equiv & \lambda x\ y.\ y
\end{array}
$$

Reduce each of the following λ-terms to normal form (if possible) and to head normal form (hnf) (if possible).

Θ	suc	Θ	$(suc\ x)$
Θ	$true$	Θ	$false$
Θ	$(\lambda x.\ x\ x)$	Θ	$(\lambda x.\ f\ x\ x)$

[12 marks]

If M has no hnf then $M[N/x]$ has no hnf, for all N. Use this fact to show the following:

If M has no hnf then $M\ N$ has no hnf, for all N. [8 marks]

11 Complexity Theory

Comment on each of the following statements about Computational Complexity. Indicate any places where their wording is not sufficiently precise. For each, decide whether the statement is true, partially true, true but improperly justified, false or just muddled.

(a) Given a variant on Quicksort that uses a median-of-three procedure to select pivots, it seems hard to identify exactly what ordering of input data will make the quicksort behave worst. But because there are $N!$ possible different orderings and $N!$ is a bit like 2^N the problem is an NP one.

(b) If we could solve the Boolean Satisfiability problem efficiently we could use that to simulate the behaviour of *any* Turing machine and hence solve all other problems efficiently.

(c) To keep your secret treasure safe you intend to dig out a series of caves and tunnels forming a maze. You invent a graph for which you know a Hamiltonian circuit (for example, you start by putting in edges to make that circuit and then add lots more to make the graph more complicated). The Hamiltonian circuit problem is known to be NP complete, so given just the graph nobody except you will be able to find the Hamiltonian circuit. You wire up the tunnels so that the treasure can only be reached (safely!) by traversing the Hamiltonian circuit. Being arrogant you then pin details of the graph on your door. NP completeness means that your treasure is almost certainly secure.

You should provide a brief overview of any result, construction or proof that you refer to, but you are not expected to work through the details.

[20 marks]

12 Semantics

The abstract syntax of IMP commands is given by the following grammar:

$$Com ::= \texttt{skip} \mid Pvar := Iexp \mid Com \ ; Com \mid$$
$$\texttt{if}\ Bexp\ \texttt{then}\ Com\ \texttt{else}\ Com \mid \texttt{while}\ Bexp\ \texttt{do}\ Com$$

where $Iexp$ and $Bexp$ are syntactic categories of integer and boolean expressions and $Pvar$ is a set of program variables. Let $States$ be $[Pvar \to \mathbb{Z}]$ and $Cont$, the cpo of *continuations*, be $[States \to A_\perp]$, where A is an unspecified set of program *answers*. A continuation represents what is to be done with the state resulting from the execution of a command in order to return the result of the whole program.

The *continuation semantics* of IMP is defined by giving the meaning $[\![C]\!]$ of each $C \in Com$ as a function which takes a continuation, representing what is to be done when the command has finished, together with a state in which the command is to be executed, and returns an answer:

$$[\![-]\!] \ : \ Com \to (Cont \to (States \to A_\perp)).$$

One clause of the definition of $[\![C]\!]$ is

$$[\![\texttt{skip}]\!]\ k\ S = k(S).$$

Complete the definition of the continuation semantics of IMP commands (expressing their usual behaviour). You may assume that the functions

$$[\![-]\!] \ : \ Iexp \to (States \to \mathbb{Z})$$
$$[\![-]\!] \ : \ Bexp \to (States \to \mathbb{B}) \quad \text{where } \mathbb{B} = \{true, false\}$$

have already been defined. [9 marks]

Now add a new command `abort` to Com and a new error value Err to A. The intended behaviour of `abort` is immediately to terminate the entire program, returning Err. Extend the continuation semantics of IMP by giving the definition of $[\![\texttt{abort}]\!]$. [2 marks]

Now add two further new command forms:

$$Com \ ::= \ \ldots \mid \texttt{abort} \mid \texttt{exit} \mid Com\ \texttt{orelse}\ Com$$

The intended behaviour of (C_1 `orelse` C_2) is that it executes exactly like C_1 unless C_1 hits an `exit` command, in which case further execution of C_1 is abandoned and C_2 is executed starting in the state at which C_1 encountered the `exit`. If C_1 does not encounter an `exit` then C_2 is ignored. An `exit` command without an enclosing `orelse` behaves like `abort`.

Give a revised continuation semantics to every command of IMP with `abort`, `exit` and `orelse` which reflects this behaviour and in which the denotation of $C \in Com$ is a function which takes *two* continuations and a state and returns an element of $A_\perp$:

$$[\![-]\!] \ : \ Com \to (Cont \to (Cont \to (States \to A_\perp))).$$

Hint: The first continuation is the ordinary default continuation and the second is the continuation to be applied if the command `exit`s. [9 marks]