COMPUTER SCIENCE TRIPOS Part IB

Thursday 8 June 2006 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

STATIONERY REQUIREMENTS
Script Paper
Blue Coversheets
Tags

SPECIAL REQUIREMENTS
Answer sheet for Question 1(c)
Answer sheet for Question 1(d)

SECTION A

1 Data Structures and Algorithms

(*a*) Explain what the *heap* data structure is, state its defining properties and explain how to convert between the tree and vector representations of a heap. [2 marks]

(*b*) Describe an optimally efficient algorithm for transforming any random vector into a heap vector and explain why it works. [4 marks]

(*c*) Using the tree instead of the vector representation for clarity, apply this algorithm to the binary tree isomorphic to the letter vector "`P I S K T Z O P V N`", producing a frame-by-frame trace of the execution. For this answer, please use the sheet for part (*c*), supplied with the question paper, and use a new tree whenever any nodes change. [5 marks]

(*d*) Explain how to rearrange the heap after having extracted its top so that what remains is still a heap. Follow this procedure to extract the top three values, one by one, from the heap you built, producing a frame-by-frame trace as above. Please use the sheet for part (*d*). [5 marks]

(*e*) Describe a way to insert a new value into an existing heap in time $O(\log n)$ where n is the heap size. [4 marks]

Further copies of the sheets for parts (*c*) and (*d*) may be obtained from the Invigilator, if required. Ensure that you attach the sheets to the rest of your answer.

2 Computer Design

(*a*) Early processors used an accumulator to hold an intermediate result whereas more modern processors use a register file or a stack.

(*i*) Why use a stack or register file rather than an accumulator? [4 marks]

(*ii*) What are the advantages and disadvantages of using a register file rather than a stack? [6 marks]

(*b*) In 1946, Burks, Goldstein and von Neumann made the observation "Ideally one would desire an indefinitely large memory capacity such that any particular word would be immediately available. We are forced to recognize the possibility of constructing a hierarchy of memories, each of which has greater capacity than the preceding but which is less quickly accessible."

(*i*) Why is this memory design issue even more pressing today? [5 marks]

(*ii*) What architectural techniques are used today to approximate a large memory which is nearly instantaneously accessible? [5 marks]

3 Digital Communication I

(*a*) Describe and contrast the processes of (*i*) forward error correction and (*ii*) error detection with retransmission. [5 marks]

(*b*) What properties should be considered when deciding which should be used to control errors? [10 marks]

(*c*) Are there circumstances when both should be used? Justify your answer. [5 marks]

4 Concurrent Systems and Applications

(*a*) Java's *Reference Objects* provide a means to interact with the runtime garbage collector.

(*i*) Provide a Java class implementing a *Leaky Array*: a fixed-size, ordered sequence of `Objects`, indexed by integers from zero upwards. The items may be discarded (individually) by the garbage collector when the system is running low on memory. Provide concurrency-safe accessor methods to get and set the items stored in the array. Ensure that any internal data structures cannot be manipulated other than via your accessor methods. It should be possible for classes in other packages to construct new Leaky Array objects and be able to invoke the accessor methods. Derived classes should have concurrency-safe means of determining the size of the array (but be unable to change it) and counting the number of not-null items stored. [6 marks]

(*ii*) Describe carefully when a `Finalizer` method defined on a Java class will be executed. Might two finalizers be executed concurrently? What guarantees does Java make about the execution of finalizers? [5 marks]

(*iii*) How can a `ReferenceQueue` be used to provide more control than finalizer methods? [4 marks]

(*b*) Does Java's garbage collector clean up memory allocated by code in a *native method*? What facilities exist to specify whether or not Java objects created by native code are eligible for garbage collection? [5 marks]

SECTION B

5 Computer Graphics and Image Processing

(*a*) Standard colour printing uses the four ink colours: cyan, magenta, yellow and black.

(*i*) Explain why this is so. [3 marks]

(*ii*) What benefits are there in using more than these four ink colours? [3 marks]

(*b*) Describe an algorithm for converting a greyscale image to a bilevel (black and white) image while retaining as good a quality as possible. Your algorithm should be for the situation where each greyscale pixel maps to a single bilevel pixel. [6 marks]

(*c*) Describe operations on images which achieve the following effects:

(*i*) lighten an image which is too dark; [2 marks]

(*ii*) remove salt and pepper noise ("shot noise") from an image; [2 marks]

(*iii*) locate 45° edges in an image; [2 marks]

(*iv*) convert a colour image (in RGB format) to a greyscale image while preserving the perceived luminance. [2 marks]

6 Compiler Construction

(*a*) Describe a difference and a similarity between the notions of *overloading* and *polymorphism*. [2 marks]

(*b*) Define the notion of *type safety* in a programming language. [2 marks]

(*c*) Describe the linking phase and the difference between static and dynamic linking. [2 marks]

(*d*) Suppose that a programming language allows nested functions. How might a stack-based implementation of `f` access the value associated with `a`?

```
int g(int a)
{
    int f(int x) {
      return a + x;
    }
    ...
    f(2);
    ...
}
```

[4 marks]

(*e*) Describe the low-level code that could be generated by compiling the following code fragment for a stack-based target machine.

```
int f(int a, int b)
{
  int z = a * b;
  return a + z;
}
...

  f(1, 2) * f(3, f(4, 5));

...
```

[10 marks]

7 Comparative Programming Languages

(*a*) An author writes:

> Most successful language design efforts share three important characteristics . . .
>
> 1. Motivating Application: The language was designed so that a specific kind of program could be written more easily.
>
> 2. Abstract Machine: There is a simple and unambiguous program execution model.
>
> 3. Theoretical Foundations: Theoretical understanding was the basis for including certain capabilities and omitting others.

Briefly discuss the merits and/or shortcomings of *one* of the above three statements of your choice, giving examples and/or counterexamples from procedural, applicative, logical, and/or object-oriented programming languages. [6 marks]

(*b*) For *two* programming languages of your choice amongst FORTRAN, Algol, Pascal and C, briefly discuss and evaluate their typing disciplines. Further compare the advantages and disadvantages that their designs impose on the programmer. [5 marks]

(*c*) Consider the following two program fragments.

```
( defvar x 1 )
( defun g(z) (+ x z) )
( defun f(y)
  (+ (g 1)
     ( let
        ( ( x (+ y 3) ) )
        (
           g(+ y x)
           ) ) ) )
( f 2 )
```

```
val x = 1 ;
fun g(z) = x + z ;
fun f(y)
  = g(1) +
    let
       val x = y + 3
    in
       g(y+x)
    end ;
f(2) ;
```

What are their respective output values when run in their corresponding interpreters? Justify your answer, explaining it in a conceptual manner. [4 marks]

(*d*) Outline the key features that a language must have to be called object-oriented. Further, briefly discuss to what extent *one* programming language of your choice amongst Simula, Smalltalk, C++, and Java has them. [5 marks]

8 Databases

Suppose we have the following relational schema

Person(<u>pid:integer</u>, name:string, street:string, postcode:string)
Car(<u>cid:integer</u>, year:integer, model:string)
OwnedBy(<u>pid:integer, cid:integer</u>)
AccidentReport(<u>rid:integer</u>, damage:integer, details:string)
ParticipatedIn(<u>pid:integer, rid:integer, cid:integer</u>)

where the underlined attributes represent the primary keys of the associated relation. The table *OwnedBy* implements a relationship between persons and cars using foreign keys. The table *ParticipatedIn* implements a relationship between persons, accident reports, and cars, where tuple $(p,\ r,\ c)$ indicates that the person p was the driver of the car c associated with the accident report r.

(*a*) Write an SQL query to return those pid's of persons driving in at least one accident, with no duplicates. [2 marks]

(*b*) Write an SQL query to return all tuples $(pid,\ c)$, where c is the number of cars owned by person pid (records where $c = 0$ do not have to be generated). [2 marks]

(*c*) Write an SQL query to return all tuples $(cid,\ c)$, where c is the number of persons owning car cid (records where $c = 0$ do not have to be generated). [2 marks]

(*d*) Write a (nested) SQL query to return all tuples $(pid,\ rid)$ where *pid* was driving in the accident reported in *rid*, but the car driven by *pid* is not owned by *pid*. [4 marks]

(*e*) Write an SQL query to return all tuples $(rid,\ c)$, where c is the number of drivers involved in the accident reported in by rid (records where $c = 0$ do not have to be generated). [2 marks]

(*f*) Write an SQL query to return all tuples $(rid,\ c)$, where c is the number of cars involved in the accident reported in by rid (records where $c = 0$ do not have to be generated). [2 marks]

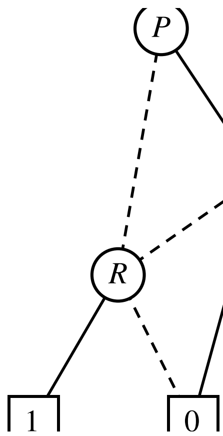
(*g*) Do the functional dependencies implied by the schema imply that the results of queries (*e*) and (*f*) will always be the same? Explain. [2 marks]

(*h*) Perhaps there is something wrong with this schema. How would you fix the schema to ensure that results of queries (*e*) and (*f*) would always be the same? [4 marks]

SECTION C

9 Logic and Proof

(*a*) Exhibit a formula that is logically equivalent to the following BDD, justifying your answer. [3 marks]



(*b*) Consider the following set of four clauses, where the variables are x, y and z:

$$\{P(x, z), P(y, x), Q(f(z))\}$$
$$\{\neg P(x, a), \neg P(a, x), Q(x)\}$$
$$\{\neg Q(x)\}$$
$$\{\neg Q(b)\}$$

(*i*) What is the Herbrand universe of these clauses? [4 marks]

(*ii*) Exhibit a Herbrand model satisfying these clauses, or prove that none exists. [13 marks]

10 Foundations of Functional Programming

(*a*) What does the combinator expression `S S S S S S` reduce to? Explain your working carefully. [4 marks]

(*b*) What would you get if you had a sequence of n `S` combinators (part (*a*) is the case $n = 6$)? [5 marks]

(*c*) If you start with a sequence of `K` combinators of general length n, as in the expression `(K K K K K K)` that arises when $n = 6$, what will the expression reduce to? [3 marks]

(*d*) Now what about sequences that start `S K S K S K` in cases where n instances of `S` alternate as shown with n of `K`? You should certainly include in your answer a tabulation of results for some small values of n. [8 marks]

11 Semantics of Programming Languages

Let L be the language with syntax below, a call-by-value left-to-right operational semantics, and the standard simple type system.

$$\begin{array}{rcl} e & ::= & n \mid \textbf{fn}\ x{:}T \Rightarrow e \mid e_1\ e_2 \mid x \\ T & ::= & \mathsf{int} \mid T_1 \rightarrow T_2 \\ n & \in & \mathbb{Z} \end{array}$$

(*a*) L is not Turing-complete: there are computable functions over the integers that are not expressible as closed L expressions of type $\mathsf{int} \rightarrow \mathsf{int}$. Why not? [2 marks]

(*b*) Define a modest extension L′ of L that is Turing-complete. Give the additional syntactic forms (for expressions and types), describe their operational semantics informally, and state their precise typing rules. [10 marks]

(*c*) Assuming that f is an L′ expression of type $\mathsf{int} \rightarrow \mathsf{int}$, give an expression e of type int that computes the smallest x such that $f\ x$ is zero. Explain how this can be used to prove completeness. [4 marks]

(*d*) Discuss whether Turing-completeness is a necessary or sufficient property for a good programming language. [4 marks]

12 Complexity Theory

(*a*) Suppose that you were provided with a black box that could accept the language of sentences describing an integer k and a graph G with a $k-$clique, and the black box accepted such languages in polynomial time. Explain how you could derive a process that would accept satisfiable instances of the problem 3-SAT in polynomial time. [10 marks]

(*b*) Suppose instead you had been provided with a black box that provided a polynomial-time acceptor for 3-SAT. Explain how you could use that to derive an efficient acceptor for the clique problem. [10 marks]

In your explanation the level of detail you are expected to give should be tuned to the level of complication in any transformations that you need to describe: simple ones should be described and justified in detail while elaborate or messy ones can be sketched and standard results quoted.

END OF PAPER