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. Consider a time-sharing CPU that has four devices attached to it (A, B, C, and D) and a number of users. Of the jobs completing at the CPU 3.125% went to the users, 25% to device A, 37.5% to device C and the remainder to device D. All jobs at device A complete to device B, and all jobs at device C complete to device A. Jobs at device B have probability 0.6 of completing to device C rather than directly to the CPU. Jobs at device D complete directly to the CPU. The service times were 3ms for device A, 7ms for device B, 9ms for device C and 16ms for device D. The average service time per visit to the CPU was 5ms, and the average user think time 5 seconds.

   a) Create and label a diagram showing the system [6]

   b) For each job, what are the visit ratios for the CPU and each of the four devices? [7]

   c) What is the total service demand for the CPU and each of the four devices? [6]

   d) What is the bottleneck device? [4]

   e) Sketch and label a graph showing response time versus system load. What is the value of N^*? [5]

   f) If the utilisation of device A is 8%, what is the average response time if there are 10 users in the system? [5]

2.

   a) What role does an *authentication server* play in secure systems? [3]

   b) *Smallman* is a small manufacturing business employing 50 people with a small LAN supporting order processing and accounting functions. It is considering allowing connection to the internet for the purposes of both marketing and to allow online payment by its customers. What threats will it face by so doing? [10]

   c) Soon after this, Smallman acquires a partner, Bigman, and they decide that all their communications should be encrypted using a secret-key algorithm. However, the government has dictated that all keys used for encrypted communications must be generated by and stored in government-authorised authentication servers. Describe in detail a protocol for obtaining keys that will allow Smallman and Bigman to communicate securely within the law. [20]

[TURN OVER]

3.

a) What conditions must exist for a system to be *deadlocked*? For each of these, argue that the absence of the condition means that deadlock cannot occur. [8]

b) In an effort to avoid problems with deadlock, a system administrator has devised a policy in which processes may request and release resources at any time subject to the following behaviour. A process may be in one of three states:

- Running, in which case it has all the resources it needs at present. It may release resources whilst running.
- Blocked, in which case a process is waiting for resources that are currently unavailable.
- Requesting, in which case a running process is asking for more resources. In the case that the request can be satisfied, the process continues running, else it becomes blocked. If a request for resources cannot be satisfied immediately, as resources are not currently available, then the operating system will examine the list of blocked processes to determine whether it is possible to satisfy the request by preempting resources that are held by blocked processes. If it is possible to do this, then the resources are taken away from the blocked processes and the vector of resources for which they are waiting is increased.

i) What major assumption is implicit in the above scheme? Is this assumption reasonable, in general?

ii) Can deadlock occur in this case. If so, give an example, if not say which of the conditions you identified in part (a) cannot occur.

iii) Can indefinite blocking occur? Justify your answer. [9]

c) The two major approaches to dealing with deadlocks are *deadlock detection* and *deadlock avoidance*.

i) What factors must be considered in deciding whether to employ deadlock avoidance as opposed to deadlock detection?

ii) Outline an algorithm for deadlock avoidance and provide an argument that shows that it is not possible for a system to deadlock if such an algorithm is used. [16]

[CONTINUED]

4.

a) What three conditions must be satisfied by any solution to the mutual exclusion problem and why? [6]

b) The following is a published proposed solution to the mutual exclusion problem for two processes (Hyman 1966). Explain its operation and provide an argument that shows that it either does or does not satisfy the requirements above.

Global variables

```
int c1=1, c2=1, turn=1
```

Process 1

```
while (true) {
{ c1 = 0
  while (turn == 2)
  { while (c2 == 0) { skip }
    turn = 1
  }

  CRITICAL SECTION

  c1 = 1

  NON-CRITICAL SECTION
}
```

Process 2

```
while (true) {
{ c2 = 0
  while (turn == 1)
  { while (c1 == 0) { skip }
    turn = 2
  }

  CRITICAL SECTION

  c2 = 1

  NON-CRITICAL SECTION
}
```

[12]

c) The multiple readers, single writer (MRSW) problem can be expressed as having the following safety properties:

- Readers may execute concurrently with one another
- Writers and readers must not execute concurrently
- Only one writer must execute at a time

Complete the following monitor to implement a solution to the above problem, adding both code and variable definitions where necessary.

[TURN OVER]
[Question 4(c) continued over]

[Question 4(c) continued]

```
// Class variables:
// Insert whatever class variables you need here.
//

MRSW::start_read()
{ // Put code here that is called before a reader
  // starts to read
}

MRSW::end_read()
{ // Put code here that is called by a reader when
  // it has finished reading
}

MRSW::start_write()
{ // Put code here that is called before a writer
  // starts to write
}

MRSW::end_write()
{ // Put code here that is called by a writer when
  // it has finished writing
}
```

Demonstrate either that your solution is fair or that there is a possibility of starvation. State any assumptions you make. [15]

5.

a) In what states can a process be and under what conditions are the transitions between the states taken? [6]

b) Explain why the queue of runnable processes in an operating system is singly linked whilst some of the device queues are doubly linked. [6]

c)

i) What is a process control block and what does it contain?

ii) Describe in detail the actions taken by an operating system as the result of a context switch [11]

d) Consider a variant of the round robin scheduling scheme in which entries in the queue of runnable processes are pointers to the PCBs, rather than PCBs themselves.

i) What would be the effect of putting two pointers to the same process in the queue? What advantages and disadvantages does this approach have?

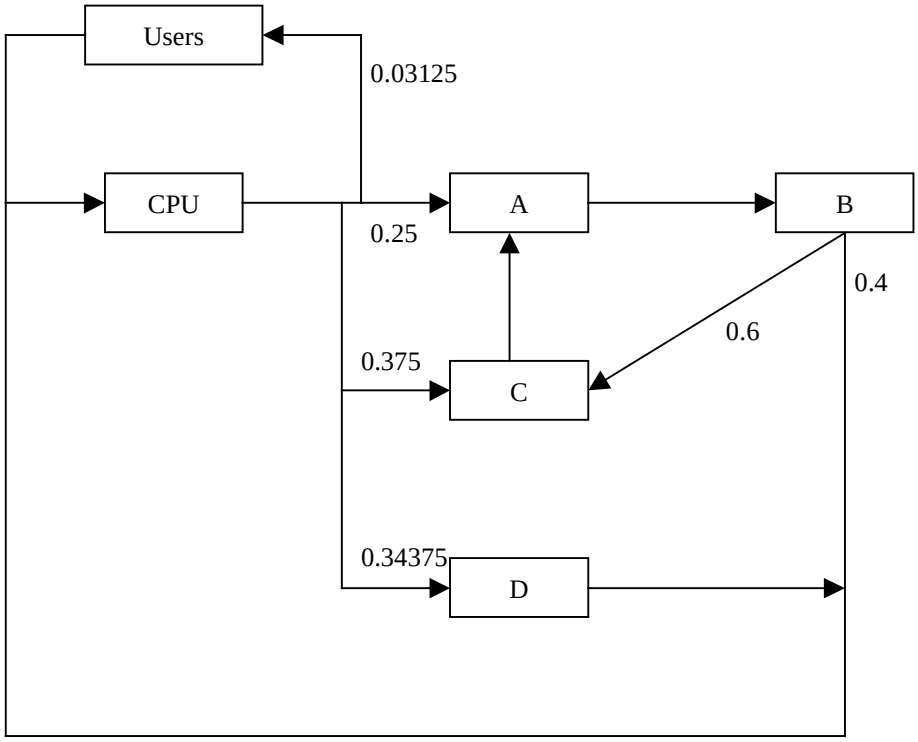
ii) How would you modify the basic round robin algorithm to achieve the same effect? [10]

[END OF PAPER]

D7 Answers

1.

a)



b) We have:

$0.03125\ V_{CPU} = 1$

$V_A = 0.25V_{CPU} + V_C$

$V_B = V_A$

$V_C = 0.375\ V_{CPU} + 0.6\ V_B$

$V_D = 0.34375\ V_{CPU}$

From which we derive:

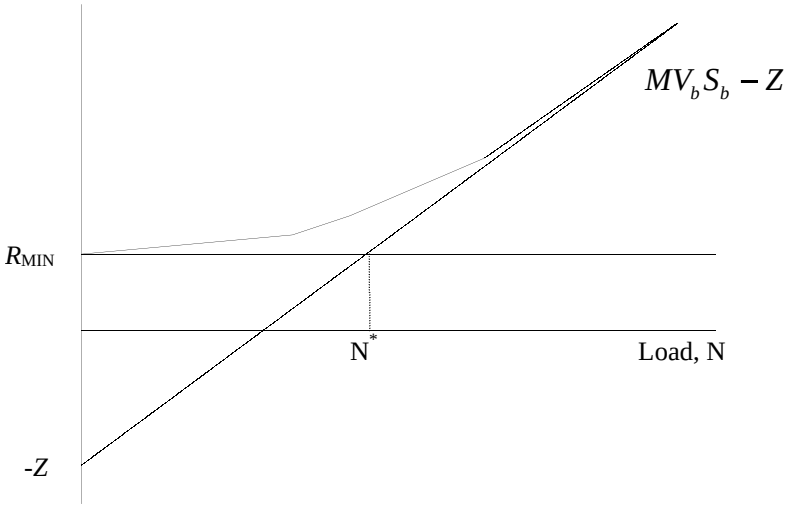
	CPU	**A**	**B**	**C**	**D**
$\mathbf{V_i}$	32	50	50	42	11

c) $D_i = V_i\ S_i$ so:

	CPU	A	B	C	D
$\mathbf{D_i}$	0.160	0.150	0.350	0.378	0.176

d) Bottleneck device has max D_i, so C is bottleneck.

e)



In this case $R_{MIN} = \Sigma D_i = 1.214$ s

Z = think time = 5 s

$$\frac{N^*}{R_{MIN} + Z} = \frac{1}{V_b S_b}$$

$$N^* = \frac{R_{MIN} + Z}{V_b S_b}$$

Thus, in this case, $N^* = 16.44$ users

f) $U_i = X D_i$, so considering A, X = 0.5333 jobs/s. Now, using R=M/X – Z, we get R=13.75s.

2.

a) An authentication server is a trusted third party that holds details of encryption keys securely. It authenticates public keys to other principals in the case of PK systems, and may generate session keys in the case of SK systems.

b) Threats are various. Some are pre-existing, like the risk of sabotage by employees; others are to do with the new networking aspect and answers should include at least the following:

- *Passive tap*

  i.e. eavesdropping, where one can see the traffic passing across a network, but cannot introduce traffic directly onto that network oneself. However, one can make use of that information elsewhere so, for example, if credit card details are recorded, they can be used to buy other services.

- *Active tap*

  With an active tap, one can both read and introduce traffic onto a network. This is more insidious, since one can pretend to be some other customer, using their stolen details, or steal business from/discredit Smallman by pretending to be them. The preferred solution to both active and passive attacks is to use encryption.

- *Denial of service*

  In this form of attack, someone introduces traffic onto a network that is properly addressed and so forth, but which contains meaningless junk. It takes the recipient of the traffic a little time to work out that the traffic is bad, and during this time it cannot be servicing legitimate traffic. If a particular service is flooded with such traffic, then it may be unable to give any reasonable level of service to legitimate traffic.

- *Faking/replay*

  Even if the traffic two parties send one another is encrypted, I may be able to guess the structure of the information and replace a legitimate message with one which has the correct structure but which has altered fields. This is known as faking. The most common form of this is replay, in which an interloper simply records then replays a message from one legitimate party to another. Say, for example, it is known that Smallman have just transferred an amount of money from their bank account into another, and that message has been captured. Then, even though the message was encrypted, it can be copied it and replayed several times, effecting several transfers.

- *Traffic analysis*

  The mere fact that A is sending messages to B is information, even if one cannot see what those messages are. On the internet, it is relatively easy to get information about where a particular message originated and to what machine it is destined. Thus, for example, if there is a lot of traffic going between Smallman and a particular customer, then it could reasonably be guessed that they have a special interest in that customer at present, even though it might not be possible to tell exactly what they are doing.

c) This is looking for the Needham-Schroeder protocol:

i) $A \rightarrow AS$: A, B, I_a

A sends a request to the authentication server saying, 'I am A and I want to talk to B'. It also passes a nonce I_a, which is just a random number which is different every time this protocol is run.

ii) AS $\rightarrow$ A: $\left\{I_a, B, K_S, \{K_S, A\}_{K_B}\right\}_{K_A}$

The authentication server replies, with a message encrypted with a key known only to it and A. Thus A is the only person able to read the contents of this message. In the message are:

- *the nonce sent in 1. to indicate that this message is not a replay of an earlier one.*
- *A session key K_S, which will be used by A and B to transfer data. A new session key is generated each time to reduce the chances of it being broken.*
- *The last part of the message, encrypted with B's key is a token, containing details of the session key and the person initiating the protocol. A just sees a sequence of bits here; since A does not have B's key, then it cannot decrypt the message. It is used below.*

iii) A $\rightarrow$ B: $\{K_S, A\}_{K_B}$

A sends B the token it got. When B receives it, then it can get access to the session key and the identity of the person initiating the protocol.

iv) B $\rightarrow$ A: $\{I_B\}_{K_S}$

B sends a nonce to A encrypted with the session key. This step ensures that A knows that B knows the session key and, since this was recently generated, that B is currently alive.

v) A $\rightarrow$ B: $\{f(I_b)\}_{K_S}$

A replies to B with some function of B's nonce (e.g. I_b - 1), encrypted with the session key. This ensures that B knows that A is alive and that the message it got in 3 was not simply a replay of an earlier message.

vi) .... DATA TRANSFER ....

Finally, messages are transferred, encrypted with the session key. A knows that B is alive, B knows that A is alive, and they both know that the only other party to the key is the authentication server, provided that neither of them has leaked the information

3.

a) Mutual exlusion, hold and wait, no preemption, circular waiting.

b)

i) That resources are preemptable. And no, it isn't reasonable in general; most resources aren't preemptable.

ii) No it can't because there is the ability to preempt resources.

iii) Yes, a process can continually have its resources preempted by other processes and may, therefore starve. This will be considerably worse for long-running processes which use many resources, since they will typically block more often and will, therefore, be subject to a greater chance of having their resources preempted.

c)

i) Avoidance costs up front, every time an allocation is attempted. Detection only costs when you run the algorithm, but there are two extra costs of detection – the cost of having a deadlock without realising it, and the cost of resolving it once detected. The longer the undetected deadlock, the more processes become involved, and the more expensive it is to resolve. Running the algorithm more often reduces the problem, but costs more up-front. Detection has the other nasty property that once a deadlock is detected, something must be done about it; processes need to be terminated and rerun – not simple if we want to avoid inconsistency. Probably OK in a transactional environment though.

ii) The Bankers algorithm is what is being asked for here. The proof will be informal, and should involve arguing that the algorithm takes a system from one safe state to another, and that deadlock is not possible from a safe state. Consequently, no deadlock is possible is the system starts in a safe state.

4.

a) Mutual exclusion (no two processes in CS at once), bounded waiting (no process should wait forever), progress (no process in the remainder section can block something in the CS).

b) Doesn't work – If P_1 enters its CS repeatedly whilst P_2 is looping on the inner while statement, then P_2 can be starved. Tough question for those without vast amounts of experience in the area.

c)

```
// Class variables:
int readers = 0
bool writing = false

condition oktoread
condition oktowrite

MRSW::start_read()
{ if (writing)
    oktoread.wait()
  readers++
}

MRSW::end_read()
{ if (--readers == 0)
    oktowrite.signal()
}

MRSW::start_write()
```

```
{ if (readers > 0 || writing)
    oktowrite.wait()
  writing = true
}

MRSW::end_write()
{ writing = false
  oktoread.signal()        // Starve writers
}
```

They haven't done this with condition variables. Above solution starves writers. Lots of extra credit for anything which is fair or which is a good stab at it. Assumed Hoare style monitors above; else include mutexes in each of entry routines.

5.

a) Bookwork. See Silberschatz and Galvin p. 90.

b) Head of run queue is always next thing to run; we only need to insert in the right place. Device completions may be misordered relative to request times for certain devices (e.g. disk) – easier to remove a PCB if one has a doubly linked list in this case.

c)

i) Should include process state, pid, PC, registers, priority, MMU info, open files, accounting info, parent/children, etc.

ii) Should relate the use of all the info given in the first part. Depending on, e.g. the memory management scheme used, this might involve restoring page tables, and flushing caches as well as reloading registers, etc..

d)

i) Process would run twice in any round. Disadvantages: pain when going to do I/O – must search run queue for other pointer, generally poor structure -- complex. Advantages: allows for processes to have higher priorities.

ii) Either add a field saying that quantum should be doubled, or use a multilevel scheme or put PCB back in the middle of the run list.