Special information for invigilators:
Students may bring any written or printed aids into the examination.

Information for candidates:
Marks may be deducted for answers that use unnecessarily complicated algorithms.

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 non-negative integer n (e.g. $f(0) = 0$).

$$f(n) = \begin{cases} 0 & n = 0 \\ f(n-1) + n & n \geq 1 \end{cases} \tag{1.1}$$

Identify six errors in the C++ code shown in Figure 1.1.

```
void calculateF (N) {
    int result = 0;
    for (i=0; i <= n; i++) {
        result = result + n;
        return result;
    }
}
```

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 set of numbers is inserted in an ordered binary tree (ascending ordered tree). Draw a tree for the following set assuming that the elements in the set are inserted in the order shown.

$\{20, 12, 25, 15\}$

[2]

ii) State whether the tree from part (i) is balanced or not, and justify your answer. If the tree is currently unbalanced, then balance it using single and/or double rotations and draw the resulting tree.

[2]

iii) Insert the number 14 into the balanced tree resulting from part (ii) and draw the new tree.

[2]

iv) State whether the tree from part (iii) is balanced or not, and justify your answer. If the tree is currently unbalanced, then balance it using single and/or double rotations and draw the resulting tree.

[2]

v) What is the height of the tree resulting from part (iv)?

[2]

vi) The same set of numbers as in part (i) is inserted into a chained hash table structure with three entries and the following hash function:

$$H(x) = x \; mod \; 3$$

Draw the resulting hash table without any ordering imposed, assuming that the numbers in the set are inserted in the order shown.

[2]

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

i) $10/12+3+4*6$

[2]

ii) $3*(2+6)+9$

[2]

[continued on the following page]

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

```
    int x=20;
    int y=5;
    int *p1 = new int;
    int *p2 = &y;
    *p1 = *p1 + x;
    y= *p1;
A
    p1 = &x;
    p2 = p1;
    x = *p1 + *p2;
    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 *id*, which is unique to each node, and *data*. Both take positive integer values. An empty list is a valid instance of the data structure.

```
struct Node {
    int id;
    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 input the *hdList* pointer and returns a pointer that points to the first element of the list that has its *data* field greater or equal to 10. If the list is empty or no such node exists, the function/procedure should return NULL.

[3]

ii) Write a C++ function/procedure that takes as input the *hdList* pointer, and increments by 1 the *data* field of all the nodes in the list.

[4]

2. Figure 2.1 illustrates the structure of a company. Each node represents an employee in the company and stores information regarding his/her salary and age. Every arrow in the figure corresponds to a employee-manager relationship. You can assume that every node has a unique number (id), each manager can supervise at most two people, and each employee has only one manager.

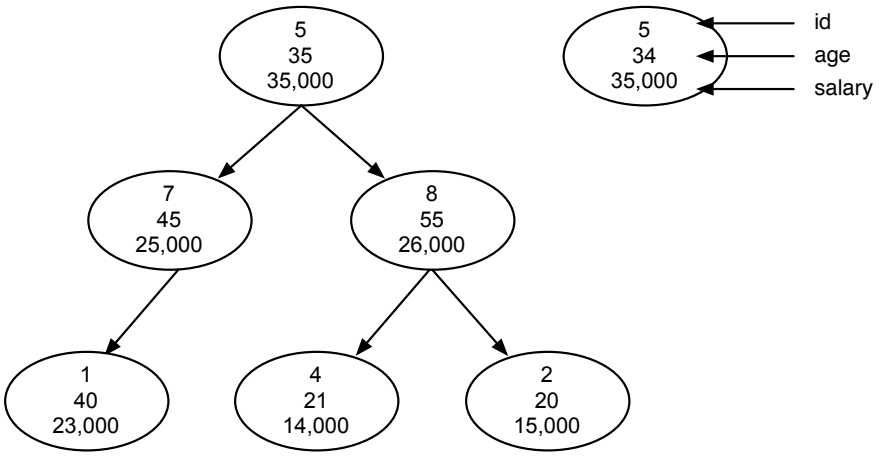

Figure 2.1 Binary tree structure.

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 people that work in that company. Show how your recursive function/procedure will be invoked. You can always pass more input arguments in your function/procedure.

[5]

c) Write a recursive function/procedure that takes as input a pointer to the root of the tree, and returns the number of managers that supervise directly only one person. For example, for the given instance of the tree, the function/procedure should return 1. Show how your recursive function/procedure will be invoked. You can always pass more input arguments in your function/procedure.

[10]

d) Write a recursive function/procedure that takes as input a pointer to the root of the tree and returns the average salary of the people that work in that company. Show how your recursive function/procedure will be invoked. You can always pass more input arguments in your function/procedure.

[20]

[continued on the following page]

e) Write a recursive function/procedure that takes as inputs a pointer to the root of the tree and an *id* value, and returns the number of people that work under the manager with the given *id*. For example, for the instance of the tree shown in Figure 2.1 and an input *id* of 5, the function/procedure should return 5. Show how your recursive function/procedure will be invoked. You can always pass more input arguments in your function/procedure.

[20]