Where an algorithm is asked for, you may write in any suitable pseudocode. Correct syntax for any computer language is not expected.

Answer 3 questions.

1)

a) What are the ACID properties of a transaction and why are they important? [4]

b) What do the terms *logging* and *timestamping* mean? [4]

c) Write a transactional array class for a centralised machine, which has the following interface:

```
class transactional_array
{ protected:
    void *array;
  public:
    transaction();                // Initialisation routine.
    start_transaction();          // Start a new transaction
    abort();
    commit();
    assign(int index, void *ele); // assign ele to array[index]
    void *get(int index);         // return array[index]
}
```

The effect of the routines in this class should be that, whenever a transaction is started then all subsequent assignments can be undone by calling abort, and are made permanent by calling commit. They should also obey the ACID properties.

- You may add any other instance variables and routine parameters or return values that you reasonably require.
- You may assume the existence of a logging class that can have any member functions you consider reasonable. You do not have to provide code for the member functions in the logging class.
- Your solution should use timestamping as the concurrency control mechanism. [15]

d) Demonstrate that your solution possesses the properties of transactions you described in part a). [5]

e) Briefly state what would need to be changed if the class were to be used as part of a distributed system [5]

[TURN OVER]

2)

a) What does it mean for a distributed program consisting of several components to be terminated? [4]

b) The following token-based algorithm is used to detect termination in a graph of processes, `p[i] (i=0..n-1)`, arranged as a logical ring.

```
1. // Actions of p[i] upon receiving a regular message:
2.   colour[i] = white
3.   active[i] = TRUE
4.
5. // Actions of p[i] upon receiving the token
6.   if (token == n_edges)
7.     announce_termination_and_halt()
8.   else if (colour[i] == white)
9.     wait until !active[i]
10.    colour[i] = black; token = 0;
11.  else if (colour[i] == black)
12.    token++
13.
14.  send token to next process in logical ring
15.
16. // Actions of p[i] on becoming idle
17. active[i] = FALSE
```

Explain the operation of this algorithm. What assumptions are made in order for it to work? Show that when termination is announced, the state of the system satisfies the requirements for termination that you identified in a). [15]

c) Now assume that the token has a colour instead of an integer value. Lines 8 to 12 are replaced by:

```
a. else if (colour[i] == white)
b.   wait until !active[i]
c.   colour[i] = black; token = white;
d. else if (colour[i] == black)
e.   token = black;
```

Using these changed rules is there any way to detect termination? If so, explain when the computation is known to have terminated (i.e. what might you replace lines 6 and 7 with?). If not, explain why the rules are insufficient. [14]

[CONTINUED]

3) According to Tanenbaum, *'The goal of RPC is to hide communication by making remote procedure calls look just like local ones.'*

a) To what extent can such a system succeed, and how fundamental are any problems? (You may recall that Nelson attempted to adopt this model of RPC in his Emissary system.) [5]

b) Briefly, outline any further problems that could arise if an RPC-based system built along Tanenbaum's ideal lines were intended for use in a *heterogeneous* distributed system, or argue that there are no such problems. [6]

c) In order even to get close to Tanenbaum's goal, it is necessary to deal with the failures that occur in distributed systems. For each of the following five classes of failure, write brief notes on how they impact the chances of achieving Tanenbaum's goal. Your answers should include both what problems each of them pose and how one might attempt to solve them:

i) The client is unable to locate the server

ii) The request message from the client to the server is lost

iii) The reply message from the server to the client is lost

iv) The server crashes after having received a request

v) The client crashes after sending a request [11]

d) It is necessary to produce a replicated RPC system in which a single RPC invocation in the client can be delivered to multiple servers. What problems might the designer of such a system face? Can they be overcome? [11]

4) *'The goal of distributed systems design is to provide complete transparency to programmers and users'*.

a) What is meant in the above statement by the word *transparency*? Illustrate your answer using examples. [8]

b) Give a reasoned critique of the statement, saying whether, and to what extent, you believe it to be true. Again, illustrate your answer with examples. [12]

c) In a mobile system, mobile nodes can become disconnected from the wired network unpredictably and for relatively long periods (e.g. a train goes into a long tunnel). What effect would this sort of environment have on the ability to provide transparency, and how would you suggest that systems and applications programmers address any problems that result so as to cause least inconvenience to end users? [13]

[TURN OVER]

5)

a) Briefly explain what is meant by *Public Key Cryptography* and by *Secret Key Cryptography* indicating typical characteristics and algorithms for each. [6]

b) Why are *Public Key Certificates* needed? Identify the essential components of a public key certificate. Show how you might use them to communicate your public key to someone in a different security domain to yourself. [8]

c) You have been asked to propose solutions to two security problems in a system using Remote Procedure Call. In the first case RPC parameters need to be protected as they pass between machines. Discuss the likely threats and suggest a suitable security protocol for such parameters. [11]

d) In the second case access control to remote objects and their methods is needed. It has been decided that *capabilities* will be used, where each *rights bit* in the capability indicates whether the holder has access to the corresponding object function or not. For example, the capability for a particular user of a file object might permit access to the read method (read rights = 1) but not the write method (write rights = 0). It is required that capabilities should be able to be passed as parameters between clients and agents that access the object on their behalf and between clients and objects. Discuss the security threats and suggest ways in which such capabilities can be adequately protected. [8]

[END OF PAPER]