COMPUTER SCIENCE TRIPOS Part IB

Wednesday 7 June 2006 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

STATIONERY REQUIREMENTS

Script Paper

Blue Coversheets

Tags

SECTION A

1 Data Structures and Algorithms

Dijkstra developed an efficient algorithm to find shortest paths on a directed graph from a designated source vertex to all other vertices, but only on graphs with non-negative edge weights.

(*a*) Give a clear and complete explanation of the algorithm. Be sure to cover its use of *relaxation* and to explain what happens if some vertices are not reachable from the source. [5 marks]

(*b*) Give a correctness proof for the algorithm. You may use the *convergence lemma* without having to prove it. [5 marks]

[Hint: here is the convergence lemma. **If** $s \rightsquigarrow u \rightarrow v$ is a shortest path from s to v, and at some time $d[u] = \delta(s, u)$, and at some time after that the edge (u, v) is relaxed, **then**, from then on, $d[v] = \delta(s, v)$.

Additional hint on notation: $s \rightsquigarrow u$ = path from s to u consisting of 0 or more edges (0 when $s \equiv u$); $u \rightarrow v$ = path from u to v consisting of precisely one edge; $d[u]$ = weight of the shortest path found so far from source s to vertex u; $\delta(s, v)$ = weight of shortest existing path from s to v.]

(*c*) Why does the algorithm require non-negative edge weights? [2 marks]

(*d*) Would the algorithm work if the only negative weights were on edges leaving the source? Justify your answer with a proof or counterexample. [5 marks]

(*e*) Consider the following approach for finding shortest paths in the presence of negative edges. "Make all the edge weights positive by adding a sufficiently large biasing constant to each; then find the shortest paths using Dijkstra's algorithm and recompute their weights on the original graph." Will this work? Justify your answer with a proof or counterexample. [3 marks]

2 Computer Design

(*a*) Why do pipelines exhibit branch and load delays? [6 marks]

(*b*) What impact does pipeline length have on clock frequency? [4 marks]

(*c*) Why might a shorter pipeline result in a more power-efficient design? [4 marks]

(*d*) Recently we have seen microprocessor manufacturers release dual-processor chips where each processor has a shorter pipeline than the earlier single-processor per chip designs. What sort of applications might run better on the older chips and *vice versa*? [6 marks]

3 Digital Communication I

(*a*) Describe the concepts of *circuit switching* and *packet switching.* [5 marks]

(*b*) What are the fundamental advantages of each over the other? [5 marks]

(*c*) What is the role of buffering and buffering policy in each approach? [5 marks]

(*d*) There is an expectation that in the near future telephony will move from circuit switching to packet switching. Why is this so in light of the advantages of each approach? [5 marks]

4 Concurrent Systems and Applications

(*a*) The following Java interface describes the API for a first-come, first-served (FCFS) mutual exclusion system, designed to work even if threads are interrupted in the `enter` and `exit` routines.

```
interface FCFS {
  public void enter();
  public void exit();
}
```

Sketch a concrete class, `FCFSImpl`, implementing this interface, which does not need to be re-entrant, ensuring that you satisfy the following requirements:

(*i*) if a thread is interrupted while executing the entry protocol, it should abort its attempt to gain entry and cleanly terminate the call; you may assume that the calling code will not then enter the critical region; [6 marks]

(*ii*) the exit protocol should notify a particular thread and not simply call `notifyAll()`. [6 marks]

(*b*) Object allocation graphs can be used to detect deadlock in a concurrent application.

(*i*) Give an example of an object allocation graph and explain the meanings of the different components. [2 marks]

(*ii*) Describe an algorithm which can use an object allocation graph and an object request matrix to determine whether or not deadlock exists. [5 marks]

(*iii*) Describe how to distinguish between cases in which deadlock has occurred and those in which deadlock is inevitable but is yet to occur. [1 mark]

SECTION B

5 Computer Graphics and Image Processing

(*a*) Give the definition of the cubic Bézier curve. [4 marks]

(*b*) Derive the conditions necessary to ensure that two cubic Bézier curves join with $C1$-continuity. [6 marks]

(*c*) Describe, in detail, an algorithm for drawing a cubic Bézier curve to a given tolerance using straight lines. You may assume that you already have an algorithm for drawing a straight line. [6 marks]

(*d*) Explain why and how homogeneous co-ordinates are used in computer graphics. [4 marks]

6 Compiler Construction

(*a*) Consider the grammar

$$S ::= (L) \mid a$$
$$L ::= L, S \mid S$$

(*i*) Present a right-most derivation for the string $(a,\ ((a,\ a),\ (a,\ a)))$. [3 marks]

(*ii*) Present a left-most derivation for the same string $(a,\ ((a,\ a),\ (a,\ a)))$. [3 marks]

(*b*) Automatic garbage collection is an important technique for the implementation of many programming languages. Define each of the following variations:

(*i*) Mark and Sweep; [3 marks]

(*ii*) Copy Collection; [3 marks]

(*iii*) Generational Collection. [3 marks]

(*c*) Write a small program that will produce different values depending on which kind of variable scoping mechanism is used, static or dynamic. Explain your answer. [5 marks]

7 Comparative Programming Languages

(*a*) A naïve programmer writes the following Prolog program to implement a quicksort.

```
quicksort( [], []).

quicksort( [X|Tail], Sorted)  :-
   split( X, Tail, Small, Big),
   append( SortedSmall, [X|SortedBig], Sorted),
   quicksort( Small, SortedSmall),
   quicksort( Big, SortedBig).

split( X, [], [], [X]).

split( X, [Y|Tail], [Y|Small], Big)  :-
   X>Y, !,
   split( X, Tail, Small, Big).

split( X, [Y|Tail], Small, [Y|Big])  :-
   split( X, Tail, Small, Big).
```

Unfortunately, there are two mistakes that will prevent it running as expected. What are these mistakes and how can they be corrected? [6 marks]

(*b*) Explain how the operator `!` in the `split` predicate works and why it is used here. [2 marks]

(*c*) Our programmer now decides to improve the efficiency of the program by using difference lists. Explain how the technique works and modify the program to use difference lists by introducing a new predicate `quicksort2`

```
quicksort( List, Sorted)  :- quicksort2( List, Sorted - [] ).
```

[6 marks]

(*d*) Comment on the space and time complexity of the execution of the two versions of quicksort for the call `quicksort([2,5,7],X)`. [6 marks]

8 Databases

(*a*) Define Boyce–Codd normal form. [3 marks]

(*b*) Suppose that a relation R has n attributes. How many distinct functional dependencies could be defined for R? [3 marks]

(*c*) The *union rule for functional dependencies* states that if $F \models X \to Y$ and $F \models X \to Z$, then $F \models X \to Y \cup Z$ (this can also be written as $F \models X \to Y, Z$).

Prove this rule using only Armstrong's axioms. [5 marks]

(*d*) *Heath's Theorem* states that if $R(A,\ B,\ C)$ satisfies the functional dependency $A \to B$, where A, B, and C are disjoint non-empty sets of attributes, then

$$R = \pi_{A,B}(R) \bowtie_A \pi_{A,C}(R),$$

where $\bowtie_A$ is the equi-join on the attributes of A. Prove this theorem. [9 marks]

SECTION C

9 Logic and Proof

(*a*) For each of the following equivalences, state whether it holds or not, justifying each answer rigorously.

(*i*) $((P \to Q) \to R) \to P \quad \simeq \quad P \vee (Q \wedge \neg R)$ [3 marks]

(*ii*) $(\forall x\, P(x)) \to (\exists x\, Q(x)) \quad \simeq \quad \exists x\, [P(x) \to Q(x)]$ [5 marks]

(*b*) For the following formula, present *either* a sequent calculus proof *or* a falsifying interpretation.

$$[\forall x\, \exists y\, P(x,y)] \to [\forall x\, P(x, f(x))]$$ [4 marks]

(*c*) For the following formula, in modal logic S4, present *either* a sequent calculus proof *or* a falsifying interpretation.

$$\Box(A \wedge \Diamond B) \to (\Box A \wedge \Diamond B)$$ [8 marks]

10 Foundations of Functional Programming

In this question you will be using a polymorphic type-reconstruction algorithm similar to that used in ML. Your answer will be expected to contain sufficient explanation of the overall method to explain how it applies in the particular cases considered.

(*a*) The fixed-point operator Y satisfies the equation $Yf = f(Yf)$. Follow through your general type deduction algorithm on this equation, and *either* deduce a polymorphic type for Y *or* show that one does not exist. [5 marks]

(*b*) The lambda-expression

$$\lambda f.(\lambda g.f(gg))(\lambda g.f(gg))$$

behaves as a fixed-point operator.

(*i*) Show that it satisfies the equation given in part (*a*). [5 marks]

(*ii*) Perform type-deduction on this lambda-expression and again either deduce a type for it or show that one cannot be found. [5 marks]

(*c*) In the context of type-derivation, explain relationships and differences between the following two lines of ML-style code:

```
let fun I x = x in (I 1, I 2, I 3) end;
(lambda I => (I 1, I 2, I 3)) (lambda x => x);
```

[5 marks]

11 Semantics of Programming Languages

(*a*) State *one* potential advantage of programming languages that do not have a static type system. [1 mark]

(*b*) Consider the following language syntax:

$$\begin{aligned} e \;::=\; & \textbf{skip} \mid b \mid n \mid \textbf{if}\ e_1\ \textbf{then}\ e_2\ \textbf{else}\ e_3 \mid \textbf{while}\ e_1\ \textbf{do}\ e_2 \mid \\ & \textbf{fn}\ x \Rightarrow e \mid e_1\ e_2 \mid x \mid \\ & \textbf{ref}\ e \mid e_1 := e_2 \mid !e \mid \ell \end{aligned}$$

where b ranges over the booleans $\{\textbf{true}, \textbf{false}\}$, n ranges over the natural numbers, and ℓ ranges over an infinite set of locations.

Design an operational semantics for this language that is well-defined and reasonable for arbitrary expressions (not just those that would be admitted by some static type system). Your semantics should:

1. involve clearly-specified notions of value v and store s;
2. define a small-step reduction relation $\langle e, s\rangle \longrightarrow \langle e', s'\rangle$;
3. be call-by-value; and
4. not be stuck for any configuration $\langle e, s\rangle$ where e is not a value.

Explain any parts of your definition that differ from those in the definition of a conventional typed language, such as the typed languages in the course notes. [15 marks]

(*c*) State property 4 precisely. [1 mark]

(*d*) Give an outline proof of property 4, including the form of induction used and one non-trivial case. [3 marks]

12 Complexity Theory

Suppose that $f(n)$ is a sensible function (you may like in some part of your answer to comment on what the term "sensible" might mean in this context), then show that the class $DTIME(f(n))$ is strictly contained in $DTIME(f(n)^4)$.

[20 marks]

END OF PAPER