Birkbeck
(University of London)

MSc Examination

Department of Computer Science and Information Systems

Fundamentals of Computing (COIY058H7)

CREDIT VALUE: 15 credits

Date of Examination: Friday, 12 June 2015
Time of Examination: 10:00 – 13:00
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.

Start each section in a separate answer book.

Answer all questions.
Use a separate answer book for each section.

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	10	10	10	10	10

Section A

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

$$F \;=\; (\neg A_2 \to A_3) \to (A_1 \wedge (A_2 \vee A_3)).$$ **(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 $\neg F$ is equivalent to

$$(A_1 \vee A_2) \to A_3.$$

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

2. (a) Represent -102 as a 32-bit two's complement binary number. Show your working. **(3 marks)**

   (b) Represent 9.625 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 1011 0000 0000 0000 0000 0000

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

3. (a) Design a deterministic finite automaton A such that $L(A)$ consists of all strings over the alphabet $\{a, b\}$ that do not contain the substring aaa. **(5 marks)**

   (b) Find a regular expression representing the language over the alphabet $\{a, b\}$ that consists of strings where the number of b's can be evenly divided by 3. **(5 marks)**

4. (a) Give a context-free grammar for the language over $\{0, 1\}$ containing all words with the substring 00 or the substring 11. Is this language regular? Explain your answer. **(7 marks)**

   (b) What is the language over $\{a, b\}$ defined by the following context-free grammar with start variable S? **(3 marks)**

$$S \to aSa, \quad S \to bTb, \qquad T \to bT, \quad T \to b$$

   Is this language regular? Give an informal explanation of your answer. **(1 marks)**

5. Consider the following $\mathbb{N} \to \mathbb{N}$ function f:

$$f(n) = \begin{cases} 2n+1 & \text{if } n \text{ is even,} \\ 2n-2 & \text{if } n \text{ is odd.} \end{cases}$$

   Representing numbers in binary:

   (a) give an implementation level description in English of a Turing machine that computes the function f; **(5 marks)**

   (b) give the complete transition table of this Turing machine. **(5 marks)**

Section B

6. (a) What representation would you choose for:
   (i) an *output*-restricted deque when the maximum number of items
   – is ***not*** known in advance?
   – ***is*** known in advance?
   (ii) an *input*-restricted deque when the maximum number of items
   – is ***not*** known in advance?
   – ***is*** known in advance? **(3 marks)**

   (b) A *general* deque is represented using a doubly-linked circular list with a list head. Give implementations of the *RightInsert* and *RightDelete* operations for the deque. **(7 marks)**

7. (a) Draw the tree T representing the following algebraic expression:

$$f(n/2, -x, 0) \; + \; \mathrm{sqrt}(x + n).$$ **(2 marks)**

   (b) Draw the binary tree B corresponding to T using the *natural correspondence* between trees and binary trees.
   Add in the threads to turn B into a *threaded* binary tree with a *head* node. **(3 marks)**

   (c) In which tree traversal order should T be traversed to give the *Reverse Polish* (Polish *postfix*) representation of the expression? To which traversal order of the binary tree B does this correspond? **(2 marks)**

   (d) In what order will the nodes be visited if B is traversed in
   (i) preorder, (ii) inorder, (iii) postorder? **(3 marks)**

8. (a) The following algorithm traverses a binary tree in *preorder*, where P is a pointer to the root of the tree. What modifications would you need to make for the traversal to be in *inorder*? (You do not have to rewrite the whole algorithm, just state what needs to be changed.) **(2 marks)**

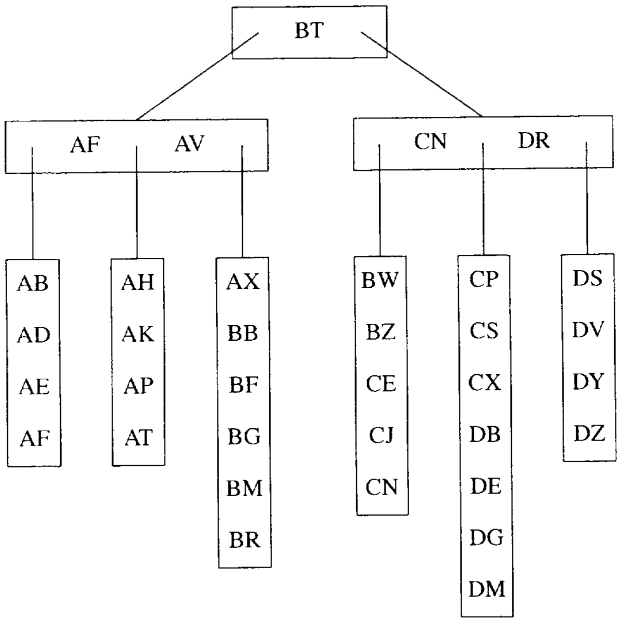
```
PointerStack S
Clear(S)
while (true)
{  if (P ≠ nil)
   {  Visit(P)
      Stack(S,P)
      P ← P↑Llink
    }
    else
   {  if Empty(S) return
      P ← Unstack(S)
      P ← P↑Rlink
    }
}
```

   (b) Draw the binary search tree obtained by inserting the following integers into an empty tree in the following order: 4, 13, 5, 14, 6, 7, 17, 8, 19, 11, 21, 25. **(2 marks)**

(c) A and B are the left and right children of a node in a *threaded* binary tree, and one of them is a *leaf* node. However, it is not specified which is the left child and which is the right child. Given pointers to A and B, write an algorithm or explain in English how to determine which is the left child and which is the right child, and which is a leaf. **(6 marks)**

9. (a) State two suitable methods of organising a large file for which fast random access is required, but sequential access is not required.
Briefly describe one disadvantage of each method. **(3 marks)**

(b) The following diagram shows a B-tree of order 5 with a maximum of 7 records per data page.



(i) Draw the B-tree after the insertion of a record with key DQ. **(2 marks)**

(ii) Inserting one record into a B-tree caused a page to split and resulted in the B-tree shown above. State which of the records might have been the one inserted and draw the B-tree before the insertion for one of these possible records. **(5 marks)**

Note: You do ***not*** need to write in the contents of any unchanged ***data*** pages.

10. We want to construct a binary search tree containing the n integer keys $a[0], a[1], \ldots, a[n-1]$, where $a[0] < a[1] < \cdots < a[n-1]$.

(a) Why would it not be a good idea to start with an empty tree and insert the keys one by one in this order? **(2 marks)**

(b) We could instead use a *divide-and-conquer* approach to write a function `Build(a,L,U)` that returns a pointer to a balanced binary search tree containing the keys $a[L], a[L+1], \ldots, a[U]$. If $L > U$, it should return an empty tree, otherwise the root of the tree should contain the middle key $a[mid]$, where $mid = (L+U)/2$. [You should assume that $k/2$ is rounded down when k is odd.]
Write the function `Build` and draw the tree returned by `Build` when the keys are:
0, 1, 2, 3, 4, 5, 6, 7, 8, 9. **(6 marks)**

(c) Explain very briefly why this method is better than the method described in (a). **(2 marks)**

