UNIVERSITY OF LONDON

IMPERIAL COLLEGE OF SCIENCE, TECHNOLOGY AND MEDICINE

EXAMINATIONS 1999

BEng Honours Degree in Computing Part I
MEng Honours Degrees in Computing Part I
BEng Honours Degree in Information Systems Engineering Part I
MEng Honours Degree in Information Systems Engineering Part I
BSc Honours Degree in Mathematics and Computer Science Part II
MSci Honours Degree in Mathematics and Computer Science Part II
for Internal Students of the Imperial College of Science, Technology and Medicine

This paper is also taken for the relevant examinations for the Associateship of the City and Guilds of London Institute

PAPER 1.5 / I 1.5 / MC 2.5

OPERATING SYSTEMS I

Friday, April 30th 1999, 2.30 – 4.00

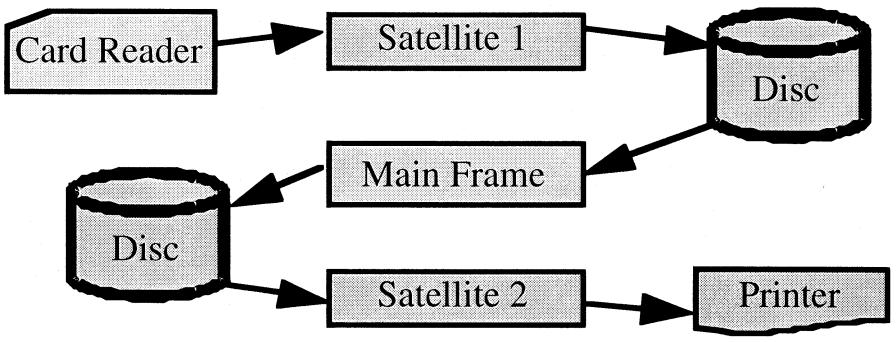
Answer THREE questions

For admin. only:
paper contains 4 questions

Section A *(Use a separate answer book for this Section)*

1a Briefly describe *four* ways in which 3rd generation computer systems were an advance on the 2nd generation. (One sentence for each should be enough.)

b A 2nd generation directly coupled off-line batch system used two satellite computers to handle slow peripheral I/O such as card reader input and printer output. Each satellite communicated with the main frame via a disc.



A batch of 3 jobs has input, compute and output times (in seconds) as follows:

Job	Input	Compute	Output
1	5	20	10
2	10	20	5
3	10	15	10

The I/O times given are for the slow peripherals only. Disc transfer rates are 10 times as fast.

i) What is the turnaround time in minutes for job 3? How much computer time (mainframe + satellites) is actually spent on it?

ii) Suppose the system is continuously executing batches similar to the one described. What is the throughput in jobs per minute? Explain why there is no need to include an interbatch gap in the calculation.

iii) An early 3rd generation system replaces the satellites by spooler processes on the mainframe. What is the best throughput that can be achieved? (Assume that I/O and compute times remain the same.) What factors might affect whether this can be achieved?

The two parts carry, respectively, 40%, 60% of the marks.

2a i) What is meant by *mutual exclusion* for two concurrent processes? Explain clearly what things are excluding each other. How can a *lock* be used to enforce mutual exclusion?

ii) Give an example to show why testing and setting a lock has to be a single indivisible operation.

iii) Why does the Simple Kernel of the lecture course not use locks? What are the situations in which they are needed?

b Two concurrent processes P1 and P2 each have a critical region consisting of an instruction to add 1 to a shared variable x. Describe what extra code is needed to protect these critical regions using

i) a lock

ii) a semaphore

iii) a monitor

Remember to include data declarations and initialization. You may use any suitable language or pseudocode.

The two parts carry, respectively, 55%, 45% of the marks.

Turn over ...

Section B *(Use a separate answer book for this Section)*

3 This question concerns the *Simple Kernel* that was developed in the course, a multiprocess operating system that supports priority preemption, semaphores and timed delay.

a i) Draw a state transition diagram to show the different states that a process might be in, and the possible transitions between states. Label each transition with the events that cause it.

ii) What information does the operating system need to preserve for each process?

iii) How is a *context switch* is performed? Give your answer in general terms, without going into details that would vary from machine to machine.

b Suppose the kernel is supporting three processes, H, L and I, with priorities high, low and idle respectively. I runs an infinite idling loop, while H and L run code as follows:

```
H:  Delay(1000)        L:  P(s)
    V(s)                   V(s)
    P(s)
    V(s)
```

Initially, all three processes are ready and the semaphore s has value 0.

Complete a history of the execution of these processes in the following format:

Running process	Operation	Ready queue	Delay queue	s queue	s value
	Initially	H, L, I			0
H	Delay(1000)	L, I	H		0
:	:	:	:	:	:
:	:	:	:	:	:

(The first running process is H because it has highest priority. It calls Delay(1000) and as a result goes on the delay queue. L and I are still ready, and s still has value 0. The four columns on the right show the system after the operation, immediately before any rescheduling (i.e. call of Dispatch).)

c The Simple Kernel uses a simple priority-preemption scheduling strategy. Explain why this is unsuitable for a time sharing system used to support multiple online users.

Outline a modification to the Simple Kernel by which it could incorporate time slicing.

The three parts carry, respectively, 40%, 30%, 30% of the marks.

4a Briefly explain what is meant by each of the following terms:

i) Overlay

ii) Store Fragmentation

iii) Thrashing

b A process executing on a system supporting paging makes references to the following page numbers:

0 1 2 3 0 1 4 0 1 2 3 4 2 0 1 2 0 3 4 0

For each of the cases shown below, specify which references cause page faults, and where a page must be removed from main memory, give its identity.

i) Page replacement policy is First In First Out (FIFO); main memory contains 3 page frames.

ii) Page replacement policy is east Recently Used (LRU); main memory contains 4 page frames.

You can give your answer in a form such as the following example (for part i):

```
                0 1 2 3 0 1 4 0 1 2 3 4 2 0 1 2 0 3 4 0
Faults:         F F F F F ...
Replaced page:      0 1 ...
```

c The space overheads introduced by paging arise from the following sources:

- The page table takes up space.
- A process is unlikely to fit into an integral number of pages, and therefore, on average, half the last page will be empty.

Show that if the average size of a process is s bytes, and that the size of a page table entry is e bytes, then the space overhead is given by:

$$\text{Space overhead} = \frac{se}{p} + \frac{p}{2}$$

where p is the size of a page.

Show that the page size which minimizes the space overhead is given by:

$$p = \sqrt{2se}$$

(*Hint:* differentiate with respect to p.)

What is the optimum page size if the average size of a process is 768K bytes and the space required for a page table entry is 8 bytes? What size is likely to be chosen in practice?

The three parts carry, respectively, 30%, 30%, 40% of the marks.

End of paper