COMPUTER SCIENCE TRIPOS Part IB

Wednesday 4 June 1997 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. On each cover sheet, write the numbers of* **all** *attempted questions, and circle the number of the question attached.*

Write on **one** *side of the paper only.*

SECTION A

1 Digital Communication I

Compare the multiplexing aspects of packet switching and circuit switching. [5 marks]

ATM has been described as a compromise between circuit switching and packet switching. Explain this with respect to multiplexing. [5 marks]

Consider a generic switch with multiple inputs and outputs. Describe the functions that are performed in moving information from an input to an output in (a) a circuit switch and (b) a packet switch. How is contention for output transmission capacity resolved in each case? [10 marks]

2 Computer Graphics and Image Processing

Describe the z-buffer polygon scan conversion algorithm. [10 marks]

Explain how the A-buffer improves on the z-buffer. [5 marks]

Explain what a form factor is (in radiosity). Outline an implementable method of calculating form factors. [5 marks]

3 Computer Architecture

Write short notes describing the operation of *each* of the following:

(*a*) DRAM memory cell

(*b*) DRAM memory chip

(*c*) DRAM memory module

(*d*) DRAM memory module refresh cycle

(*e*) interleaved DRAM memory modules

[4 marks each]

4 Processor Architecture

The ARM processor allows every instruction to be conditionally executed whereas many processors allow only branches to be conditional. In ARM assembler conditional execution is indicated by one of the following postfix mnemonics:

Condition code mnemonic	Meaning
`EQ`	equal
`NE`	not equal
`CS`	unsigned higher or same
`CC`	unsigned lower
`MI`	negative
`PL`	positive or zero
`VS`	overflow
`VC`	no overflow
`HI`	unsigned higher
`LS`	unsigned lower or same
`GE`	greater or equal
`LT`	less than
`GT`	greater than
`LE`	less than or equal
`AL`	always execute (default)

(*a*) Using the following C-code excerpt as a basis for argument, briefly explain how conditional execution of every instruction can reduce the size of the code when compared with an instruction set which allows only conditional branches.

```
if (x==0) a=y; else a=y*x;
```

[7 marks]

(*b*) What effect do branch instructions have on a processor pipeline which does not perform branch prediction (i.e. as on the ARM 7)? [7 marks]

(*c*) What effect do conditional instructions have on the pipeline if the condition fails, and why is this effect preferable to that of short conditional branches (assuming no branch prediction)? [6 marks]

SECTION B

5 Programming in C and C++

A slightly clumsy programmer had been lagging behind the company productivity targets and needed to write some C++ code in a hurry. Almost remembering an old optimising compilers question from student days, this programmer produced a file containing the text:

```
struct List { int head; struct list *tail; };

struct List readlist()
{   int i;
    struct List *p, *q, *t;
L1: p = NULL;
L2: while (scanf("%d", i) = 1)
    /* scanf reads an integer and returns 1
       if it finds one correctly /*
    [
L3:     t = malloc(sizeof(List *));
        if (t == 0) printf("oops no memery\n");
        else t->hd = i;
L4:          t->tl = 0;
        if (p == NULL)
            p = q = t;
        else
            q->tl = t, q = t;
    }
L5: return p;
}
```

Unfortunately the programmer had forgotten what this was supposed to achieve; you are asked to help re-create an explanation for the code and to identify problems (of either style or correctness) in it. You do not need to provide a correct version of the program: just draw attention to as many errors or oddities as you can. Suggest two ways in which a move from C to C++ might allow the structure of the code to be improved. [20 marks]

6 Compiler Construction

Describe an efficient tree pattern-matching algorithm that could be used to find a cheapest covering of an abstract syntax tree by pattern templates with given costs. Illustrate your algorithm using the following templates:

```
#1 R  <- k                    cost: 1

#2 R  <- f(R, k)              cost: 2

#3 R  <- f(R, R)              cost: 2

#4 R  <- f(R, f(R, k))        cost: 3

#5 R  <- f(f(R, k), R)        cost: 4
```

and the following tree:

```
f(f(k,k),f(k,k))
```

[20 marks]

7 Prolog for Artificial Intelligence

The *next-highest member* of a list of integers is the second-largest member of the list. For example, for the list `[1, 4, 1, 5, 2]`, the next-highest member is `4`.

Write a Prolog program to find the next-highest member of a list of integers. For example, the goal `nexthi([1, 4, 1, 5, 2], X)` should instantiate `X` to `4`. Your program may assume that the largest member is not repeated in the list. The goal should fail if the next-highest member does not exist.

[20 marks]

8 Databases

Describe the *ANSI/SPARC architecture* for database management, and show how its use contributes to the enforcement of *data independence.* [6 marks]

Explain briefly the terms *entity*, *attribute*, *relationship.* [3 marks]

Both *network* and *relational* Database Management Systems provide Data Definition Languages so that a conceptual model can be expressed as a formal database schema. Explain the ways in which the primitives supplied for data representation differ between the two models. Illustrate your answer by considering the following database scenario:

> The employees of a company are grouped into a number of departments. Each department has its own director, who is not necessarily a member of that department. Independently, many employees are assigned a manager who is responsible for their career development.

[Any assumptions that you make about the nature of the data should be stated explicitly.] [11 marks]

SECTION C

9 Complexity Theory

Comment on each of the following assertions, explaining whether they are right, wrong or imprecisely stated. State explicitly any standard results needed to justify your assertions and, in cases where the statement made is almost but not quite correct, attempt to clarify or mend it.

(*a*) If one had a special piece of hardware that could solve the boolean-satisfiability problem 3-SAT in time $n \log(n)$ then there would be some constant K such that it would be possible to solve all NP-complete problems in time no worse than n^K.

(*b*) The square root of an n-bit number can be computed in time proportional to $n \log(n) \log\log(n)$ on an ordinary computer, therefore computing square roots is not an NP problem.

(*c*) Every task that can be performed in Polynomial Time will be solvable on a conventional computer in a reasonable amount of time.

(*d*) Any task that is NP will take an unreasonable amount of time if an attempt is made to solve it using an ordinary computer.

(*e*) Deciding whether black or white will win the game of chess if both players behave totally logically is not an NP problem because it is of finite size.

[20 marks]

10 Logic and Proof

Using binary predicate symbols EQ ($=$) and LT ($<$) and binary function symbols SUM ($+$) and $PROD$ ($\times$), write down predicate calculus formulae that formalise the following statements (some of which are false) about the natural numbers:

(a) *there is a smallest number* [2 marks]

(b) *there is no largest number* [2 marks]

(c) *every number is the sum of two squares* [2 marks]

(d) *there exist two numbers whose product is less than their sum* [2 marks]

For each of the formulae (e) to (j) below, state whether it is valid (true in all interpretations) or not. Either give an informal justification of the validity, or outline a falsifying interpretation.

(e) $(\forall x\ P(x)) \to (\exists x\ P(x))$ [2 marks]

(f) $(\exists x\ P(x)) \to (\forall x\ P(x))$ [2 marks]

(g) $((\forall x\ P(x))\ \wedge\ (\forall x\ Q(x))) \to (\forall x\ (P(x) \wedge Q(x)))$ [2 marks]

(h) $((\exists x\ P(x))\ \wedge\ (\exists x\ Q(x))) \to (\exists x\ (P(x) \wedge Q(x)))$ [2 marks]

(i) $(\forall x\ \exists y\ P(x,y)) \to (\exists y\ \forall x\ P(x,y))$ [2 marks]

(j) $(\exists x\ \forall y\ P(x,y)) \to (\forall y\ \exists x\ P(x,y))$ [2 marks]

11 Foundations of Functional Programming

(a) Give the definition of a head-normal form and head reduction of a λ-term. Argue that every normal form is a head-normal form. [4 marks]

(b) Let $Y_M \equiv \lambda f.WWM$, where $W \equiv \lambda x.\lambda z.f(xxz)$ and M is an arbitrary term. Give a head-normal form and normal form for the following λ-terms, or indicate why they do not exist:

(i) Y_M

(ii) $Y_M(KI)$, where $K \equiv \lambda x.\lambda y.x$ and $I \equiv \lambda x.x$

(iii) $Y_M(K)$, where K is as above

[You may assume that head reduction always terminates when a head-normal form exists.] [6 marks]

(c) A λ-term is solvable if there exist variables $x_1, \ldots, x_n$ and λ-terms $N_1, \ldots, N_m$ for $n, m \geqslant 0$ such that $(\lambda x_1 \ldots \lambda x_n.M)N_1 \ldots N_m = I$.

Show that every head-normal form is solvable. [4 marks]

For each term in (b), prove that it is solvable or that it is unsolvable. [6 marks]

12 Semantics of Programming Languages

The phrases, P, of the language LC are specified by:

$$\begin{aligned} P &::= C \mid E \mid B \\ C &::= \textbf{skip} \mid \ell := E \mid C\,;C \mid \textbf{if } B \textbf{ then } C \textbf{ else } C \mid \textbf{while } B \textbf{ do } C \\ E &::= n \mid !\ell \mid E \; iop \; E \\ B &::= \textbf{true} \mid \textbf{false} \mid E \; bop \; E \end{aligned}$$

where ℓ ranges over storage locations, n over integers, *iop* over integer-valued operations, and *bop* over boolean-valued operations. Describe the operational semantics of LC in terms of an inductively defined transition relation, $\to$, between configurations $\langle P, s\rangle$, where s is a finite partial function from locations to integers. State which are the *terminal* configurations and explain what it means for a configuration to be *stuck*. [6 marks]

Call a configuration $\langle P, s\rangle$ *sensible* if the set of locations on which s is defined, $dom(s)$, contains all the locations that occur in the phrase P. Prove by induction on the structure of P that for all s, if $\langle P, s\rangle$ is sensible then $\langle P, s\rangle$ is not stuck. [6 marks]

Prove by Rule Induction for $\to$ that if $\langle P, s\rangle \to \langle P', s'\rangle$ and $\langle P, s\rangle$ is sensible, then so is $\langle P', s'\rangle$ and $dom(s') = dom(s)$. Deduce that a stuck configuration can never be reached by a series of transitions from a sensible configuration. [8 marks]