COMPUTER SCIENCE TRIPOS Part Ib

Wednesday 4 June 2003 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.*

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*) A million singleton sets each containing a distinct integer are to be successively combined by calls of `union`(S_1, S_2). The result represents the union of the two disjoint sets represented by S_1 and S_2. Interspersed among these calls are several calls of `inSameSet` where `inSameSet`(m, n) yields `true` if and only if m and n are integers now in the same set. Describe in detail how you would implement `union` and `inSameSet` assuming they will be called about one million and five million times, respectively. Explain why your solution is efficient. [10 marks]

(*b*) Describe in detail an implementation of Kruskal's algorithm for finding a minimum cost spanning tree of an undirected graph with positive integer costs on the edges that uses your version of `union` and `inSameSet`. [5 marks]

(*c*) Explain why the spanning tree is unique if all the edge costs are distinct. [5 marks]

2 Computer Design

(*a*) Name and describe *three* reasons for a cache miss. [6 marks]

(*b*) For *each* reason, suggest a technique for reducing the number of misses. [6 marks]

(*c*) Why might it be advantageous to use a set-associative cache instead of a fully associative one? [4 marks]

(*d*) Describe *two* techniques for reducing the miss penalty. [4 marks]

3 Digital Communication I

(*a*) Define the terms *capacity* and *latency* as applied to a communications channel. [4 marks]

(*b*) How can variable latency cause problems? You may wish to consider

(*i*) XON/XOFF flow control;

(*ii*) streaming media;

(*iii*) protocol timeouts. [6 marks]

(*c*) Describe the operation of a simple ARQ protocol with a window of a single packet. [4 marks]

(*d*) A simple ARQ scheme is used to provide reliable transport over a link where 80% of packets other than short acknowledgements experience a 1 ms delay, 10% experience a 10 ms delay, and 10% are lost. Acknowledgements always experience a 1 ms delay and are never lost. What would be the expected throughput in packets/sec if the timeout was

(*i*) 10 ms?

(*ii*) 12 ms?

Assume that the transmitter always has information to send and that transmission time is negligible.

It may be helpful to note that

$$\sum_{i=0}^{\infty} i\, x^i = \frac{x}{(1-x)^2}$$

[6 marks]

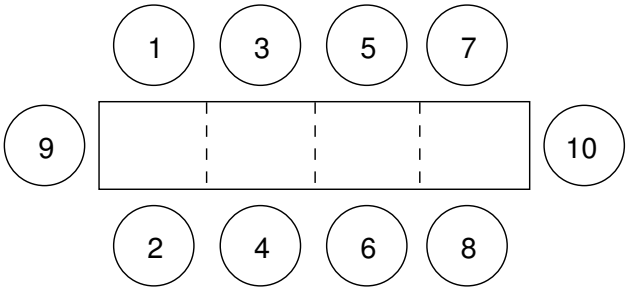
4 Concurrent Systems and Applications

In a concurrent system it is usual to distinguish between *safety properties* and *liveness properties*.

(*a*) What is meant by *mutual exclusion*, and how does a programmer ensure it is enforced in Java? Indicate whether it relates to safety or to liveness. [4 marks]

(*b*) What is *deadlock* and how can it occur? Again, indicate whether it relates to safety or to liveness. [4 marks]

A group of ten *feeding philosophers* has gathered to eat at a long trough, as shown below. They eat by leaning over the trough and so must be careful not to let their heads collide in the middle. To ensure safety they decide that around each position in the trough at most one philosopher is allowed to be eating at any one time. For instance, only one of 1, 2 or 9 could be eating at once, but there is no problem with 9, 3, 6 and 10 all eating together.



In a Java system, each philosopher is modelled by a thread which loops between eating and non-eating phases. There are four `TroughPosition` objects, corresponding to the four positions along the trough. Each philosopher has a reference to the position at which he or she is eating.

(*c*) Define a Java class `TroughPosition` which supports methods `startEating()` and `finishEating()` to ensure safe execution for the philosophers at that position. [6 marks]

(*d*) Can your solution suffer from deadlock? Either explain why it cannot occur, or explain how it could be avoided. [3 marks]

(*e*) Can your solution suffer from starvation – that is, can one thread continuously remain in the `startEating()` method? Either explain why this cannot occur, or explain how it could be avoided. [3 marks]

SECTION B

5 Computer Graphics and Image Processing

(*a*) Describe the A-buffer polygon scan conversion algorithm using 4×4 sub-pixels in each pixel. [10 marks]

(*b*) It is possible to represent continuous tone greyscale images using just black ink on white paper because of limitations in the human visual system. Explain how and why. [4 marks]

(*c*) Describe an algorithm which, given a greyscale image, will produce a black and white (bi-level) image of four times the resolution in each dimension which provides a good approximation to the greyscale image. [6 marks]

6 Compiler Construction

(*a*) A Java static method is defined in `class C` by

```
class C {
  public static int f(int x, int y) { int z = x; ...; return x+y*z;
                                      }
}
```

where '`...`' represents commands the details of which are not important to this question. It is called in an expression e of the form

```
f(f(1,2), f(3,4))
```

Give JVM (or other stack machine) code corresponding to the expression e and explain how this is derived from the syntax tree for e. [6 marks]

(*b*) Explain how the body of `f` above is mapped into JVM (or other stack machine) code, explaining the rôle of the registers FP and SP (precise details are not important, but their rôle should be well explained). You may write '`...`' for the translation of the '`...`' in `f`. [6 marks]

(*c*) Consider the Java class definitions:

```
class A {
  public int a1, a2;
  public void m() { println("I am an A with " + a1 + " and " + a2);
                  }
}
class B extends A {
  public int b1, b2;
  public void m() { println("I am a B with " + a1 + " and " + a2 +
                            " also with " + b1 + " and " + b2);
                  }
}
```

Describe the run-time storage layout for objects of `class A` and for those of `class B`, particularly noting the size and offsets of members and how a *cast* of an object of type `class B` to one of `class A` can be achieved.

Explain how calls to `m()` work, particularly in code like:

```
public static void g(B x) { h(x); }
public static void h(A x) { x.m(); }
```

[8 marks]

7 Artificial Intelligence I

The following Prolog relation appends a list `A` to a list `B` to give a list `C`.

```
append([],Y,Y).
append([H|T],Y,[H|Z]) :- append(T,Y,Z).
```

(*a*) Using the `append` relation, write a Prolog predicate `insert(X,Y,Z)` that is true if `X` can be inserted into a list `Y` to give a list `Z`. Your relation should be capable of using backtracking to generate all lists obtained from `Y` by inserting `X` at some point, using a query such as:

```
insert(c,[a,b],Z).
```

to obtain `Z=[c,a,b]`, `Z=[a,c,b]`, and `Z=[a,b,c]` and it should generate each possibility exactly once. [5 marks]

(*b*) Using the `insert` relation, write a Prolog predicate `perm(X,Y)` that is true if a list `Y` is a permutation of a list `X`. Your predicate should respond to a query such as

```
perm([a,b,c],Y)
```

by using backtracking to generate all permutations of the given list. [6 marks]

(*c*) We have a list of events `[e1,e2,...,en]`. A partial order can be expressed in Prolog by stating

```
before(e3,e4).
before(e1,e5).
```

and so on, where `before(a,b)` says that event `a` must happen before event `b` (although not necessarily immediately before). No ordering constraints are imposed other than those stated using `before`.

Given a list of events, a *linearisation* of the list is any ordering of its events for which none of the `before` constraints are broken. Given the example above and the list `[e1,e2,e3,e4,e5]`, one valid linearisation would be `[e3,e1,e2,e5,e4]`. However, `[e4,e2,e1,e5,e3]` is not a valid linearisation because the first `before` constraint does not hold.

Using the `perm` predicate or otherwise, and assuming that your Prolog program contains `before` constraints in the format suggested above, write a Prolog predicate `po(X,Y)` that is true if `Y` is a valid linearisation of the events in the list `X`. Your relation should be capable of using backtracking to generate all valid linearisations as a result of a query of the form

```
po([e1,e2,e3,e4,e5],Y).
```

[9 marks]

8 Databases

(*a*) (*i*) Define the operators in the core relational algebra. [5 marks]

(*ii*) Define the domain relational calculus. [4 marks]

(*iii*) Show how the relational algebra can be encoded in the domain relational calculus. [3 marks]

(*b*) A *constraint* can be expressed using relational algebra. For example, $R = \emptyset$ specifies the constraint that relation R must be empty, and $(R \cup S) \subseteq T$ specifies that every tuple in the union of R and S must be in T.

Consider the following schema.

`RockStar(`$\overline{\texttt{name}}$`, address, gender, birthday)`
`RockManager(`$\overline{\texttt{managername}}$`, starname)`

(*i*) Give a constraint to express that rock stars must be either male or female. [1 mark]

(*ii*) Give a constraint to express the referential integrity constraint between the `RockStar` and `RockManager` relations. (Note: `starname` is intended to be a foreign key.) [3 marks]

(*iii*) Give a constraint to express the functional dependency `name`$\rightarrow$`address` for the `RockStar` relation. [4 marks]

SECTION C

9 Logic and Proof

(*a*) For $k > 0$, let $\phi(k)$ be the formula

$$[(P_1 \leftrightarrow Q_1) \wedge \ldots \wedge (P_k \leftrightarrow Q_k)] \rightarrow R.$$

Prove that there exists an ordering of the propositional variables, namely P_1, P_2, …, Q_1, Q_2, …, R, such that the size of the ordered binary decision diagram (OBDD) for $\phi(k)$ increases linearly with k. [5 marks]

(*b*) Prove that there exists a variable ordering such that the size of the OBDD for $\phi(k)$ increases exponentially with k. [6 marks]

(*c*) Give a set of clauses suitable for attempting to prove $\phi(k)$ using resolution. [3 marks]

(*d*) Describe the computation that would result if the Davis–Putnam (DPLL) procedure were applied to these clauses. [6 marks]

10 Foundations of Functional Programming

Suppose that lists are to be represented in a pure functional manner using a convention where, for instance, a list with three members a_1, a_2 and a_3 is modelled by a lambda term

$$\lambda f.\lambda x.fa_1(fa_2(fa_3x))$$

(a) Give the lambda term that corresponds to an empty list. [2 marks]

(b) Explain how the normal list operations can be achieved on lists that are represented in this way. Specifically show how to create pure functional implementations of

(i) a test for an empty list, [2 marks]

(ii) adding a new item to the front of a list, and [4 marks]

(iii) finding the head and tail of a non-empty list. [9 marks]

(c) Show how a map function can be implemented for use with these functional lists, so that

```
map f [a,b,c] -> [f a, f b, f c]
```

[3 marks]

11 Semantics of Programming Languages

The Global Computer Corporation has just started working on a new language, tentatively named HMM, with syntax:

$$\begin{aligned} T &::= \mathsf{int} \mid \mathsf{unit} \mid T \to T \\ e &::= n \mid \textbf{skip} \mid \ell := e \mid !\ell \mid \textbf{fn}\ x : T \Rightarrow e \mid e\ e \mid x \end{aligned}$$

Their initial implementation, written in ML, has a one-step reduction function as follows.

```
fun reduce (Integer n,s) = NONE
  | reduce (Skip,s) = NONE
  | reduce (Deref l,s) = (case lookup  (s,l) of
         SOME n => SOME(Integer n,s)
       | NONE => SOME(Integer 0, (l,0)::s ))
  | reduce (Assign (l,e),s) = (case e of
        Integer n => (case update (s,(l,n)) of
                          SOME s' => SOME(Skip, s')
                        | NONE => SOME(Skip, (l,n)::s))
      | _ => (case reduce (e,s) of
                  SOME (e',s') => SOME(Assign (l,e'), s')
                | NONE => NONE  ) )
  | reduce (Var n,s) = raise (Reduce "bogus unbound Var")
  | reduce (Fn (t,e),s) = NONE
  | reduce (App (e1,e2),s) = (case e1 of
        Fn (t,e) =>  SOME (subst e2 0 e,s)
      | _ => (case reduce (e1,s) of
                  SOME (e1',s') => SOME(App (e1',e2), s')
                | NONE => NONE ))
```

This uses the standard auxilary functions below for manipulating association lists and for substituting an expression for a De Bruijn index in another expression. Following their tradition, they sometimes execute badly-typed programs.

```
update:(string*int) list * (string*int) -> (string*int) list option
lookup:(string*int) list * string -> int option
subst:expr -> int -> expr -> expr
```

(*a*) Give an inductive definition of a reduction relation $\langle e, s\rangle \longrightarrow \langle e', s'\rangle$ which corresponds exactly to their implementation. Say carefully what e and s range over in your semantics. [11 marks]

(*b*) Explain how the HMM semantics differs from the "standard" L2 semantics. For each difference, give the standard semantic rule(s), an expression that would behave differently using them, and any reason(s) why one or the other would be a better language design. [9 marks]

12 Complexity Theory

If $A \subseteq \Sigma_1^*$ and $B \subseteq \Sigma_2^*$ are two languages over the alphabets Σ_1 and Σ_2 respectively, we write $A \leq_P B$ to denote that A is polynomial-time reducible to B.

(a) Give a precise definition of $\leq_P$ [2 marks]

(b) Is the relation $\leq_P$ on languages:

(i) reflexive?

(ii) symmetric?

(iii) transitive?

Give a proof for your answer in each case. [9 marks]

(c) If Σ is an alphabet, show that if P = NP then every language $L \subseteq \Sigma^*$ in NP is NP-complete except $\emptyset$ and Σ^*. Why are these two exceptions not NP-complete? [9 marks]

END OF PAPER