Birkbeck
(University of London)

MSc Examination

Department of Computer Science and Information Systems

Fundamentals of Computing (COIY058H7)

CREDIT VALUE: 15 credits

Date of Examination: Monday, 22 May 2017
Time of Examination: 14:30 – 17:30
Duration of Paper: Three hours

*This paper is split into **two** sections.*
There are 50 marks for each section.
There are 5 questions in Section A and 5 questions in Section B.

Use a separate answer book for each section.

Answer all questions.
Start each answer on a new page.

The use of electronic calculators is not permitted.

The breakdown of marks per question is as follows:

Question	1	2	3	4	5	6	7	8	9	10
Marks	10	9	10	11	10	11	10	10	10	9

Section A

1. (a) Construct the truth-table for the Boolean function given by the formula

$$\neg\big((A \to \neg B) \wedge (C \to A)\big).$$ **(3 marks)**

(b) Find a Boolean circuit with AND, OR and NOT gates only that computes the Boolean function in (a) above and contains as few gates as possible. **(4 marks)**

(c) Determine whether the formula in (a) is equivalent to the formula

$$(B \to \neg A) \to (C \wedge \neg A).$$

Show your working. **(3 marks)**

2. (a) Represent -101_{10} as a 32-bit two's complement binary number. Show your working. **(3 marks)**

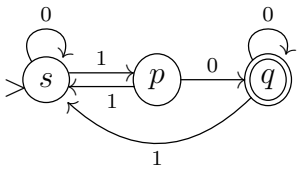
(b) Represent -17.125_{10} as an IEEE 754 32-bit floating-point number. Show your working. **(3 marks)**

(c) Find the decimal number represented by the 32-bit word

1100 0001 0101 0000 0000 0000 0000 0000

assuming it is a single precision IEEE 754 floating-point number. **(3 marks)**

3. (a) What is the language recognised by the following deterministic finite automaton?



– Describe the language in English. **(2 marks)**

– Describe the language by means of a regular expression. **(3 marks)**

(b) Design a deterministic finite automaton A such that $L(A)$ consists of all strings over the alphabet $\{0, 1\}$ in which every occurrence of 0 is immediately followed by a 1. **(3 marks)**

Give a regular expression for $L(A)$. **(2 marks)**

4. (a) Give a context-free grammar for the language over the alphabet $\{0, 1\}$ containing

– all words of even length **(3 marks)**

– all words of odd length **(3 marks)**

(b) What is the language defined by the following context-free grammar?

$S \to S_1,\ S \to S_2,\ \ S_1 \to 0S_1,\ S_1 \to 0A,\ \ S_2 \to S_2 1,\ S_2 \to A1,\ \ A \to \varepsilon,\ A \to 0A1.$

(5 marks)

5. (a) Give an implementation level description in English of a Turing machine computing the function $f : \mathbb{N} \to \mathbb{N}$ given by $f(n) = 8 \times n + 2$. **(3 marks)**

(b) Give a transition table of your Turing machine and the computation of this machine for $n = 0$. **(7 marks)**

Section B

6. In addition to the usual stack operations, *concatenable stacks* also support the operation of *concatenating* two stacks. So, if $S1$ and $S2$ are concatenable stacks, we have the usual operations: $Stack(S1, y)$, $Unstack(S1)$, $Stack(S2, y)$, $Unstack(S2)$, and also $Concat(S1, S2)$ and $Concat(S2, S1)$.

   The effect of $Concat(S1, S2)$ is to put the elements of $S2$ at the bottom of $S1$ and make $S2$ empty. For example, if $S1$ and $S2$ are initially empty, then after the sequence of operations:

   $Stack(S1, 1)$, $Stack(S1, 2)$, $Stack(S1, 3)$, $Stack(S2, 4)$, $Stack(S2, 5)$, $Stack(S2, 6)$,

   $Concat(S1, S2)$,

   the contents of $S1$ will be $[3, 2, 1, 6, 5, 4]$, where the top element is 3, and $S2$ will be empty.

   (a) Suggest a suitable representation for a concatenable stack. **(2 marks)**

   (b) Give algorithms for the operations $Stack(S1, y)$, $Unstack(S1)$ and $Concat(S1, S2)$ for the representation you have suggested. **(9 marks)**

7. (a) Draw the *binary search tree* B obtained by inserting the following strings into an empty tree in the order:

   $$fly,\ sow,\ ape,\ dog,\ hen,\ cod,\ cow,\ ant,\ pig,\ ram,\ bat,\ owl.$$

   Add in the threads to turn B into a *threaded* binary tree with a *head node*. **(3 marks)**

   (b) What are the *pre-order*, *in-order* and *post-order predecessors* and *successors* in B of *ape* and *cow*? **(3 marks)**

   (c) Draw the *forest* F corresponding to B using the *natural correspondence* between forests and binary trees. **(2 marks)**

   (d) What implications does the fact that B is a binary *search* tree have for the relationship between the string values at the nodes of F? **(2 marks)**

8. (a) A programmer tries to modify the standard non-recursive algorithm for *pre-order* traversal of a binary tree using a stack so that the traversal is in *in-order*. He produces the following algorithm:

```
PointerStack S       //S is initially empty
while (true)        //initially P points to the root
  if (P ≠ nil)
  {   Stack(S,P)
      P ← P↑Llink
      Visit(P)
  }
  else
  {  if Empty(S) return
      P ← Unstack(S)
      P ← P↑Rlink
  }
```

Show that this does not work as intended by tracing the execution of the algorithm on the (unthreaded) binary tree B that you constructed in Question 7 (a), and write down the order in which the nodes are visited.

What modification should the programmer have made so that the nodes are visited in *in-order*? **(5 marks)**

(b) In a *threaded* binary tree, node B is the *in-order successor* of node A, that is, $B = insucc(A)$. Draw diagrams to illustrate the two possible relationships between A and B.

By mistake, the tag fields of B may not have not been set correctly, although all the threads and the tags of all the nodes other than B are set correctly. Modify the code of the function $insucc(.)$ so that it correctly finds the in-order successor of node A (although it may not work correctly for node B). **(5 marks)**

9. (a) At 1 January 2010 (time $t1$) a new file of 40 million records was constructed and held on disk. It would not be completely reconstructed until 31 December 2010. By 30 November 2010 (time $t2$), there had been many insertions and deletions, but more insertions, so that the file then contained nearly 80 million records.

Copy the following table and write A, B or C in each of the cells to indicate whether for the specified method of file organisation and pattern of access:

A – Access is good at $t1$ and still pretty good at $t2$

B – Access is good at $t1$ but markedly worse at $t2$

C – Access is poor even at $t1$

	Key-ordered sequential access	Random access
Key-ordered sequential		
Indexed-sequential		
Hashed/random		
B-tree		

(4 marks)

(b) A B-tree of order 6 with a maximum of 5 records per data page currently has 3 levels. The keys of the records are three-digit positive integers, e.g. 037.

What are the *minimum* and *maximum* number of records that there could currently be in the B-tree? Draw a possible B-tree $B1$ with the *minimum* number of records.

Select some record in $B1$ and draw the resulting B-tree $B2$ after this record has been deleted from $B1$. For $B2$, you should include the keys on all pointer pages, but only need include the keys on those data pages that are changed after the deletion. **(6 marks)**

10. (a) You are given an unordered array of n, not necessarily distinct, positive integers $a[1], a[2], \ldots a[n]$. Give an algorithm that returns some duplicate element, i.e. some integer that occurs in the array more than once, or returns 0 if all n elements are distinct. Your algorithm must not change the array and may only use a constant amount of additional space, i.e. not dependent on n. State the order of magnitude of the time complexity of your algorithm using O(.) notation. **(5 marks)**

(b) Now suppose that your algorithm may change the array and may use any amount of additional space. Describe, without writing any code, how you could construct an algorithm to solve this problem that would run significantly faster than your original algorithm. What would the time complexity of this new algorithm be?
(4 marks)

