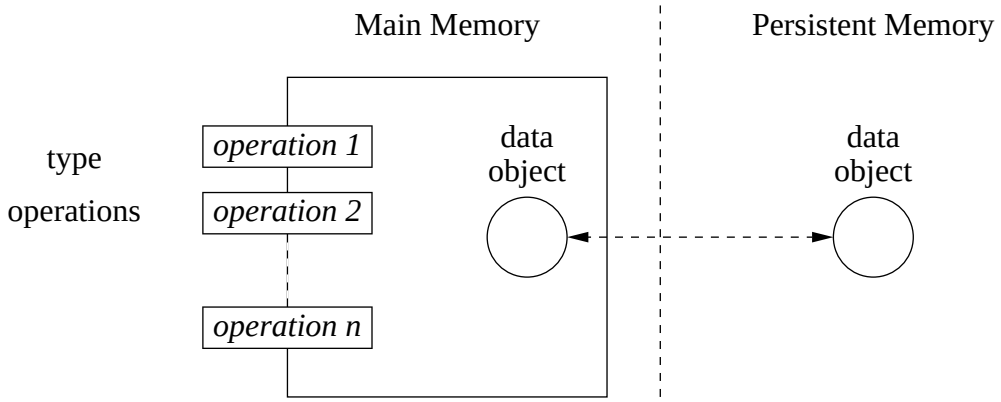
1993 Paper 5 Question 7

Concurrent Systems



The figure illustrates an object model which is used in a concurrent software system. We are concerned with how to implement atomic operations in the presence of concurrency and crashes.

In the descriptions given below, the term *client* indicates an external user of the system. A single-machine multiprocessor implementation should be assumed.

(a) A data object exists in main memory only. Invocations of its type operations involve no writes to persistent memory and no output to clients. Concurrent processes may invoke the object.

How can the operations be made atomic? [8 marks]

(b) A data object exists in persistent memory.

(i) A single operation is invoked on it in response to a request from a client. The result of the invocation is output to the client.

How can the operation be made atomic? [4 marks]

(ii) A client requests a high-level operation which comprises more than one of the type operations on the data object.

How can the high-level operation be made atomic? [8 marks]

1993 Paper 6 Question 7

Concurrent Systems

Monitors are often provided in concurrent programming languages to support the interaction of processes by means of shared data.

Discuss the monitor mechanism and its implementation. [10 marks]

How could the monitor mechanism be adapted to allow for many shared data objects of the same type? [2 marks]

What advantages, if any, could be obtained from using an active object instead of a monitor; that is, a monitor-like structure with one or more internally bound processes? [8 marks]

1994 Paper 5 Question 7

Concurrent Systems

Semaphores are used to achieve exclusive access to a resource for writing but to allow simultaneous read-only access. *Reader* and *Writer* processes have the following structure:

call a procedure to acquire the resource;
use the resource;
call a procedure to release the resource.

The procedures to acquire the resource *for reading* and to release it are as shown below. Writers have priority over readers.

```
procedure acquire_for_reading()
  WAIT(CGUARD);
  ar := ar + 1;      # increment the count of active readers
  if aw = 0 then     # if there are no active writers
    begin
      rr := rr + 1;  # increment the count of reading readers
      SIGNAL(READ)
    end;
  SIGNAL(CGUARD);
  WAIT(READ);

procedure release_after_reading()
  WAIT(CGUARD);
  rr := rr - 1;
  ar := ar - 1;
  if rr = 0 then
    while ww<aw do   # ww counts "writing writers"
      begin
        ww := ww + 1;
        SIGNAL(WRITE)
      end;
  SIGNAL(CGUARD);

# note that "writing writers" are assumed to take turns to use
# the resource.
```

What is the function of the operations on the semaphore `CGUARD`? [3 marks]

Why is the following, apparently simpler, version of `acquire_for_reading()` incorrect?

```
WAIT(CGUARD);
ar := ar + 1;
if aw ≠ 0 then WAIT(READ);
rr := rr + 1;
SIGNAL(CGUARD);
```
[3 marks]

What is the function of the operations on the semaphores `READ` and `WRITE`? [4 marks]

The procedures to acquire the resource for writing and to release it after writing are similar to those given but, in addition, exclusive access to the resource must be enforced. Outline how this could be programmed. [2 marks]

Write the procedure `release_after_writing()`. [8 marks]

1994 Paper 6 Question 7

Concurrent Systems

A transaction processing system for a banking application is to be implemented using object semantics. Bank account objects include the following among their operations:

```
credit (account_id, amount)
debit (account_id, amount)
add_interest (account_id)
```

By defining "transactions" based on this example, show what is meant by

(*a*) a non-serialisable execution schedule [3 marks]

(*b*) a non-strict execution schedule leading to a cascading abort [3 marks]

Explain the ACID properties of transactions, drawing on the above application for examples. Indicate which properties are concerned with failure resilience and which with concurrency control. [8 marks]

Again using the above application for examples, explain concurrency control based on

(*a*) two-phase locking [3 marks]

(*b*) time-stamp ordering [3 marks]

1995 Paper 3 Question 1

Concurrent Systems

Why are database transactions necessary? [4 marks]

Describe the ACID properties of transactions. [4 marks]

What are the differences between optimistic and pessimistic mechanisms of concurrency control (with particular reference to the ACID properties)? [4 marks]

What factors determine whether an optimistic or pessimistic concurrency control policy is appropriate for a transaction processing system? [4 marks]

With reference to these factors, state whether an optimistic or pessimistic policy would lead to a more efficient system in the following cases:

(a) a flight booking system

(b) a police criminal record database

(c) a banking transaction processing system

(d) a database maintaining patient records

[4 marks]

1995 Paper 4 Question 2

Concurrent Systems

An expansion card to manage a number of telephone lines is to be added to a workstation. The hardware is able to pick up and put down lines, dial a number and answer the phone. It generates interrupts to signal conditions such as a phone line being answered.

(*a*) What additions have to be made to the operating system to allow programs to use the phone hardware? [4 marks]

(*b*) Suggest a mechanism by which the operating system can block a user process until a phone line is answered. [4 marks]

A *voice mail server* is to be developed and given exclusive use of the lines. It is to be activated by a user clicking the button on his/her active badge, a mobile device allowing a user's location to be determined. The voice mail server then phones the user at the phone nearest to him/her. After authentication by typing a code, the server reads the user's mail to him/her using a speech synthesiser. The pseudo-code procedure shown below is part of this service. Every time a badge click is detected, a thread is forked to execute procedure `ActiveBadge_Click_Handler`.

```
VAR line : ARRAY [1..NUM_LINES] OF {Free,Busy};

PROCEDURE ActiveBadge_Click_Handler(user: STRING;
                                    location: STRING);
VAR which_line: INTEGER;
BEGIN
  which_line := 1;
  WHILE (line[which_line] = Busy) DO  (* Loop until a line *)
    which_line := which_line+1;       (* becomes free *)
    IF (which_line > NUM_LINES) THEN which_line := 1;
  END;
  line[which_line] := Busy;           (* Mark line as busy *)
  Pickup(which_line);
  Dial_Authenticate_Speak(which_line,user,location);
                                      (* Does clever stuff *)
  Hangup(which_line);
  line[which_line] := Free;           (* Mark line as free *)
END;
```

(*c*) Explain whether or not it is possible to implement the server using threads provided by a run-time system. [4 marks]

(*d*) Outline a possible race condition in the program. On what types of system could it occur? [4 marks]

(*e*) How could the program be amended to avoid the race condition? Note: Language-specific details are unimportant. [4 marks]

1996 Paper 3 Question 1

Concurrent Systems

For a transaction model based on objects:

(*a*) Define how conflict may be specified in terms of object operation semantics. Give an example of conflicting operations. Give an example of non-conflicting operations that would conflict with read–write semantics. [3 marks]

(*b*) Define the necessary and sufficient condition for two transactions to be serializable. Give an example of a non-serializable pair of transactions. [2 marks]

(*c*) Define the necessary and sufficient condition for a concurrent execution schedule of a number of transactions to be serializable. Give an example of a serialization graph for four transactions that are non-serializable. [2 marks]

(*d*) Discuss how the three general approaches to providing concurrency control for transaction systems are designed to enforce the property you have defined in (*c*) above. [13 marks]

1996 Paper 4 Question 1

Concurrent Systems

Give a detailed criticism of the following options for supporting interactions between processes in separate address spaces:

(a) pipes

(b) asynchronous message passing

(c) synchronous message passing

[20 marks]

1997 Paper 3 Question 2

Concurrent Systems

In a given application, processes may be identified as "readers" or "writers". Multiple-reader, single-writer access to data is required to be implemented with priority for writers. Readers execute procedures `startread` and `endread` before and after reading. Writers execute `startwrite` and `endwrite`.

Give pseudocode for the `startread` procedure using

(a) semaphores

(b) conditional critical regions

(c) monitors

(d) guarded commands

Explain via comments in your pseudocode how concurrency control is achieved in each case and criticise each approach from the viewpoint of the programmer and the implementer.

[20 marks]

1997 Paper 4 Question 2

Concurrent Systems

Explain the concept of serialisability of access to objects by concurrent processes. Consider the cases where operations that are meaningful to an application involve (a) only a single object and (b) related operations on a number of objects. Explore the cases where (c) the objects exist only in the main memory of a single computer for the duration of a program execution and (d) the objects exist in persistent store independent of program execution. [20 marks]

1998 Paper 3 Question 2

Concurrent Systems

An n-process mutual exclusion algorithm has entry and exit protocols given below. In order to express the algorithm concisely we use a lexicographic "less than" relation on ordered pairs of integers, so that:

$$(a, b) < (c, d) \text{ if } a < c \text{ or if } a = c \text{ and } b < d.$$

Entry protocol for the critical region for process i:

$taking[i] := true;$
$ticket[i] := max(ticket[0], ticket[1], \ldots, ticket[n-1]) + 1;$
$taking[i] := false;$
for $j := 0$ **to** $n - 1$ **do**
  begin
    while $taking[j]$ **do** *no-op*;
    while $ticket[j] \neq 0$ **and** $(ticket[j], j) < (ticket[i], i)$ **do** *no-op*;
  end

Exit protocol for the critical region for process i:

$ticket[i] := 0;$

(a) Illustrate fully the operation of the algorithm by showing, for a small value of n, successive values of the arrays *taking* and *ticket* under a variety of concurrent executions. Explain by means of short comments on the values. [14 marks]

(b) Is it possible or likely that a value in the array *ticket* might overflow? Why? [2 marks]

(c) A RISC processor has an atomic *read-and-clear-memory* instruction. Give pseudo-code for the entry and exit protocols using the instruction. [4 marks]

1998 Paper 4 Question 2

Concurrent Systems

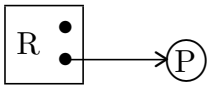
In a system which allocates resources dynamically

(*a*) What are the resource allocation policies that make it necessary to consider the possibility of deadlock? [3 marks]

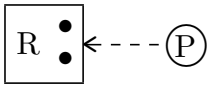
(*b*) If there is one instance of each resource type what is the necessary and sufficient condition for deadlock to exist? [2 marks]

Using the notation

- for an instance of a resource



process P has an instance of resource R



process P is requesting an instance of resource R

(*c*) Draw a resource–wait graph for five processes where at least three are deadlocked. [5 marks]

(*d*) Give the allocation and request matrices corresponding to your graph. [5 marks]

(*e*) Illustrate a deadlock detection algorithm using your matrices as an example. [5 marks]

1999 Paper 3 Question 1

Concurrent Systems

The runtime system of a concurrent programming language includes semaphore management and user-thread management modules. The design assumption was that the operating system on which the concurrent programs are to run supports only one process per address space. Outline the functionality and interaction of semaphore and user-thread management. [6 marks]

The runtime system is to be ported to run on an operating system which supports multi-threaded processes and symmetric multiprocessing.

(a) Describe how the two modules must be changed in order to support

- one kernel-thread per user-thread [5 marks]
- scheduler activations [5 marks]

State any assumption you make about the functionality of the operating system.

(b) Discuss the problem of concurrent execution of the semaphore management module and how this concurrency might be controlled. [4 marks]

1999 Paper 3 Question 3

Further Java

Describe the facilities in Java for defining classes and for combining them through composition, inheritance and interfaces. Explain with a worked example how they support the principle of encapsulation in an object-oriented language. [15 marks]

What is meant by *reflection* or *introspection* in Java? Give an example of its use. [5 marks]

1999 Paper 4 Question 2

Concurrent Systems

Assume that on a power fail the entire contents of main memory are lost. Discuss the various problems that can be caused and how they might be solved in

either a centralised transaction processing system

or a remote procedure call protocol

[20 marks]

1999 Paper 4 Question 3

Further Java

Describe the facilities in Java for restricting concurrent access to critical regions. Explain how shared data can be protected through the use of objects. [8 marks]

The built-in facilities for restricting concurrency in Java allow only one thread at a time to be executing within the critical region. A different approach is to distinguish *shared* and *exclusive* access to a critical region: any number of *readers* may share access at the same time, but only one *writer* may acquire exclusive access (excluding any readers while it does so).

Specify a `MultiSync` class in Java with methods to acquire and release both read and write access, and sketch its implementation. [6 marks]

Derive a `MultiBuffer` class that extends `MultiSync` with methods to store and read a data field, ensuring that any locks are released when a waiting thread is interrupted. Your example may use a simple data field such as an integer but you should explain why such an elaborate scheme of concurrency control is unnecessary in this case. [6 marks]

2000 Paper 3 Question 1

Concurrent Systems

(a) A software module controls a car park of known capacity. Calls to the module's procedures *enter*() and *exit*() are triggered when cars enter and leave via the barriers.

Give pseudocode for the *enter* and *exit* procedures

(i) if the module is a monitor [8 marks]

(ii) if the programming language in which the module is written provides only semaphores [4 marks]

(b) Outline the implementation of

(i) semaphores [4 marks]

(ii) monitors [4 marks]

2000 Paper 3 Question 2

Further Java

Write brief notes on the facilities for lightweight concurrency in Java. You should cover the following topics, using code fragments to illustrate your answer:

(a) creating a thread

(b) mutual exclusion

(c) signalling

(d) asynchronous interruption

(e) managing priority

[4 marks each]

2000 Paper 4 Question 1

Concurrent Systems

In a proposed, next-generation banking system a number of transactions are to be scheduled to run concurrently:

- Debit (D) transactions to make payments from customer accounts to a credit card company.
- Interest (I) transactions to add daily interest to customer account balances.
- Transfer (T) transactions which first check whether the source account contains sufficient funds then either abort or continue the transfer from source to destination accounts. Customer x is running a T to transfer £1000 from A to B. Customer y is running a T to transfer £200 from B to A.

(a) Discuss the potential for interference between any of these transactions. [7 marks]

(b) Demonstrate the effect of concurrency control based on strict two-phase locking in relation to the discussion in (a). [8 marks]

(c) Comment on the scope of concurrency control in relation to the discussion in (a). [5 marks]

[Hint: you may assume that operations on bank account objects, such as debit, credit and add-interest are atomic.]

2000 Paper 4 Question 3

Further Java

Describe the model for handling graphical output and interactive input in the Abstract Windowing Toolkit (AWT) for Java. Your answer should cover the use of

- hierarchies of classes
- overriding methods
- interfaces
- inner classes
- spatial hierarchy

[4 marks each]

2001 Paper 3 Question 1

Concurrent Systems

A committed ACID transaction transforms a database from one consistent state to another.

(*a*) Define the property of isolation in a transaction system. [2 marks]

(*b*) The property of isolation may be enforced either at transaction commit or at all times throughout transaction execution. Describe how each of these approaches may be implemented, and the tradeoffs between the approaches, for

(*i*) two-phase locking [5 marks]

(*ii*) timestamp ordering [5 marks]

(*iii*) optimistic concurrency control [8 marks]

2001 Paper 3 Question 2

Further Java

(*a*) Describe the differences and similarities between abstract classes and interfaces in Java. How would you select which kind of definition to use? [5 marks]

(*b*) An enthusiast for programming with *closures* proposes extending Java so that the following method definition would be valid:

```
Closure myCounter (int start) {
  int counter = start;
  return {
    System.out.println (counter ++);
  }
}
```

The programmer intends that no output would be produced when this method is executed, but that it would return an object of a new type, `Closure`, and that invoking `apply()` on that object would cause successive counter values to be printed.

By using an *inner class* definition, show how this example could be re-written as a valid Java program. [5 marks]

(*c*) A common programming mistake is to try to define a class to have more than one superclass. For example a naïve programmer may write

```
class AutoTree extends Thread, BinaryTree {
  ...
}
```

when defining a new kind of data structure which uses an additional thread to keep the tree balanced. Describe *three* ways in which this problem can be resolved to produce (one or more) valid class definitions. State, with a brief justification, which you would use here. [10 marks]

2001 Paper 4 Question 2

Concurrent Systems

An interprocess communication environment is based on *synchronous* message passing. A server is to be designed to support a moderate number of simultaneous client requests.

Clients send a request message to the server, continue in parallel with server operation, then wait for the server's reply message.

Discuss the design of the server's interaction with the clients. Include any problems you foresee and discuss alternative solutions to them. [20 marks]

2001 Paper 4 Question 3

Further Java

(*a*) Describe how mutual-exclusion locks provided by the `synchronized` keyword can be used to control access to shared data structures. In particular you should be clear about the behaviour of concurrent invocations of different synchronized methods on the same object, or of the same synchronized method on different objects. [6 marks]

(*b*) Consider the following class definition:

```
class Example implements Runnable {
  public static Object o = new Object();
  int count = 0;

  public void run() {
    while (true) {
      synchronized (o) {
        count ++;
      }
    }
  }
}
```

Show how to start two threads, each executing this `run` method. [2 marks]

(*c*) When this program is executed, only one of the `count` fields is found to increment, even though threads are scheduled preemptively. Why might this be? [2 marks]

(*d*) Define a new class `FairLock`. Each instance should support two methods, `lock` and `unlock`, which acquire and release a mutual exclusion lock such that calls to `unlock` never block the caller, but will allow the longest-waiting blocked thread to acquire the lock. The lock should be *recursive*, meaning that the thread holding the lock may make multiple calls to `lock` without blocking. The lock is released only when a matched number of `unlock` operations have been made.

You may wish to make use of the fact the `Thread.currentThread()` returns the instance of `Thread` that is currently executing. [10 marks]

2002 Paper 4 Question 1

Concurrent Systems and Applications

Below are four potential problems and two proposed solutions for each one. For each of the problems, give a brief example showing the proposed solutions and explain the advantages and disadvantages of each.

(*a*) Data held in one object is to be made available throughout a large, possibly distributed, application.

  (*i*) Store the data in a field with the `public` modifier.

  (*ii*) Store it in a field with the `private` modifier but provide `public` methods to access its value.

(*b*) A class `C` implements an interface `I1` but some code is designed to access it through an alternative interface `I2`. The two interfaces support similar operations.

  (*i*) Define a new class using inheritance.

  (*ii*) Use the *Adapter* design pattern.

(*c*) You are designing a data structure and need to decide how to perform concurrency control in case it is used in a multi-threaded application.

  (*i*) Use `synchronized` methods (or other features) to make the methods safe for concurrent use.

  (*ii*) Do not manage concurrency here and add comments to the source code.

(*d*) You have a class that defines how to communicate with a remote server using a TCP socket. The connection is established in the constructor and you must decide how to close it.

  (*i*) Provide an explicit `close` method in your class.

  (*ii*) Use a `finalize` method.

[5 marks each]

2002 Paper 5 Question 4

Concurrent Systems and Applications

(*a*) The `suspend()` and `resume()` methods defined on `java.lang.Thread` can be used to block and unblock a designated thread. Explain why these methods should not be used. [4 marks]

(*b*) Define a Java class `Barrier` that pairs together two different kinds of thread (`A` and `B`) in a concurrent system. It should support two methods, `enterA(Object o)` and `enterB(Object o)`. A call to one `enter` method blocks until a call is made to the other. At that point both invocations continue, `enterA` returning the value passed to `enterB` and *vice versa*.
[8 marks]

(*c*) A number of barbers and customers have to co-ordinate their actions in a barbers' shop. Each barber has a chair in which a customer sits while receiving a haircut and in which that barber dozes when he has no customer. There is plenty of waiting space.

Define Java classes `Barber`, `Customer` and `Shop` to model this situation. The class `Customer` should support two methods:

(*i*) `Barber getHaircut(Shop s)` – request service at `s`, blocking until the customer is served and returning the allocated barber;

(*ii*) `void leaveChair(Barber b)` – signal that `b` can take another customer.

The class `Barber` should support two corresponding methods:

(*i*) `Customer getCustomer(Shop s)` – wait to be allocated a customer at `s`;

(*ii*) `void finishedCustomer(Customer c)` – signal to `c` that the haircut is finished and wait for `c` to leave the chair.

[8 marks]

2002 Paper 6 Question 4

Concurrent Systems and Applications

A system is to support abortable transactions that operate on a data structure held only in main memory.

(*a*) Define and distinguish the properties of *isolation* and *strict isolation*. [2 marks]

(*b*) Describe *strict two-phase locking* (S-2PL) and how it enforces strict isolation. [4 marks]

(*c*) What impact would changing from S-2PL to ordinary 2PL have (*i*) during a transaction's execution, (*ii*) when a transaction attempts to commit and (*iii*) when a transaction aborts? [6 marks]

(*d*) You discover that the system does not perform as well as intended using S-2PL (measured in terms of the mean number of transactions that commit each second). Suggest why this may be in the following situations and describe an enhancement or alternative mechanism for concurrency control for each:

(*i*) The workload generates frequent contention for locks. The commit rate sometimes drops to (and then remains at) zero. [2 marks]

(*ii*) Some transactions update several objects, then perform private computation for a long period of time before making one final update. [2 marks]

(*iii*) Contention is extremely rare. [4 marks]

2003 Paper 4 Question 8

Concurrent Systems and Applications

For each of the following topics explain what features Java provides and give a brief example of when you might use them:

(*a*) scheduling clean-up operations when a data structure ceases to be reachable;

(*b*) calling a method given its name as a string;

(*c*) converting a data structure based on objects to one held in a `byte[]` array;

(*d*) creating and instantiating a new class given its definition in a `byte[]` array;

(*e*) implementing a method in a language other than Java.

There is no need to recall the precise names and parameters of library facilities. So long as you are clear about the principles involved, you may assume that your reader will be able to look up the details. [4 marks each]

2003 Paper 5 Question 4

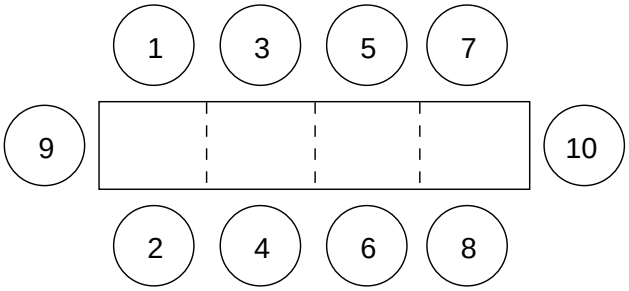
Concurrent Systems and Applications

In a concurrent system it is usual to distinguish between *safety properties* and *liveness properties*.

(*a*) What is meant by *mutual exclusion*, and how does a programmer ensure it is enforced in Java? Indicate whether it relates to safety or to liveness. [4 marks]

(*b*) What is *deadlock* and how can it occur? Again, indicate whether it relates to safety or to liveness. [4 marks]

A group of ten *feeding philosophers* has gathered to eat at a long trough, as shown below. They eat by leaning over the trough and so must be careful not to let their heads collide in the middle. To ensure safety they decide that around each position in the trough at most one philosopher is allowed to be eating at any one time. For instance, only one of 1, 2 or 9 could be eating at once, but there is no problem with 9, 3, 6 and 10 all eating together.



In a Java system, each philosopher is modelled by a thread which loops between eating and non-eating phases. There are four `TroughPosition` objects, corresponding to the four positions along the trough. Each philosopher has a reference to the position at which he or she is eating.

(*c*) Define a Java class `TroughPosition` which supports methods `startEating()` and `finishEating()` to ensure safe execution for the philosophers at that position. [6 marks]

(*d*) Can your solution suffer from deadlock? Either explain why it cannot occur, or explain how it could be avoided. [3 marks]

(*e*) Can your solution suffer from starvation – that is, can one thread continuously remain in the `startEating()` method? Either explain why this cannot occur, or explain how it could be avoided. [3 marks]

2003 Paper 6 Question 4

Concurrent Systems and Applications

A distributed system is being designed to hold details of appointments on behalf of its users. The system comprises a single server holding the appointments and a number of clients which interact with the server over a network supporting an unreliable datagram-based protocol such as UDP.

(*a*) The server is to provide operations using a remote procedure call (RPC) interface with exactly-once semantics. For instance,

```
boolean addEntry (Client c, Client d, Appointment a);
```

is used by client `c` to add an entry to the diary of client `d`. Describe how this RPC system can be implemented, including

(*i*) how parameters are marshalled and unmarshalled;

(*ii*) how exactly-once semantics are achieved;

(*iii*) how threads are used, both within a client and within the server.

You may assume that the clients already know an appropriate network address to contact the server and that separate mechanisms are used for authentication and for security in general. [10 marks]

(*b*) The system is to be extended to support transactional-style accesses by providing three further operations,

```
void startTransaction (Client c);
void abortTransaction (Client c);
boolean commitTransaction (Client c);
```

Define *strict isolation* and describe how, in this case, it could be enforced by *timestamp ordering*. [10 marks]

2004 Paper 4 Question 8

Concurrent Systems and Applications

A multi-threaded application is using a long linked list of integers. The list is accessed through `synchronized` methods on a `ListSet` object.

The list itself comprises a chain of `ListNode` objects in ascending numerical order. The chain always starts and ends with special *sentinel* nodes conceptually containing $-\infty$ and $+\infty$ respectively. This simplifies the implementation of operations on the list: they do not have to deal with inserting elements at the very start or at the very end.

(*a*) Sketch the definition of `ListSet` and `ListNode` as Java classes. You need only give appropriate field definitions and the implementation of an `insert` method on `ListSet`. [4 marks]

(*b*) An engineer suggests that, instead of holding a lock on a `ListSet` object, threads only need to lock a pair of `ListNode` objects in the region that they are working.

(*i*) Define methods `lock` and `unlock` for your `ListNode` class to allow a thread to acquire a mutual exclusion lock on a given node. [6 marks]

(*ii*) Show how your insert method could be updated to incorporate the engineer's idea. [8 marks]

(*iii*) Do you think the new implementation will be faster than the original one? Justify your answer. [2 marks]

2004 Paper 5 Question 4

Concurrent Systems and Applications

(*a*) Consider a simple client-server system implemented using Java Remote Method Invocation (RMI). Describe:

(*i*) how the interface between the client and the server is defined; [4 marks]

(*ii*) how a particular instance of the server is named. [4 marks]

(*b*) One operation provided by the server is to merge the contents of two hashtables, returning the result in a new hashtable. On a centralized system the operation's signature could be defined as follows:

```
Hashtable mergeTables(Hashtable a, Hashtable b)
```

Describe in detail the semantics with which parameters are passed and results are returned when this operation is implemented over RMI. [4 marks]

(*c*) The designer of a new RMI system proposes lazily copying the contents of objects that are passed between the client and the server. For instance, a large data item passed to `mergeTables` would have to be sent only if the server actually tries to access it. It is hoped that this scheme will make distributed systems faster because it will reduce the volume of data sent over the network.

(*i*) Describe a situation in which the new system is likely to be faster than traditional RMI and a separate situation in which it is likely to be slower. [2 marks each]

(*ii*) How would this new scheme change the semantics with which parameters are passed and results are returned? [4 marks]

2004 Paper 6 Question 4

Concurrent Systems and Applications

(*a*) A transaction processing system is using a *write-ahead log* as a mechanism for providing persistent storage. What information must be written to the log

(*i*) when a transaction starts;

(*ii*) when a transaction performs an operation on a persistent object;

(*iii*) when a transaction commits? [2 marks each]

(*b*) Describe how the log can be used to roll-back a transaction that has aborted or become deadlocked. [6 marks]

(*c*) The log can also be used to recover after some kinds of system failure.

(*i*) Describe how the log can be used to recover after a *fail-stop* crash. [2 marks]

(*ii*) What is meant by *checkpointing*? How will using it affect the structure of the log and the recovery procedure after a crash? [6 marks]

2005 Paper 4 Question 8

Concurrent Systems and Applications

(*a*) Java includes support for converting between objects and a stream of bytes.

(*i*) Describe the Java serialization mechanism. [4 marks]

(*ii*) What circumstances can cause exceptions to be thrown during serialization? [2 marks]

(*iii*) What is the effect of the `transient` modifier? To which components of a class is it legal to apply the `transient` modifier? [3 marks]

(*iv*) Why do some programmers prefer to use `Externalizable` instead of `Serializable`? [1 mark]

(*b*) Java includes support for converting a stream of bytes into a class definition. Describe the facilities that provide this functionality and ensure that the type-safety of the Java Virtual Machine cannot be compromised. [4 marks]

(*c*) What are *native methods* and why might they be useful? [2 marks]

(*d*) What are *reference objects*? How does a *ReferenceQueue* provide a means to release resources when they are no longer needed? Explain how reference objects can be used to construct data structures that permit the garbage collector to release memory when the Java Virtual Machine might otherwise run out of memory. [4 marks]

2005 Paper 5 Question 4

Concurrent Systems and Applications

(*a*) Mutual exclusion is an important consideration in many multi-threaded processes.

(*i*) Describe the syntax and semantics of each of the different ways of using the `synchronized` keyword in Java. [5 marks]

(*ii*) What is a *re-entrant* mutual exclusion lock and why is it helpful that the locks used by the Java Virtual Machine to implement synchronized methods be re-entrant? [2 marks]

(*iii*) Accesses to fields of most data types in Java are atomic but some are not. Give an example of a field access that is not atomic and explain how read and write access can be achieved in a thread-safe fashion. [2 marks]

(*iv*) Recent editions of the Java language include *generics*. What is the scope of the mutual exclusion caused by the use of the `synchronized` keyword in a generic class definition? [1 mark]

(*b*) *Deadlock* can occur in multi-threaded applications.

(*i*) What *four* properties hold when deadlock exists? [4 marks]

(*ii*) Which of these are properties of the Java language and which depend on the program being executed? [1 mark]

(*iii*) A practical strategy to avoid deadlock is to enforce an ordering on acquiring locks. Explain how this ensures that deadlock is never possible. [2 marks]

(*c*) Mutual exclusion locks are language-level features. Explain how they can be implemented in terms of **either** Counting Semaphores provided by the operating system **or** atomic compare-and-swap operations provided by the hardware. [3 marks]

2005 Paper 6 Question 4

Concurrent Systems and Applications

(*a*) *Transactions* provide a means of grouping together a collection of separate operations to form a single logical operation. Explain the difference between implementations that enforce *strict isolation* and *non-strict isolation*. What are *dirty reads* and *cascading aborts*? [8 marks]

(*b*) Optimistic Concurrency Control (OCC) is a mechanism used to enforce isolation of transactions.

(*i*) Explain the OCC algorithm. [4 marks]

(*ii*) Is deadlock possible? Are cascading aborts possible? [2 marks]

(*iii*) Under what circumstances might this mechanism yield poor performance? [1 mark]

(*c*) Explain how to use a *write-ahead log* to provide persistence by detailing what information must be written to the log and how it can be used to recover from a fail-stop catastrophe. What are *checkpoints* and why are they useful? [5 marks]

2006 Paper 4 Question 8

Concurrent Systems and Applications

(*a*) To which components of a class definition is it legal to apply the `strictfp` modifier in Java and what effect does it have in each case? [3 marks]

(*b*) When the `transient` modifier is applied to a field of type *object reference to an array of Objects* in a class definition, Java's default Serialization mechanism will omit the array itself and all of the elements in the array from a serialized representation. A useful and reusable class would offer a *partially transient array*: a class containing an ordered list of elements indexed by integers from zero upwards, which allows the programmer to toggle each item separately between being transient and non-transient. The class should provide methods to get and set the values in each position of the array and to indicate that an element should behave as though it is/is not transient.

(*i*) Define a *generic* Java interface, `PartiallyTransientArray`, appropriate for this purpose. [4 marks]

(*ii*) Give the definition of a *generic* class providing the functionality of a partially transient array ensuring that, when serialized, the class and (only) the non-transient elements are included in the output. It is sufficient to state the size of the array at construction-time and for it to be unchangeable thereafter. [6 marks]

(*c*) Explain, drawing examples from your partially transient array class, how formal type parameters can be restricted in the Java language using the `extends` and `super` keywords. [4 marks]

(*d*) What are the drawbacks of the "Generics" features of the Java language for providing type-polymorphism? [3 marks]

2006 Paper 5 Question 4

Concurrent Systems and Applications

(*a*) The following Java interface describes the API for a first-come, first-served (FCFS) mutual exclusion system, designed to work even if threads are interrupted in the `enter` and `exit` routines.

```
interface FCFS {
  public void enter();
  public void exit();
}
```

Sketch a concrete class, `FCFSImpl`, implementing this interface, which does not need to be re-entrant, ensuring that you satisfy the following requirements:

(*i*) if a thread is interrupted while executing the entry protocol, it should abort its attempt to gain entry and cleanly terminate the call; you may assume that the calling code will not then enter the critical region; [6 marks]

(*ii*) the exit protocol should notify a particular thread and not simply call `notifyAll()`. [6 marks]

(*b*) Object allocation graphs can be used to detect deadlock in a concurrent application.

(*i*) Give an example of an object allocation graph and explain the meanings of the different components. [2 marks]

(*ii*) Describe an algorithm which can use an object allocation graph and an object request matrix to determine whether or not deadlock exists. [5 marks]

(*iii*) Describe how to distinguish between cases in which deadlock has occurred and those in which deadlock is inevitable but is yet to occur. [1 mark]

2006 Paper 6 Question 4

Concurrent Systems and Applications

(*a*) Java's *Reference Objects* provide a means to interact with the runtime garbage collector.

(*i*) Provide a Java class implementing a *Leaky Array*: a fixed-size, ordered sequence of `Objects`, indexed by integers from zero upwards. The items may be discarded (individually) by the garbage collector when the system is running low on memory. Provide concurrency-safe accessor methods to get and set the items stored in the array. Ensure that any internal data structures cannot be manipulated other than via your accessor methods. It should be possible for classes in other packages to construct new Leaky Array objects and be able to invoke the accessor methods. Derived classes should have concurrency-safe means of determining the size of the array (but be unable to change it) and counting the number of not-null items stored. [6 marks]

(*ii*) Describe carefully when a `Finalizer` method defined on a Java class will be executed. Might two finalizers be executed concurrently? What guarantees does Java make about the execution of finalizers? [5 marks]

(*iii*) How can a `ReferenceQueue` be used to provide more control than finalizer methods? [4 marks]

(*b*) Does Java's garbage collector clean up memory allocated by code in a *native method*? What facilities exist to specify whether or not Java objects created by native code are eligible for garbage collection? [5 marks]

2007 Paper 4 Question 1

Concurrent Systems and Applications

(*a*) What is meant by a *serializable* order for two or more transactions? [2 marks]

(*b*) Explain how *timestamp ordering* (TSO) enforces isolation. [5 marks]

(*c*) Draw and explain a history graph for two transactions whose invocations of a set of conflicting operations are serializable but which are rejected by TSO. [5 marks]

(*d*) Considering Optimistic Concurrency Control (OCC):

(*i*) State the properties of a transaction's set of shadow copies that must be verified at commit time. [2 marks]

(*ii*) Carefully explain the algorithm used by a single-threaded commit-time validator. [4 marks]

(*e*) Consider a system in which the transactions cause updates to objects which are not all located on a single server but which are distributed widely around the Internet. What factors would influence your choice of using TSO or OCC to enforce isolation? [2 marks]

2007 Paper 5 Question 4

Concurrent Systems and Applications

(*a*) A software engineer is developing computer-aided design software for an architect who works with designs for three different types of building: Contemporary, Victorian, and Roman. When laying out buildings, the architect places windows, doorways, walls and roofs in the appropriate style for the building on which she is working. Explain, including all the Java interfaces and code fragments, how the software can use the Abstract Factory design pattern to construct and maintain its internal data structures representing a building. [6 marks]

(*b*) In terms of software testing, what is *equivalence partitioning*, how can it be done, and how can the partitioned sets be used to select test patterns? [6 marks]

(*c*) What are *decision coverage* and *condition coverage*? [2 marks]

(*d*) Describe Java's Remote Method Invocation (RMI) system, including the means by which objects and servers are named, how clients perform binding, and the purpose of the RMI CodeBase. [6 marks]

2007 Paper 6 Question 4

Concurrent Systems and Applications

(*a*) For *each* of the following tasks, give a code fragment that achieves as much of the task as is possible using the introspection API of the Java programming language and state which aspect(s) of the requirement is/are impossible in Java.

(*i*) Given a (non-null) object reference, determine whether or not the object has a *public*, *static* method named `myMethod` which takes no arguments. [4 marks]

(*ii*) Invoke the static method `public foo(java.lang.Integer x)` on a class definition named `MyClass` with argument `myInt` when the overloaded, static method `foo(java.lang.Number x)` is also defined on `MyClass`. [4 marks]

(*iii*) Given an object reference to an instance of a class named `Rocket`, set the value of its public field `numberOfEngines` to the (primitive) `int` value `5` and make the method `launch()` into a synchronized method. [4 marks]

(*b*) You are porting the JVM to a new processor that does not have a compare-and-swap (CAS) instruction but does offer test-and-clear (TAC): `tac(addr)` atomically reads the value stored at memory address `addr`, overwrites it with zero, and returns the value that was seen. Construct a Java-style *re-entrant* mutex using TAC. [4 marks]

(*c*) A server daemon has an object of type `Client` for each currently-active connection. Instances of `Client` each contain an object reference to a `java.lang.Socket` which must be closed (by calling `close()`) when the `Client` object is garbage collected. Show, by means of Java code fragments, how Phantom References and Reference Queues can be used to invoke the `close()` method in a timely fashion following an instance of `Client` becoming unreachable. [4 marks]

2008 Paper 4 Question 1

Concurrent Systems and Applications

(*a*) Describe the syntax of the `synchronized` keyword in Java and explain the effect on the runtime behaviour of a program. [5 marks]

(*b*) Compare and contrast the approaches of using a single mutex to guard access to an entire data structure and using individual mutexes on each unit of storage within the data structure. [5 marks]

(*c*) Consider a queue data structure, based on a linked list. The operations `pushTail` and `popHead` are to be provided and it should be possible to execute both concurrently whenever doing so would be safe (but not when it would be unsafe). Provide a Java implementation of the data structure and concurrency control mechanism, including the methods to push new items on the tail of the queue and to pop items from the head. [10 marks]

2008 Paper 5 Question 4

Concurrent Systems and Applications

(*a*) A web server is an application that listens for incoming network connections on TCP port 80. Once a connection is established, the task of processing client requests and sending replies can be handled by an instance of a `Worker` class which you may assume already exists. `Worker` implements the `java.lang.Runnable` interface and has an accessible constructor that takes as argument a `java.net.Socket` object representing the network connection to a client.

Provide the Java code for a webserver which, upon start-up, attempts to listen on TCP port 80 and starts a new `Thread` running a new `Worker` for every connection. Your program should print helpful error messages indicating the likely cause of problems when it is unable to proceed as expected. [10 marks]

(*b*) A busy web server might expect to handle concurrent requests to read and update some shared data and could use Timestamp Ordering (TSO) to enforce isolation between concurrent transactions.

(*i*) Explain how TSO enforces isolation. [5 marks]

(*ii*) Is TSO appropriate for a web server application? Explain your reasoning. [5 marks]

2008 Paper 6 Question 4

Concurrent Systems and Applications

(*a*) What are *Class Literals* in the context of generics in Java? [2 marks]

(*b*) Under what circumstances might *type erasure* occur in a Java program that uses generics? Illustrate your answer with sample Java code. [4 marks]

(*c*) Consider the two overloaded methods shown below. Is it valid to have these overloaded methods in a Java class? Explain your answer.

```
public static <T extends Long>
 void myMethod(List<T> x)
{
 // ...
}

public static <S extends String>
 void myMethod(List<S> x)
{
 // ...
}
```

[5 marks]

(*d*) Consider two class definitions: `A`, and `B` which extends `A`. In the context of the relationship between the types `A[]` and `B[]`, explain the difference between covariance and contravariance. Is Java covariant or contravariant? [4 marks]

(*e*) If a field is declared with `protected` access, from where can it be accessed? Is there any situation in which a `private` field defined on some class `A` can be accessed but a `protected` field defined on `A` cannot? [5 marks]

2009 Paper 3 Question 3

Concurrent Systems and Applications

(*a*) Define the ACID properties of transactions. [4 marks]

(*b*) In the context of transactions, what does it mean to say that an execution schedule is *serialisable*? [1 mark]

(*c*) Transaction systems can enforce either *strict* or *non-strict* isolation.

(*i*) What is the difference between the two? [1 mark]

(*ii*) Why do many systems enforce only non-strict isolation? [1 mark]

(*iii*) When using non-strict isolation, executing transactions may experience *lost updates*, *dirty reads* or *unrepeatable reads*. For *each* of these, describe with the aid of an example what it means. [1 mark each]

(*d*) Compare and contrast *two-phase locking* (2PL) and *optimistic concurrency control* (OCC) as means of providing isolation. You should discuss the level of isolation achieved, the degree of concurrent execution enabled, the behaviour in case of aborts, and the likely performance in the presence of contention. [10 marks]

2009 Paper 5 Question 5

Concurrent Systems and Applications

(*a*) Reflection.

(*i*) Give Java code fragments demonstrating *two* different ways of obtaining a `Class` object that describes an array of `java.lang.String`s. [2 marks]

(*ii*) Given an object `x`, write a Java expression that uses reflection to create a new object of the same datatype as `x`. [2 marks]

(*iii*) The `clone()` method creates an exact copy of an object, including all of its fields. Briefly describe how you might implement this functionality using reflection, ignoring inherited fields. Assume that the object is not an array, has a zero-argument constructor, and contains only primitive fields. (You need not give code for an actual complete implementation.) [4 marks]

(*b*) Generics.

(*i*) Suppose a class `B` is a subclass of `A`. Is the class `Set<B>` a subclass of `Set<A>`? Explain why or why not, with regard to type safety. [2 marks]

(*ii*) The default clone method returns an `Object` that must be cast to the correct type. Using generics, give a declaration of a static `myclone` method that takes a single argument of any type and returns an object of the same type. [2 marks]

(*iii*) The `Contraster` interface is used to compare two objects. Its declaration is:

```
interface Contraster<T> {
  boolean greaterThan(T obj1, T obj2);
}
```

Suppose we want to declare a class `SortedList<E>` whose constructor takes a single `Contraster` argument that will be used to compare its elements. Give a declaration for the constructor that permits the choice of contraster implementation to be as general as possible, and explain your reasoning. [4 marks]

(*c*) Reference objects.

The `get()` method of the `PhantomReference` class always returns `null`. Why is this so, and why must a `PhantomReference` always be used together with a `ReferenceQueue`? [4 marks]

2009 Paper 5 Question 6

Concurrent Systems and Applications

(*a*) The following method is intended to return unique integer values to callers:

```
volatile int x = 0;
int getNext() {
  x = x + 1;
  return x;
}
```

(*i*) Two threads call `getNext` concurrently on the same object. Explain how both threads can receive the result `1`. [1 mark]

(*ii*) Explain the semantics of the `synchronized` keyword in Java, and illustrate this by correcting `getNext` (you may ignore the possibility of integer overflow). [6 marks]

(*iii*) Explain the meaning of the `volatile` modifier. Explain whether or not you need to use it with your new implementation of `getNext`. [2 marks]

(*b*) The following method is intended to implement a *barrier* for synchronization between four threads. The first three threads to call the `barrier` method are meant to block. These threads are all unblocked when the fourth call is made.

```
int barrierCount = 0;
void synchronized barrier() throws InterruptedException {
  barrierCount ++;
  if (barrierCount < 4) {
    wait();
  } else {
    notifyAll();
} }
```

(*i*) A programmer finds that some threads return early, although there have been fewer than four calls to `barrier`. How can this happen? [2 marks]

(*ii*) Rewrite `barrier` so that threads wait correctly. [2 marks]

(*iii*) Explain whether or not it would be correct to use `notify` in place of `notifyAll` in your solution. [2 marks]

(*iv*) If a thread is *interrupted* while waiting within `barrier` then the call to `wait` will fail with `InterruptedException`. Rewrite `barrier` so that, if one thread is interrupted when using a barrier, then any future (or concurrent) calls to that barrier will also fail with `InterruptedException`. [5 marks]

2010 Paper 3 Question 9

Further Java

Fellows at Norisbon College dine at a circular table on which there is a single fork between each Fellow. Fellows either eat or think, and always start dinner thinking. To eat, a Fellow first picks up the fork immediately to his left and, once successful, picks up the fork immediately to his right. When a required fork is not on the table, the Fellow waits, neither eating nor thinking, until the fork is returned to the table. After eating, a Fellow returns both forks to the table. No cutlery is required to think.

Your task is to model the above scenario in Java.

(*a*) Write a class called `Fork` with two public methods, `pickUp` and `putDown`. The methods should take no arguments and return no result. An instance of `Fork` should act as a lock to prevent concurrent access. In other words, once `pickUp` has been called, all further calls to `pickUp` should block until `putDown` is called; when `putDown` is called, one caller (if any) who is blocked should proceed. [7 marks]

(*b*) Write a class called `Fellow` which inherits from the `Thread` class and implements the abstract method `run`. The `Fellow` class should have a single constructor which takes two `Fork` objects, one representing the fork to the Fellow's left, and one to the right. When run, an instance of `Fellow` should think for ten seconds, eat for ten seconds and think for ten seconds before terminating. [7 marks]

(*c*) Describe when and why your implementation may suffer deadlock. [2 marks]

(*d*) By altering the order in which the forks are picked up, describe how you would modify your implementation so that it does not suffer deadlock. [4 marks]

2010 Paper 5 Question 9

Further Java

Consider the following client program extract:

```
1  Socket s = new Socket("localhost",10000);
2  ObjectInputStream ois = new ObjectInputStream(s.getInputStream());
3  Object o = ois.readObject();
4  Class c = o.getClass();
5  for(Field f :  c.getDeclaredFields())
6    System.out.println(f.get(o));
7  c.getMethod("run").invoke(o);
```

(*a*) Describe the execution of this extract, assuming that no exceptions are thrown. [5 marks]

(*b*) Identify **five** distinct exceptions that may occur during execution of the client program extract. Your answers should include the line number at which the exception would be thrown and a brief description of the problem which would cause it. General virtual machine errors such as OutOfMemoryError or StackOverflowError should not be included in your answer. [2 marks each]

(*c*) Write a server program extract that is capable of communicating successfully with the client extract shown above. You should assume the existence of an Object variable called **sendMe** which contains the object to be sent to the client. You do not need to write any error handling code. [5 marks]