IMPERIAL COLLEGE LONDON

DEPARTMENT OF ELECTRICAL AND ELECTRONIC ENGINEERING
EXAMINATIONS 2015

EEE PART II: MEng, BEng and ACGI

ALGORITHMS AND DATA STRUCTURES

Corrected Copy

Monday, 8 June 2:00 pm

Time allowed: 1:30 hours

There are TWO questions on this paper.

Answer BOTH questions.

Question One carries 40% of the marks. Question Two carries 60%.
This exam is OPEN BOOK.

Any special instructions for invigilators and information for candidates are on page 1.

Examiners responsible First Marker(s) : C. Bouganis
Second Marker(s) : D.B. Thomas

Special information for invigilators:
Students may bring any written or printed aids into the examination. The students are allowed to take a copy of the exam.

Information for candidates:
Marks may be deducted for answers that use unnecessarily complicated algorithms, or more than the minimum number of functions.

The Questions

1. a) Figure 1.1 shows a C++ function that calculates the value of the function described in equation (1.1), for any positive integer n (e.g. $n > 0$).

$$f(n) = \begin{cases} 1, & n = 1 \\ 2 \times f(n-1) + 1, & n > 1 \end{cases} \quad (1.1)$$

There are six syntactic or functional errors in the C++ code shown in Figure 1.1. Please identify them.

```
void calculateF (int N) {
    int result = 10;
    int temp = 1;
    for (i=1; i < N; i=i+2) {
        return result = 3*result + 1;
    }
    return temp;
}
```

Figure 1.1 calculateF() function.

[6]

b) Write a recursive C++ function that performs the calculation described in part (a). [6]

[continued on the following page]

c) i) A sequence of numbers is inserted in an ordered binary tree (ascending ordered tree). Draw a tree for the following sequence assuming that the elements in the sequence are inserted in the order shown, and state whether the tree is balanced or not (justify your answer).

$$[5, 10, 15, 3, 11]$$

[2]

ii) The same sequence of numbers as in part (i) is inserted into an AVL tree (ascending ordering is assumed). Draw the final AVL tree, and state whether the tree is balanced or not. [6]

iii) What is the depth of the node in the final tree from part (ii) that contains the element 3? [2]

iv) Using the same sequence of numbers as in part (i), and assuming that the numbers are coming in the order shown, draw the Heap data structure (i.e. the Heap tree), assuming that the largest number is at the root of the Heap. [2]

d) Draw a parse tree for the following expressions, assuming the normal priorities of the operators:

i) $1 * 2 * 3 * 4$ [2]

ii) $5 + (7 + 3)$ [2]

[continued on the following page]

e) Consider the C++ code segment in Figure 1.2. With justification, state the values of variables x and y at points A and B of the code. With justification, state whether this code segment has a memory leak or not.

```
    int x=5;
    int y=10;
    int *p1 = new int;
    int *p2 = new int;
    *p1 = 3;
    p2 = p1;
    x = *p2 + 1;
A:
    x = *p1 + y;
    y = *p2;
B:
```

Figure 1.2 Code segment. [5]

[continued on the following page]

f) Figure 1.3 shows the type declaration for a dynamic linked list, where each node stores an integer in the *data* field.

```
struct Node {
    int data;
    Node * next;
};

typedef Node * NodePtr;
NodePtr hdList = NULL;
```

Figure 1.3 Linked list declaration.

i) Write a C++ function/procedure that takes as its argument a pointer to the linked list and returns a pointer that points to the last node of the list. If the list is empty, your function/procedure should return the NULL value. [3]

ii) Write a C++ function/procedure that takes as argument a pointer to the linked list, and sets all values stored in the *data* field of the nodes to the zero value, except from the *data* field of the last node. [4]

2. Consider an ordered binary tree structure, where each node of the tree structure stores an integer number, and another pair of integers. The pair corresponds to the lower and upper bounds of the range of the numbers stored in the subtree that has that node as a root. Assume that ascending ordering has been imposed on the tree. An instance of the binary tree structure is shown below.

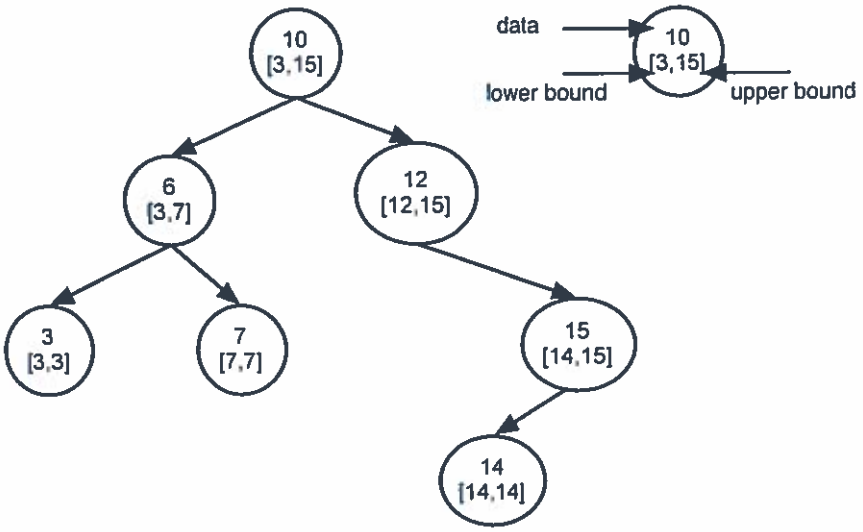

Figure 2.1 Tree structure.

a) Define a structure *treeNode* capable of representing a node of the tree. [5]

b) Write a recursive function/procedure that takes as input the pointer to the root of the tree and returns the number of nodes in the tree. Show how your recursive function/procedure will be invoked. You can always pass extra input arguments in your function/procedure. [10]

[continued on the following page]

c) Write a recursive function/procedure that takes as input the pointer to the root of the tree, and two other integer variables LB and UB that define the lower and the upper bound of a range $[LB, UB]$, and returns the number of nodes in the tree that their data values belong in that range. Show how your recursive function/procedure will be invoked. You can always pass extra input arguments in your function/procedure. For example, for the example provided, if the values of the lower and the upper bound are 13 and 15 respectively, your function/procedure should return the value 2 (i.e. the nodes with values 14 and 15). [10]

d) Write a recursive function/procedure that takes as inputs the pointer to the root of the tree and an integer number, and inserts it in the tree structure in the correct place. While it performs the insertion, it should update the ranges (i.e. lower and upper bounds) of the existing nodes in the tree. Show how your recursive function/procedure will be invoked. You can always pass extra input arguments in your function/procedure. [10]

e) Write a recursive function/procedure that takes as input the pointer to the root of the tree and updates the lower and the upper bound fields for all nodes in the tree. [10]

f) The number of nodes in a tree with node n as its root is denoted by $K(n)$. The density of a node n is defined as follows:

$$\text{density}(n) = \frac{K(n)}{ub(n) - lb(n) + 1.0}$$

where $ub(n)$ and $lb(n)$ are the upper and lower bound fields stored in node n. Write a function/procedure that takes as input the pointer to the root of the tree, and returns the pointer to the node that has the largest density value. Show how your function/procedure will be invoked. You can always pass extra input arguments in your function/procedure. [15]