COMPUTER SCIENCE TRIPOS Part II (General)
DIPLOMA IN COMPUTER SCIENCE

Wednesday 7 June 2006 1.30 to 4.30

PAPER 12 (PAPER 3 OF DIPLOMA IN COMPUTER SCIENCE)

Answer ***five*** *questions.*

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

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** $s \leadsto u \to v$ 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: $s \leadsto u$ = path from s to u consisting of 0 or more edges (0 when $s \equiv u$); $u \to v$ = 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 Distributed Systems

(*a*) Describe, with examples, the function of a naming service for a large-scale distributed system. Include definitions for "name space" and "naming domain". [6 marks]

(*b*) Discuss consistency *versus* availability for naming data in large-scale systems. [4 marks]

(*c*) How can any distributed naming service be engineered so that invocations on behalf of users can be resolved efficiently in the presence of failures and heavy load? [4 marks]

(*d*) Contrast the assumptions under which DNS was designed originally for the Internet, with the properties of dynamically formed groups of mobile hosts using wireless communication (MANETS). How might DNS-like services be provided for MANETS? [6 marks]

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 $F \models X \rightarrow Y$ and $F \models X \rightarrow Z$, then $F \models X \rightarrow Y \cup Z$ (this can also be written as $F \models X \rightarrow Y, Z$).

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 $A \rightarrow B$, 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]

9 Numerical Analysis II

(a) In Peano's theorem, if a quadrature rule integrates polynomials of degree N exactly over an interval $[a, b]$, then the error in integrating $f \in C^{N+1}[a, b]$ is expressed as

$$E(f) = \int_a^b f^{(N+1)}(t) K(t)\, dt$$

where

$$K(t) = \frac{1}{N!} E_x[(x - t)_+^N].$$

Explain the notation $E(f)$, E_x, $(x - t)_+^N$. [4 marks]

(b) Assuming $x \in [a, b]$, and writing Taylor's theorem in the form

$$f(x) = P_N(x - a) + \frac{1}{N!} \int_a^x f^{(N+1)}(t)(x - t)^N\, dt$$

where P_N is a polynomial of degree N, prove Peano's theorem, explaining each step clearly. [8 marks]

(c) For the trapezium rule, what is N? [1 mark]

(d) If $K(t)$ does not change sign in $[a, b]$ then

$$E(f) = \frac{f^{(N+1)}(\xi)}{(N + 1)!} E(x^{N+1})$$

for some $\xi \in (a, b)$. Use this result to simplify

$$E(f) = \int_{-1}^1 f(x)\, dx - f(-1) - f(1).$$

[7 marks]

10 Bioinformatics

(*a*) Why do we use dynamic programming algorithms for pairwise sequence alignment problems but not for multiple pairwise alignment? [5 marks]

(*b*) Compare the use of the affine gap penalty with the constant gap penalty. [3 marks]

(*c*) Discuss the properties and assumptions of the Jukes–Cantor and the Kimura 2-parameter models of DNA evolution. [5 marks]

(*d*) Describe the UPGMA algorithm. [4 marks]

(*e*) What does the ultrametric property of a tree tell us about the evolutionary process? [3 marks]

11 Software Engineering and Design

There are many tasks on software development projects that do not in themselves require programming experience, but where good software engineering practices are important. Typical examples are creation of system documentation in HTML, or design of user interface layouts.

(*a*) Describe *four* general-purpose software engineering tools that might improve the quality of even non-programming tasks. [12 marks]

(*b*) How would you go about the design of these tools in a user-centric manner? [8 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