# COMPUTER SCIENCE TRIPOS Part Ib

---

Thursday 5 June 2003 1.30 to 4.30

---

Paper 6

*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.*

**You may not start to read the questions printed on the subsequent pages of this question paper until instructed that you may do so by the Invigilator**

## SECTION A

### 1 Data Structures and Algorithms

(*a*) In the Burrows–Wheeler Block Compression algorithm it is necessary to sort all the suffixes of a vector of possibly tens of millions of bytes.

(*i*) Explain why Shell sort using a simple character by character string comparison function is unlikely to be satisfactory. [2 marks]

(*ii*) Describe in detail the data structures and algorithms you would use to sort the suffixes and explain why your method is an improvement. You may assume that you have plenty of RAM available. [8 marks]

(*b*) In the Burrows–Wheeler Block Compression algorithm a sequence of small positive integers are transmitted using Huffman encoding.

(*i*) Describe how the Huffman code is constructed. [4 marks]

(*ii*) Outline how it can be represented compactly in the compressed file. [6 marks]

### 2 Computer Design

(*a*) Briefly describe the differences between a microprocessor's interface bus, a system I/O bus (such as PCI), and a peripheral interface. Consider the bandwidth characteristics and physical implementation of each. [8 marks]

(*b*) Show how four 64Mbit byte-wide DRAM chips could be interfaced to a simple 32-bit microprocessor. Give a schematic diagram showing the connections, and a timing diagram to demonstrate the operation of the control signals. [8 marks]

(*c*) Why might accesses to sequential DRAM addresses be treated differently from non-sequential ones? [4 marks]

## 3 Digital Communication I

(*a*) Define the term *multiplexing* as applied to communication systems. [4 marks]

(*b*) Describe *three* types of multiplexing, identifying in each case

(*i*) mechanisms by which symbols are associated with particular channels;

(*ii*) mechanisms by which transmitters are assigned channel resource;

(*iii*) characteristics of the multiplexed channels;

(*iv*) applications which are suited to the type of multiplexing. [16 marks]

## 4 Concurrent Systems and Applications

A distributed system is being designed to hold details of appointments on behalf of its users. The system comprises a single server holding the appointments and a number of clients which interact with the server over a network supporting an unreliable datagram-based protocol such as UDP.

(*a*) The server is to provide operations using a remote procedure call (RPC) interface with exactly-once semantics. For instance,

```
boolean addEntry (Client c, Client d, Appointment a);
```

is used by client `c` to add an entry to the diary of client `d`. Describe how this RPC system can be implemented, including

(*i*) how parameters are marshalled and unmarshalled;

(*ii*) how exactly-once semantics are achieved;

(*iii*) how threads are used, both within a client and within the server.

You may assume that the clients already know an appropriate network address to contact the server and that separate mechanisms are used for authentication and for security in general. [10 marks]

(*b*) The system is to be extended to support transactional-style accesses by providing three further operations,

```
void startTransaction (Client c);
void abortTransaction (Client c);
boolean commitTransaction (Client c);
```

Define *strict isolation* and describe how, in this case, it could be enforced by *timestamp ordering*. [10 marks]

## SECTION B

### 5 Computer Graphics and Image Processing

($a$) We use homogeneous coordinates to represent transformations in 3D space:

$$\begin{bmatrix} x'_H \\ y'_H \\ z'_H \\ w'_H \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & b_1 \\ a_{21} & a_{22} & a_{23} & b_2 \\ a_{31} & a_{32} & a_{33} & b_3 \\ c_1 & c_2 & c_3 & d \end{bmatrix} \begin{bmatrix} x_H \\ y_H \\ z_H \\ w_H \end{bmatrix}$$

($i$) Explain how to convert standard 3D coordinates, $(x, y, z)$, to homogeneous coordinates and how to convert homogeneous coordinates to standard 3D coordinates. [2 marks]

($ii$) Describe the types of transformations provided by each of the four blocks of coefficients in the matrix ($a_{11} \cdots a_{33}$, $b_1 \cdots b_3$, $c_1 \cdots c_3$, and $d$). [6 marks]

($iii$) Explain what transformation is produced by each of the following matrices:

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \qquad \begin{bmatrix} 1 & 0 & p & -p(1+r) \\ 0 & 1 & q & -q(1+r) \\ 0 & 0 & 1+r & -r(1+r) \\ 0 & 0 & 1 & -r \end{bmatrix}$$

[4 marks]

($b$) Describe an algorithm (in 2D) which clips an arbitrary polygon against an arbitrary axis-aligned rectangle. [8 marks]

## 6 Compiler Construction

Each of the following statements may be true, false, or nonsensical. Indicate which and (respectively) provide a (one-sentence) justification of why it holds, a counter-example or other explanation of why it fails, or corrected statement.

($a$) In a language with stack-allocated free variables, the static chain pointer always points to the caller's stack frame.

($b$) If each procedure sets up one exception handler whose scope includes all of the procedure calls it makes, then the entries in the exception handler stack and those in the dynamic chain will be in 1–1 correspondence.

($c$) Fortran gives semantic meaning to parentheses, i.e. `a+b+c` permits compiler re-arrangement whereas `(a+b)+c` does not. This matters for floating point. However, since Fortran '`+`' is left associative the above two expressions yield the same parse tree, and we have a contradiction.

($d$) Two alternatives to conservative garbage collection are liberal garbage collection or new labour style de-allocation.

($e$) Any type 3 (regular) grammar is also a type 2 (context-free) grammar. Hence any lexer generated by `lex` (or `JLex`) could instead be generated by `yacc` (or `Cup`).

($f$) When a compiler for a language like Java compiles $e_1$+$e_2$, it computes the types of $e_1$ and $e_2$ so that it can treat it as $e_1$+`(float)`$e_2$ if $e_1$ is of type `float` and $e_2$ is of type `int`.

($g$) A syntax-tree interpreter can fail (give an error) when evaluating a variable name which does not appear in the current environment. Any program which fails using static scoping will fail using dynamic scoping.

($h$) A syntax-tree interpreter can fail (give an error) when evaluating a variable name which does not appear in the current environment. Any program which fails using dynamic scoping will fail using static scoping.

($i$) A dynamically-typed language is one in which the type of a value is carried around at run-time; a type error is given at run-time if values are used inappropriately. Such languages must be dynamically linked otherwise type errors will occur.

($j$) Java `.class` files and ELF-style `.o` (or `.obj`) object files represent similar information, i.e. compiled code, a list of symbols made available to other functions, and a list of undefined symbols which the file expects to be defined elsewhere.

[2 marks each]

## 7 Artificial Intelligence I

A simple game works as follows. We have a board divided into $n$ by $m$ square cells. We also have an unlimited number of L-shaped tiles, each made to cover exactly three squares. The tiles can appear in any of the four possible orientations. Our aim is to cover the board completely with non-overlapping tiles.

($a$) A single tile on the board can be described using a list such as `[[1,1],[1,2],[2,1]]` containing three tuples, specifying the position of each part of the tile on the board. Consider the following Prolog predicate, which is true if the six variables describe a correct, L-shaped tile.

```
tile([[A,B],[C,D],[E,F]]) :- C is A+1, D is B, E is A, F is B-1;
                             C is A+1, D is B, E is A, F is B+1;
                             C is A-1, D is B, E is A, F is B+1;
                             C is A-1, D is B, E is A, F is B-1.
```

Explain what happens in response to a query of the form

```
tile([[4,5],[B,C],[D,E]]).
```

Keep in mind the effects of backtracking. [2 marks]

($b$) Write a Prolog predicate `goodplace([[A,B],[C,D],[E,F]],[N,M])` that is true if `[[A,B],[C,D],[E,F]]` is a validly shaped tile and all of its parts lie within an `N` by `M` board. Your predicate should behave under backtracking in such a way that the response to a query of the form

```
goodplace([[10,4],[B,C],[D,E]],[10,10]).
```

is to find the unspecified values for all tiles which have a valid shape and fall within the board. In this example there would be two such tiles. [6 marks]

($c$) Write a Prolog predicate `tiling(Available,Solution,Size)`. Here, `Size` is the size of the board represented as above, `Solution` is a list of tiles that solves the problem, and `Available` is a list of available positions on a board of the given size. For example, if `Size` is `[2,2]` then `Available` is `[[1,1],[1,2],[2,1],[2,2]]`.

Your predicate should be true if the `Solution` given is a valid one, and should be capable of finding a valid `Solution` in response to a query such as

```
tiling([[1,1],[1,2],...,[10,10]],X,[10,10]).
```

Full marks will only be given for predicates that can exploit backtracking to find all possible solutions. [12 marks]

## 8 Databases

(*a*) Define the ACID properties of a transaction. [6 marks]

(*b*) Define what is meant by *strict two-phase locking* (strict 2PL). [4 marks]

(*c*) Assume that in addition to traditional Read and Write actions a DBMS supports increment and decrement actions: Inc and Dec (both are assumed to perform blind writes).

Consider the following two transactions.

$$T1 : [\mathsf{Inc}(A), \mathsf{Dec}(B), \mathsf{Read}(C)]$$
$$T2 : [\mathsf{Inc}(B), \mathsf{Dec}(A), \mathsf{Read}(C)]$$

(*i*) By considering some possible schedules of the above transactions, describe carefully the concurrency permitted by strict 2PL using just shared/read and exclusive/write locks. [3 marks]

(*ii*) Detail a way to gain more interleaving whilst still maintaining strict 2PL. [7 marks]

# SECTION C

## 9 Logic and Proof

(*a*) Define the concepts of a *true sequent*, a *valid sequent* and a *basic sequent*. You may take the concepts of true and valid formulæ as primitive. [3 marks]

(*b*) Gentzen claimed that his proof systems performed natural logical reasoning. Consider the following sequent calculus rules:

$$\frac{A, \Gamma \Rightarrow \Delta \quad B, \Gamma \Rightarrow \Delta}{A \vee B, \Gamma \Rightarrow \Delta} \qquad \frac{A, \Gamma \Rightarrow \Delta}{\exists x\, A, \Gamma \Rightarrow \Delta}$$

State the second rule's proviso and explain the intuitions behind each rule. [5 marks]

(*c*) Choose one of these rules and give a rigorous argument for its soundness, using the concept of a valid sequent. [6 marks]

(*d*) Precisely define the concept of the Most General Unifier (MGU) of two terms, giving examples. [6 marks]

## 10 Foundations of Functional Programming

Explain

($a$) a technique that you could use to implement a pure functional programming language using applicative order (eager) evaluation; [6 marks]

($b$) a technique suitable if your implementation must use some form of lazy or normal-order evaluation. [6 marks]

Comment on whether the behaviour of your implementation ($b$) is strictly and exactly the same as normal order evaluation in lambda-calculus. [3 marks]

Comment about similarities and differences between your two techniques and any predictions you can make about their relative simplicity, performance or convenience. [5 marks]

## 11 Complexity Theory

($a$) For each $k$, the $k$-clique problem is defined as the following decision problem:

> Given a graph $G$, does it contain a clique with at least $k$ vertices?

Show that $k$-clique is in P for each $k$. [6 marks]

($b$) The problem Clique is defined as the following decision problem:

> Given a graph $G$ and an integer $k$, does $G$ contain a clique with at least $k$ vertices?

Show that Clique is NP-complete, using the assumption that 3-SAT is NP-complete. [10 marks]

($c$) Explain why, if P=NP then there is a polynomial time algorithm for factorising numbers. [4 marks]

## 12 Semantics of Programming Languages

Consider the language below, in which $r$ ranges over the reals $\mathbb{R}$ and $n$ ranges over the subset $\mathbb{Z} \subset \mathbb{R}$ of integers.

$$\begin{aligned} T &::= \mathsf{int} \mid \mathsf{real} \mid \mathsf{bool} \mid T \to T \mid \{lab : T, \ldots, lab : T\} \\ e &::= n \mid r \mid \textbf{true} \mid \textbf{false} \mid \textbf{fn}\ x : T \Rightarrow e \mid e\ e \mid x \mid \{lab = e, \ldots, lab = e\} \mid \#lab\ e \end{aligned}$$

To allow any $\mathsf{int}$ to be used as a $\mathsf{real}$ we can use a subtype relation that satisfies

$$(\text{num})\frac{}{\mathsf{int} <: \mathsf{real}}$$

Suppose the typing rules are the standard ones, including the subsumption rule:

$$(\text{sub})\frac{\Gamma \vdash e : T \qquad T <: T'}{\Gamma \vdash e : T'}$$

($a$) Give the remainder of the rules required to define the subtype relation $T <: T'$. [7 marks]

($b$) For each of the following expressions, either give a type derivation (including derivations of any subtype relationships used) or explain why it is not typeable.

($i$) $(\textbf{fn}\ \mathrm{x} : \{\mathrm{p} : \mathsf{real}\} \Rightarrow \#\mathrm{p}\ \mathrm{x})\{\mathrm{p} = 1, \mathrm{q} = \textbf{true}\}$

($ii$) $(\textbf{fn}\ \mathrm{x} : \{\mathrm{q} : \mathsf{bool}\} \Rightarrow \#\mathrm{p}\ \mathrm{x})\{\mathrm{p} = 2.7, \mathrm{q} = \textbf{true}\}$

($iii$) $(\textbf{fn}\ \mathrm{x} : \{\mathrm{r} : \{\mathrm{p} : \mathsf{int}, \mathrm{q} : \mathsf{bool}\}\} \Rightarrow \#\mathrm{p}(\#\mathrm{r}\ \mathrm{x}))\{\mathrm{r} = \{\mathrm{p} = 3\}\}$

[11 marks]

($c$) List all the types $T$ for which $\{\} \vdash (\textbf{fn}\ \mathrm{y} : \mathsf{real} \Rightarrow 6) : T$ is derivable. [2 marks]

**END OF PAPER**