COMPUTER SCIENCE TRIPOS Part IB

Monday 31 May 2010 1.30 to 4.30

COMPUTER SCIENCE Paper 3

*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 cover sheets
Tags

SPECIAL REQUIREMENTS
Approved calculator permitted

1 Algorithms II

(*a*) Describe Dijkstra's shortest-path algorithm, making the priority queue operations explicit. [4 marks]

(*b*) Provide a small example demonstrating that Dijkstra's shortest-path algorithm will not work correctly when negative weights are used on some arcs. [3 marks]

(*c*) Suppose some arcs in a directed graph have negative weights, and that $-W$ is the least negative weight among all arcs. Suppose that we add W to all arcs in the graph to obtain a new graph with non-negative arc weights. Will the resulting graph have the same shortest paths as the original graph? Explain your answer. [3 marks]

(*d*) For each of the data structures listed below, describe the computational complexity of Dijkstra's shortest-path algorithm when this data structure is used to implement the algorithm's priority queue. Justify your answers.

(*i*) An unsorted array, indexed by node number. [2 marks]

(*ii*) A linked list, sorted by key (in this case a distance estimate). [2 marks]

(*iii*) A binary heap. [2 marks]

(*iv*) A binomial heap. [2 marks]

(*v*) A Fibonacci heap. [2 marks]

2 Algorithms II

(*a*) Describe the basic operations on the disjoint-set data structure. [3 marks]

(*b*) Describe the simple linked-list implementation of the disjoint-set data structure, without the weighted-union or path compression optimisations. Explain the complexity of each operation. [6 marks]

(*c*) Describe Kruskal's algorithm for finding a minimum spanning tree. [6 marks]

(*d*) If the simple linked-list implementation (part (*b*)) is used to implement the disjoint-set data structure in Kruskal's algorithm, what is the resulting complexity? [5 marks]

3 Compiler Construction

(*a*) What are the principal features of a regular language and of a context-free language and their respective parsers? [4 marks]

(*b*) When implementing a compiler, why is a regular language commonly used for lexical analysis and why are context-free grammars commonly used for the main syntax analysis phase? [3 marks]

(*c*) Give **two** main features that distinguish a recursive-descent parser from an SLR(k) parser (or similar). [2 marks]

(*d*) The C++ language uses the "`>>`" character sequence to denote the right-shift operator. This character sequence can also appear when a template (generic) type takes another template as its argument, as in "`stack<list<int>>`" that is supposed to denote a stack of integer lists. What problem can this cause? [1 mark]

Comment on how you might solve this problem

(*i*) in the syntax analyser; and [2 marks]

(*ii*) in the lexical analyser. [2 marks]

Comment on the relative elegance of the two solutions. [1 mark]

(*e*) A common language construct is "`T v;`" to declare a variable called "`v`" or type "`T`". What parsing problem can arise if "`T`" is a user-defined type? Explain how the lexical analyser could benefit from feedback from the syntax analyser in this situation. [5 marks]

4 Compiler Construction

(*a*) How is inheritance of class variables typically implemented by a compiler for an object-oriented language in which each class has exactly one parent? [3 marks]

(*b*) When supporting multiple inheritance, what extra run-time complexity arises when casting or coercing an object handle up or down and why is this avoided using the Java concept of an interface? [3 marks]

(*c*) When supporting multiple inheritance, what identifier clash problem can arise and what are the possible solutions? [4 marks]

(*d*) Explain, using an example, the potential for an erroneous downcast in an object-oriented language such as Java, C++ or C#. [4 marks]

(*e*) Sketch assembly-level code that

(*i*) detects an erroneous downcast;

(*ii*) finds an exception handler for it; and

(*iii*) correctly jumps to the handler.

Assume that the handler is already registered with the run-time system and that there are no arguments to the exception. Use any well-known implementation techniques, such as keeping an object's identity in its virtual method table and stacking exception handlers on an extra run-time stack. [6 marks]

5 Concepts in Programming Languages

(*a*) List **two** differences between the programming languages FORTRAN and Pascal. [4 marks]

(*b*) List **two** differences between the programming languages SIMULA and Smalltalk. [4 marks]

(*c*) Explain what is understood by type checking and by type inference in the context of programming languages. Compare the two approaches, presenting their advantages and disadvantages. [4 marks]

(*d*) Give an example in the SML Modules language of a signature and of a structure. State whether or not the structure matches the signature, explaining your answer. [4 marks]

(*e*) The programming language Scala introduced *case classes*. Explain why they are useful in programming, giving code samples. [4 marks]

6 Programming in C and C++

(*a*) Popular programming journal *Obscure C Techniques for Experts* has published a novel way to save space for a doubly-linked list program. Instead of storing two pointers (one next and one previous), this new technique stores a single value: the `XOR` of *previous* and *next* pointers.

A traditional two-pointer linked list might be illustrated as:

$$\begin{array}{ccccccccccc} \cdots & \text{A} & & \text{B} & & \text{C} & & \text{D} & & \text{E} & \cdots \\ & & \rightarrow & \text{next} & \rightarrow & \text{next} & \rightarrow & \text{next} & \rightarrow & & \\ & & \leftarrow & \text{prev} & \leftarrow & \text{prev} & \leftarrow & \text{prev} & \rightarrow & & \end{array}$$

In contrast, the proposed new technique stores a bit-wise `XOR` of the *previous* and *next* pointers within a single field.

$$\begin{array}{ccccccccccc} \cdots & \text{A} & & \text{B} & & \text{C} & & \text{D} & & \text{E} & \cdots \\ & & \leftrightarrow & \oplus & \leftrightarrow & \oplus & \leftrightarrow & \oplus & \leftrightarrow & & \end{array}$$

You have been engaged to provide code examples of this approach for publication.

Ensure your code illustrates the creation and initialization of such a list as well as the insertion, and deletion, of elements from such a list. Additionally, you must provide examples of a forward or backward traversal of the list permitting examination of each element in turn. [15 marks]

(*b*) Comment on this form of linked list. Consider the comparative speed, memory overheads, maintenance and other advantages or disadvantages of the `XOR` doubly-linked list approach when compared with an approach that stores both previous and next pointers. [5 marks]

7 Prolog

(*a*) Consider the following clauses:

```
a(1).
a(a).
b(2).
b(3).
c(X,X) :- a(X).
c(X,Y) :- a(X),!,b(Y).
c(X,X) :- b(X).
```

List all of the possible solutions to the query `c(A,B)` in order. [2 marks]

(*b*) Consider the predicate `p/4` defined as:

```
p(_,[],[],[]).
p(P,[X|Xs],[X|L1],L2) :- X < P, p(P,Xs,L1,L2).
p(P,[X|Xs],L1,[X|L2]) :- X >= P, p(P,Xs,L1,L2).
```

(*i*) What does `p/4` do? Show an example query and response. [2 marks]

(*ii*) What types of terms cannot be used as the first `p/4` argument? [2 marks]

(*iii*) Describe where and why red and green cuts can be usefully employed within the `p/4` predicate, modifying the predicate if necessary. [3 marks]

(*iv*) The quicksort algorithm requires selection of a pivot value, forming two sublists, and then building the sorted list from the results of recursively quicksorting the sublists. The first sublist contains all elements that are less than the pivot value. The second contains the remaining elements.

Implement quicksort in a predicate `qs(+In,-Out)` that binds `Out` to a sorted version of the list `In`, and uses the `p/4` predicate above. Assume that `In` only contains numbers. Do not optimise your choice of pivot value: just use the head of the input list. [5 marks]

(*c*) The predicate `flatten(+In,-Out)` is defined to expect that `In` be unified with a list that may contain elements that are themselves lists (and likewise for those lists, up to any depth). In such cases, `Out` is unified with a list that never has elements that are lists—all lists are expanded in place. For example, `flatten([1,2,[3,4],[[5],6]],[1,2,3,4,5,6])` would be true.

Implement the `flatten/2` predicate. When given a valid query, your solution should not provide any spurious results. [6 marks]

8 Software Engineering

(*a*) Discuss the lessons learned from the London Ambulance Service disaster from the following viewpoints.

(*i*) Requirements engineering.

(*ii*) Human factors.

(*iii*) Testing.

(*iv*) Project management. [12 marks]

(*b*) You have been hired by the Department of Energy and Climate Change (DECC) to manage a project to replace Britain's 47 million gas and electricity meters with "smart" meters that report energy use every 30 minutes. These reports go to a central service, and energy companies have access to their customers' readings. The goals of the system are to facilitate more flexible pricing, so customer demand can be brought more into line with supply; to make it simpler for customers to switch energy companies; and to help DECC predict and manage energy demand.

Describe what measures you would take to reduce the likelihood of a project disaster. [8 marks]

9 Further Java

Fellows at Norisbon College dine at a circular table on which there is a single fork between each Fellow. Fellows either eat or think, and always start dinner thinking. To eat, a Fellow first picks up the fork immediately to his left and, once successful, picks up the fork immediately to his right. When a required fork is not on the table, the Fellow waits, neither eating nor thinking, until the fork is returned to the table. After eating, a Fellow returns both forks to the table. No cutlery is required to think.

Your task is to model the above scenario in Java.

(*a*) Write a class called `Fork` with two public methods, `pickUp` and `putDown`. The methods should take no arguments and return no result. An instance of `Fork` should act as a lock to prevent concurrent access. In other words, once `pickUp` has been called, all further calls to `pickUp` should block until `putDown` is called; when `putDown` is called, one caller (if any) who is blocked should proceed. [7 marks]

(*b*) Write a class called `Fellow` which inherits from the `Thread` class and implements the abstract method `run`. The `Fellow` class should have a single constructor which takes two `Fork` objects, one representing the fork to the Fellow's left, and one to the right. When run, an instance of `Fellow` should think for ten seconds, eat for ten seconds and think for ten seconds before terminating. [7 marks]

(*c*) Describe when and why your implementation may suffer deadlock. [2 marks]

(*d*) By altering the order in which the forks are picked up, describe how you would modify your implementation so that it does not suffer deadlock. [4 marks]

END OF PAPER