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

Tuesday 6 June 2006 1.30 to 4.30

PAPER 11 (PAPER 2 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.*

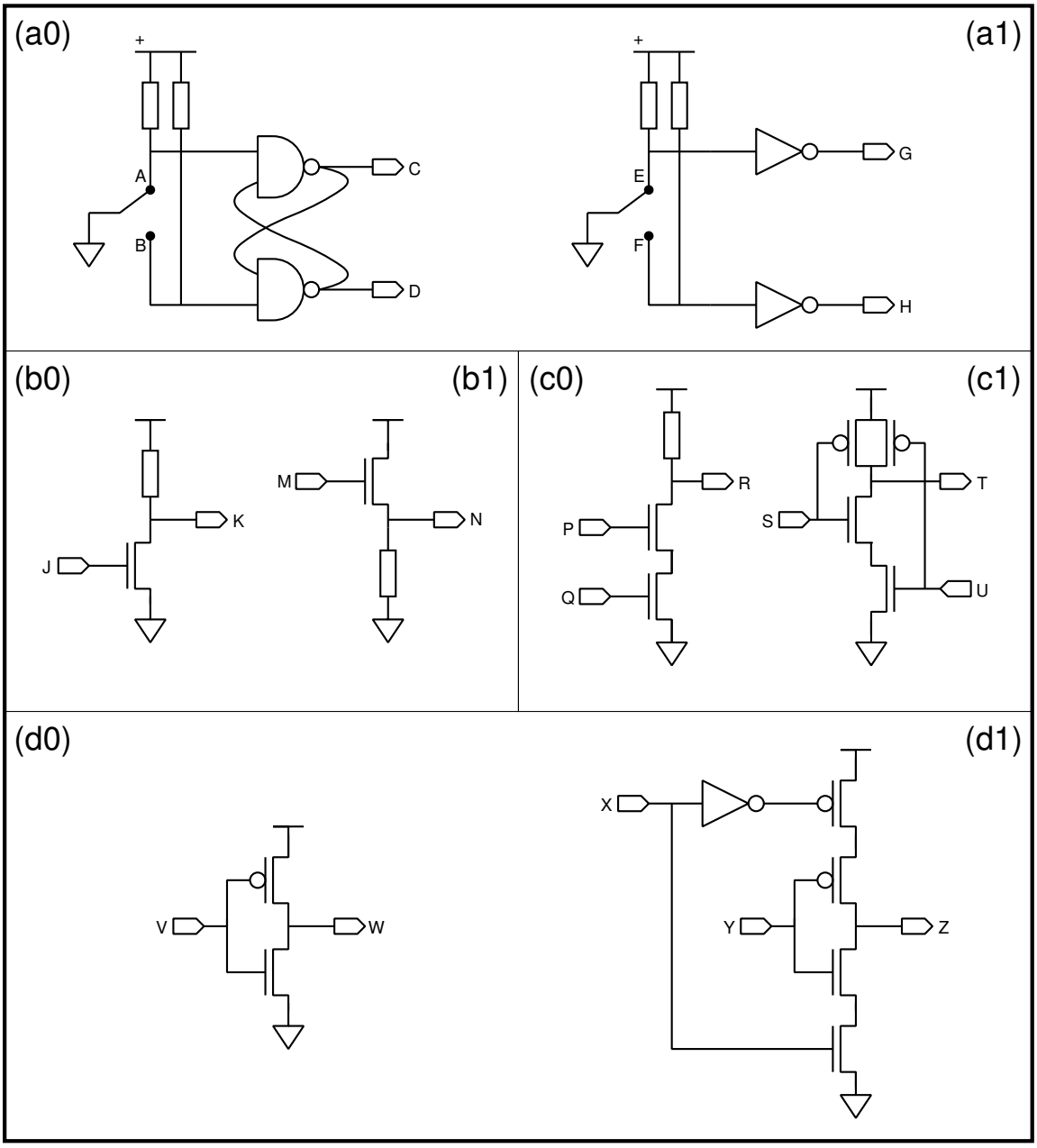
You may not start to read the questions printed on the subsequent pages of this question paper until instructed that you may do so by the Invigilator

STATIONERY REQUIREMENTS
Script Paper
Blue Coversheets
Tags

1 Digital Electronics

Four pairs of circuits (a0, a1), (b0, b1), (c0, c1) and (d0, d1) are presented below. What is the operation of each circuit and how does it compare with its pair?

[5 marks each]



2 Digital Electronics

(a) An electronic die may be constructed from seven LEDs laid out in the pattern below. The LEDs are to be driven by signals (a,b,c,d).

a		c
b	d	b
c		a

A binary-to-die decoder is described in the left-hand table below with inputs (n2,n1,n0) and outputs (a,b,c,d). X represents *don't care*.

(i) What are the minimum sum-of-product equations mapping the inputs to the outputs? [4 marks]

(ii) If the inputs to the decoder were to be driven by a three D flip-flop state machine, what are the minimum sum-of-products equations for the next state functions for (n2,n1,n0) to count continuously $1, 2, 3, 4, 5, 6, 1, \ldots$? [6 marks]

(b) An alternative implementation is to use a 1-hot state machine plus a different decoder to form a rolling die (see right-hand table below). The states are (h1,h2,h3,h4,h5,h6) and the die output this time is (A,B,C,D).

(i) What is the minimal free running 1-hot state machine constructed from D flip-flops? You may assume that the D flip-flops have preset and clear inputs. [3 marks]

(ii) What are the minimum sum-of-product equations for mapping the 1-hot states to die outputs? [4 marks]

(iii) Is the first implementation in part (a) quicker or slower than the one in part (b)? [3 marks]

binary to die decoder

input			output			
n2	n1	n0	a	b	c	d
0	0	0	X	X	X	X
0	0	1	0	0	0	1
0	1	0	1	0	0	0
0	1	1	1	0	0	1
1	0	0	1	0	1	0
1	0	1	1	0	1	1
1	1	0	1	1	1	0
1	1	1	X	X	X	X

1-hot to die decoder

input						output			
h6	h5	h4	h3	h2	h1	A	B	C	D
0	0	0	0	0	1	0	0	0	1
0	0	0	0	1	0	1	0	0	0
0	0	0	1	0	0	1	0	0	1
0	0	1	0	0	0	1	0	1	0
0	1	0	0	0	0	1	0	1	1
1	0	0	0	0	0	1	1	1	0

3 Compiler Construction

Consider the following grammar.

$$E ::= \text{NUM} \mid E \times E \mid E/E \mid E + E \mid E - E \mid (E)$$

(*a*) Show that this grammar is ambiguous. [2 marks]

(*b*) Rewrite the grammar to eliminate ambiguity. [4 marks]

(*c*) Transform the grammar of part (*b*) into an equivalent one that could be used for recursive descent parsing. Explain your answer. [5 marks]

(*d*) Write a recursive descent parser for your grammar. [9 marks]

4 Business Studies

(*a*) Name *five* different types of intellectual property. [5 marks]

(*b*) Distinguish between "Deep linking" and "Direct linking". Can a search engine deep link without infringing the copyright of the original site? [5 marks]

(*c*) Why is the use of thumbnails of pictures by a search engine fair use? [5 marks]

(*d*) Why are there likely to be only a few dominant search engines? [5 marks]

5 Artificial Intelligence I

(*a*) Give a brief description of the way in which *if-then rules* can be used as a basis for knowledge representation and reasoning. What essential elements would you expect to be included in such a system? [3 marks]

(*b*) In the context of such a system, give detailed descriptions, illustrating your answers with specific examples, of the following concepts:

(*i*) pattern matching; [2 marks]

(*ii*) reason maintenance; [2 marks]

(*iii*) forward chaining; [4 marks]

(*iv*) conflict resolution strategies; [4 marks]

(*v*) backward chaining with backtracking. [5 marks]

6 Operating System Foundations

(*a*) (*i*) Define a scheduling algorithm based on multilevel feedback queues. [6 marks]

(*ii*) Discuss the properties of your algorithm. Where appropriate, outline different design decisions that could have been taken. [6 marks]

(*b*) Discuss the implications of an operating system supporting

(*i*) single-threaded processes only;

(*ii*) multi-threaded processes. [8 marks]

7 Numerical Analysis I

(a) The Newton–Raphson iteration for solution of $f(x) = 0$ is

$$\tilde{x} = x - \frac{f(x)}{f'(x)}.$$

By drawing a carefully labelled graph, explain the graphical interpretation of this formula. What is the order of convergence? [4 marks]

(b) Consider $f(x) = x^3 + x^2 - 2$. The following table shows successive iterations for each of the three starting values (i) $x = 1.5$, (ii) $x = 0.2$, (iii) $x = -0.5$. Note that, to the accuracy shown, each iteration finds the root at $x = 1$.

n	(i)	(ii)	(iii)
0	1.50000×10^{0}	2.00000×10^{-1}	-5.00000×10^{-1}
1	1.12821×10^{0}	3.95384×10^{0}	-8.00000×10^{0}
2	1.01152×10^{0}	2.57730×10^{0}	-5.44318×10^{0}
3	1.00010×10^{0}	1.70966×10^{0}	-3.72976×10^{0}
4	1.00000×10^{0}	1.22393×10^{0}	-2.56345×10^{0}
5	1.00000×10^{0}	1.03212×10^{0}	-1.72202×10^{0}
6		1.00079×10^{0}	-9.62478×10^{-1}
7		1.00000×10^{0}	1.33836×10^{0}
8		1.00000×10^{0}	1.06651×10^{0}
9			1.00329×10^{0}
10			1.00000×10^{0}
11			1.00000×10^{0}

Sketch the graph of $f(x)$ and show the first iteration for cases (i) and (ii) to show why (i) converges faster than (ii). In a separate sketch, show the first two iterations for case (iii). [Hint: a very rough sketch will suffice for case (iii).] [10 marks]

(c) Now consider $f(x) = x^4 - 3x^2 - 2$. Calculate two Newton–Raphson iterations from the starting value $x = 1$. Comment on the prospects for convergence in this case. [6 marks]

8 Mathematics for Computation Theory

(*a*) What is a *deterministic finite automaton* (DFA) over the finite alphabet Σ? [2 marks]

(*b*) Define the *event E accepted by* the DFA M over Σ. [3 marks]

(*c*) State Kleene's Theorem, which characterises the algebraic structure of events that are accepted by some DFA. [3 marks]

(*d*) Design a DFA over $\Sigma = \{a, b\}$ that accepts precisely those strings ending in b that do not contain two successive occurrences of a. [4 marks]

(*e*) Give an algebraic specification of the event, proving that your expression has the required properties. [8 marks]

[You may if you wish assume that if $M = \begin{pmatrix} A & B \\ C & D \end{pmatrix}$ is a partitioning of the transition matrix of a DFA so that A and D are square, then

$$M^* = \begin{pmatrix} (A + BD^*C)^* & A^*B(D + CA^*B)^* \\ D^*C(A + BD^*C)^* & (D + CA^*B)^* \end{pmatrix}$$

with the same partitioning.]

9 Computation Theory

(*a*) Define the collection of *primitive recursive* functions. [6 marks]

(*b*) Why is a primitive recursive function always total? [1 mark]

(*c*) Show that the function m from $\mathbb{N}^2$ to $\mathbb{N}$ given by

$$m(x, y) = \begin{cases} x - y & \text{if } x \geq y \\ 0 & \text{if } x < y \end{cases}$$

is primitive recursive. [3 marks]

(*d*) Define the collection of *partial recursive* functions. [3 marks]

(*e*) What is meant by a *total recursive* function? [1 mark]

(*f*) Show that there exist total recursive functions that are not primitive recursive. Any standard results about register machines or recursive functions that you use need not be proved, but should be clearly stated. [6 marks]

10 Introduction to Security

(*a*) Alice and Bob participate in a public-key infrastructure that enables them to exchange legally binding digital signatures.

(*i*) Name *two* reasons why, for some purposes, Alice might prefer to use a message authentication code, instead of a digital signature, to protect the integrity and authenticity of her messages to Bob. [4 marks]

(*ii*) Outline a protocol for protecting the integrity and authenticity of Alice's messages to Bob that combines the benefits of a public-key infrastructure with those of using a message authentication code. [4 marks]

(*b*) Your colleague proposes a new way for constructing a message authentication code using a block cipher $E : \{0,1\}^{64} \times \{0,1\}^{128} \to \{0,1\}^{128}$. He takes the n-bit input message M, appends $p = 64 \cdot \lceil n/64 \rceil - n$ zero-bits, and splits the result into $k = (n+p)/64$ 64-bit blocks $M_1||M_2||\ldots||M_k = M||0^p$. He then calculates the message authentication code as

$$C_K(M) = E_{M_1}(E_{M_2}(E_{M_3}(\ldots E_{M_k}(K)\ldots)))$$

where K is the 128-bit secret key shared between sender and recipient. Show *two* different ways in which an attacker who observes a pair $(M, C_K(M))$ can, without knowing K, create a new pair $(M', C_K(M'))$ with $M' \neq M$. [6 marks]

(*c*) Show how a 128-bit message authentication code $C_K(M)$ with 64-bit key K can be constructed for an n-bit long message M using

(*i*) a secure hash function $H : \{0,1\}^* \to \{0,1\}^{256}$, such as SHA-256; [2 marks]

(*ii*) a block cipher $E : \{0,1\}^{128} \times \{0,1\}^{256} \to \{0,1\}^{256}$. [4 marks]

11 Natural Language Processing

The following shows a simple context free grammar (CFG) for a fragment of English.

```
S -> NP VP              Adj -> angry          Vbe -> is
VP -> Vbe Adj           Adj -> big            N -> dog
NP -> Det N             Adj -> former         N -> cat
N -> Adj N              P -> at
Adj -> Adj PP           P -> on
PP -> P NP              Det -> the
```

(*a*) Show the parse tree that this grammar would assign to (1).

(1) the dog is angry at the cat

[3 marks]

(*b*) One respect in which this grammar overgenerates is that some adjectives, including *former*, occur only before a noun (see (2)) and that PPs do not combine with adjectives occurring before a noun (see (3)).

(2) * the dog is former

(3) * the angry at the cat dog is big

Show how the grammar given above could be modified to prevent this type of overgeneration. [4 marks]

(*c*) The grammar also behaves incorrectly with examples (4), (5) and (6):

(4) * the dog is big at the cat (*big* does not take a PP)

(5) * the dog is angry on the cat (*angry* only takes PPs where the P is *at*)

(6) * the dog is angry at the cat at the cat (adjectives may not combine with multiple PPs)

Show modifications to the grammar which would prevent these types of overgeneration. [5 marks]

(*d*) Describe how the overgeneration in part (*c*) could be dealt with in a feature structure (FS) grammar, giving full lexical entries for *angry* and *big* and details of rules and other lexical entries as necessary to explain your account.

[8 marks]

12 Introduction to Functional Programming

Let `datatype 'a bintree = empty | node of 'a * 'a bintree * 'a bintree` be the polymorphic data type representing binary trees.

(*a*) Define the curried function `treemap` that given an `'a -> 'b` function and then an `'a bintree` produces a `'b bintree` of exactly the same shape as the given one but with all nodes having been *mapped* by the function. [4 marks]

(*b*) Use `treemap` to define the function `lift` that converts an `'a bintree` into an `'a option bintree`. Further use `treemap` to define the curried functional `prefilter` that given a predicate `'a -> bool` and then an `'a bintree` returns an `'a option bintree` in which the information on the nodes that do satisfy the predicate is omitted. [3 marks]

(*c*) Define the function

```
('a, 'b) DFfoldRtoL : ('a * 'b -> 'b) -> 'b -> 'a bintree -> 'b
```

that *folds* a binary tree (using its second input as base case and its first input in each recursive step) as the tree is traversed in a *depth-first* manner from *right to left*.

Your definition should be such that the function

```
fun inorder t = DFfoldRtoL op:: [] t
```

lists a binary tree in *infix order*. [5 marks]

(*d*) Use `DFfoldRtoL` to define the function

```
'a inorderData : 'a option bintree -> 'a list
```

that lists *the data* in its input tree in infix order. [2 marks]

(*e*) Define a function

```
'a cleanup : 'a option bintree -> 'a bintree
```

satisfying the specification `inorder`(`cleanup` t) = `inorderData`(t) for all types α and values t of type α `option bintree` and such that `cleanup`(`lift` t) = t for all types α and values t of type α `bintree`. [6 marks]

Note that `cleanup` and `prefilter` can be composed to yield a *tree-filter* functional that preserves the infix order.

END OF PAPER