CLASS TEST

S.No.:03SK_CDEFGHIJKL_221024

MADE EASY
India's Best Institute for IES, GATE & PSUs

Delhi | Bhopal | Hyderabad | Jaipur | Pune | Kolkata

Web: www.madeeasy.in | **E-mail:** info@madeeasy.in | **Ph:** 011-45124612

Operating System

COMPUTER SCIENCE & IT

Date of Test : 22/10/2024

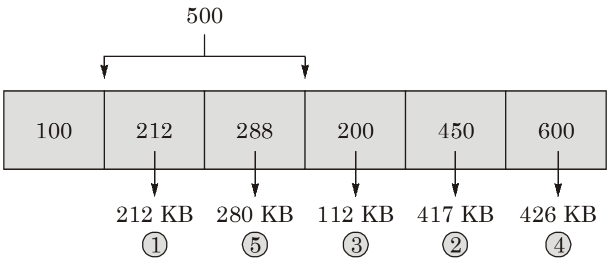
ANSWER KEY

1.	(b)	7.	(c)	13.	(c)	19.	(a)	25.	(b)
2.	(d)	8.	(a)	14.	(a)	20.	(b)	26.	(d)
3.	(a)	9.	(c)	15.	(b)	21.	(b)	27.	(d)
4.	(b)	10.	(a)	16.	(d)	22.	(d)	28.	(d)
5.	(d)	11.	(b)	17.	(d)	23.	(d)	29.	(b)
6.	(b)	12.	(c)	18.	(a)	24.	(d)	30.	(b)

DETAILED EXPLANATIONS

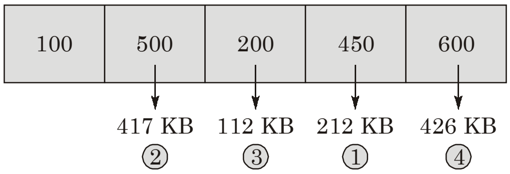
1. **(b)**

1. Using First Fit:



All request will be fit in memory.

2. Using Best Fit:



Request 280 KB will not fit in memory.

2. **(d)**

Suppose, page size = P Bytes

Process overhead = Page table overhead + Overhead due to internal fragmentation

Page table overhead = Number of pages per process × Page table entry size

$$= \left(\frac{\text{Process size}}{\text{Page size}}\right) \times 4\,\text{B} = \frac{32\,\text{MB}}{\text{P B}} \times 4\,\text{B}$$

Average overhead due to internal fragmentation $= \frac{0+P}{2} = \frac{P}{2}$

0 = Minimal internal fragmentation

P = Maximum internal fragmentation

$$\text{Overhead is paging} = \frac{32\text{ M} \times 4}{P} + \frac{P}{2}$$

To minimize overhead i.e. take differentiation with respect to ‘P’.

$$\frac{-128\text{ M}}{P^2} + \frac{1}{2} = 0$$

$$P^2 = 2 \times 128\text{ MB}$$

$$P = \sqrt{256\text{ MB}}$$

$$P = 16\text{ KB}$$

3. **(a)**

Size of virtual addresses = 48 bits

Page size = 8 KB

Page offset = 2^{13} = 13 bits

Number of bits used for indexing = 48 – 13 = 35 bits

$$\text{Number of set} = \frac{256}{4} = \frac{2^8}{2^2} = 2^6 = 64 \text{ require 6 bits}$$

Total tag bits = 35 – 6 = 29 bits.

4. **(b)**

Option (a) and (c) do not satisfy progress condition since (a), (c) leads to deadlock. Option (b) is correct to satisfy progress condition.

5. **(d)**

Considering each option of List-I:

- Mutual exclusion can be solved by spooling everything.
- Hold and wait can be solved by requesting all the resources before hand.
- No preemption can be solved by taking request away or by releasing all resources.
- Circular wait can be solved by numbering the resources in some order.

6. **(b)**

Gantt chart:

P_1	P_2	P_3	P_2	P_4	P_1

0 2 4 5 7 11 16

Processes	Arrival Time	Burst Time	Waiting Time
P_1	0	7	16 – 7 = 9
P_2	2	4	7 – 6 = 1
P_3	4	1	5 – 5 = 0
P_4	5	4	11 – 9 = 2
			Average = 3

7. **(c)**

Initial value of semaphore = x

3rd step: 22 V operations = $x + 22$

2nd step: 12 P operations = $x + 22 - 12 = x + 10$

1st step: 3 V operations = $x + 10 + 3 = x + 13$

As per question, $x + 13 = 20$

$\Rightarrow$ $x = 7$

8. **(a)**

S_1 **:** If the time quantum of the Round-Robin scheduling algorithm is larger than the longest CPU burst time of processes. Then, it will works as FCFS only. Hence, this is not always correct that RR give better performance compared to FCFS. S_1 incorrect

S_2 **:** An advantage of system call to provide the interface between program and operating system.

9. **(c)**

The exec( ) system call replaces the current process image with a new process image. It loads the program into the current process space and runs it from the entry point.

Kill( )

In Unix and Unix-like operating systems, kill is a command used to send a signal to a process. By default, the message sent is the termination signal, which requests that the process exit. But kill is something of a misnomer; the signal sent may have nothing to do with process killing.

10. **(a)**

- S_1 is correct, if the file is very-very large indexed allocation will fail to accommodate then we will use unix inode.
- S_2 is false because metadata does not include the actual data or contents of files, it only includes the file system structure.

11. **(b)**

Effective Memory Access Time = (1 – P) × Memory Access Time + P(Non modified% × Page fault service time + Modified% × Page fault service time with page modified)

= 0.80 × 120 nsec + 0.20 (0.15 × 800 nsec + 0.85 × 950 nsec)

= 96 nsec + 0.20 (120 nsec + 807.5)

= 96 nsec + 185.5 nsec

= 281.5 nsec

12. **(c)**

- In given solution, atleast one process will enter into critical section i.e. process which arrive late enter first. So, not deadlock.
- Only one process can enter into critical section at any time because of checking of turn variable in while loop.

13. **(c)**

Gantt chart:

P_1	P_2	P_3	P_1	P_2	P_1	P_4	P_5	P_1	P_4	P_5	P_1	P_5	P_1	P_5

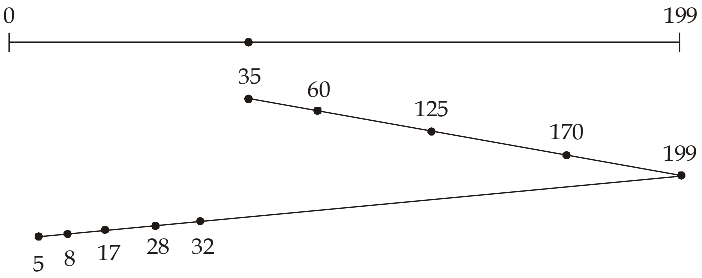
0 15 30 40 55 70 85 100 115 130 135 150 165 180 185 190

Processes	Arrival Time	Burst Time	Turn Around Time
P_1	0	80	185
P_2	10	30	60
P_3	10	10	30
P_4	80	20	55
P_5	85	50	105
			Average = 87 msec

14. (a)

Cylinder number : 5, 17, 60, 125, 28, 170, 8, 32

In SCAN algorithm disk head move till the last cylinder with servicing the requests then change the direction move till inner most cylinder with servicing requests.



Total moves = (199 – 35) + (199 – 5) = 358

15. (b)

Using LRU:

9	8	7	9	3	0	2	9	8	3	9	2	0	9
					0	0	0	0	0	0	0	0	0
				3	3	3	3	3	3	3	3	3	3
		7	7	7	7	7	7	8	8	8	8	8	8
	8	8	8	8	8	2	2	2	2	2	2	2	2
9	9	9	9	9	9	9	9	9	9	9	9	9	9
F	F	F		F	F	F		F					

= 7 faults

By using FIFO:

9	8	7	9	3	0	2	9	8	3	9	2	0	9
					0	0	0	0	0	0	0	0	0
				3	3	3	3	3	3	3	3	3	3
		7	7	7	7	7	7	8	8	8	8	8	8
	8	8	8	8	8	8	9	9	9	9	9	9	9
9	9	9	9	9	9	2	2	2	2	2	2	2	2
F	F	F		F	F	F	F	F					

= 8 faults

16. (d)

Process	Allocated			Need		
	R_1	R_2	R_3	R_1	R_2	R_3
P_1	1	0	1	0	0	1
P_2	1	0	0	0	1	1
P_3	0	1	1	0	1	0
P_4	0	0	0	0	1	1

$$\text{Available} = \begin{matrix} R_1 & R_2 & R_3 \\ (0 & 1 & 0) \end{matrix}$$

P_3 need = (0, 1, 0) so P_3 can execute, after that available resources (0, 2, 1).

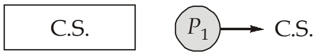
Now, anyone of P_1 or P_2 or P_4 can execute.

17. (d)

$$S_1 = 4, S_2 = 3, S_3 = 2, S_4 = 1$$

Suppose P_1 first enters critical section.

$P(S_1)$;
$P(S_2)$;
$P(S_3)$;
$P(S_4)$;



Now, if any other process will try to enter C.S.
It will be suspended because $S_4 = 0$ (currently).
Now, when P_1 will exit C.S, it will run $V(S_4)$ and invoke the suspended process.
Suspended process will get the next chance to enter to C.S.
There for mutual exclusion, progress and bounded waiting is satisfied.

18. (a)

When both process P_0 and P_1 executing concurrently P_0 and P_1 make turn_0 = true and turn_1 = true respectively process P_0 execute while loop before executing turn_0 = false it preempted and P_1 start executing it make turn_1 = false and preempted. P_0 make turn_0 = false, both process enter into critical section.
Therefore it does not ensure mutual exclusion.

19. (a)

- Some system calls are blocking, while some are non-blocking. When a process makes a blocking system call, the process cannot proceed further immediately, as the result from the system call will take a little while to be ready.
- Context switching happens from the Kernel mode of one process to the Kernel mode of another: the scheduler never switches from the user mode of one process to the user mode of another. A process in user mode must first save the user context, shift to Kernel mode, save the Kernel context, and switch to the Kernel context of another process.

20. (b)

	P_2	P_1	P_3	P_4	P_0	
0	1	5	8	10	15	21

$$\text{Waiting time} = \text{Turn around} - \text{Execution (Burst time)}$$

$$\text{Average waiting time} = \frac{\sum_{i=0}^{n} \text{Waiting time of } P_i}{\text{Total number of processes}}$$

$$= \frac{10+3+0+2+2}{5} = \frac{17}{5} = 3.4 \text{ ms}$$

21. (b)

There are these fork calls. So, $2^n - 1$ child process will be created i.e. 7.
And we have to include parent process also, so total 8 processes are there.

22. (d)

23. (d)

The purpose of dynamic loading is optimal utilization of memory.
Option (a) all routine are kept on disk in a relocatable load format.
Option (b) is property of static loading

24. (d)

(a) The CPU utilization increases as the degree of multiprogramming increase up to threshold (some limit), after that utilization start decreasing.

(b) Test and set is a special hardware instruction that does two operations **atomically** but does not to avoid busy wait.

(c) User level threads are faster to switch among them than kernel level thread since user level thread have less context.

25. (b)

Page size = 1 KB = 2^{10} bytes
Virtual address space = 4 GB = 2^{32} bytes

← 32 →

22	10
	offset

22 bits are used for number of entries in page table.

Page table entry = 1 valid bit + Frame number
Maximum bits needed for frame = 32 k physical frame
= 2^{15} = 15 bits frame
⇒ Page table entry = 1 valid bit + 15 bits of frame
= 16 bits = 2 bytes
∴ Page table size = Number of page table entries × Entry size
= 2^{22} × 2 bytes
= 8 MB

26. (d)

Round Robin with quantum 6

P_1	P_1		P_3	P_2	P_3	P_4	P_2	P_4	P_2

0 6 10 12 18 24 26 32 38 41 44

~~P_1~~ ~~P_1~~ ~~P_3~~ ~~P_2~~ ~~P_3~~ ~~P_4~~ ~~P_2~~ ~~P_4~~ ~~P_2~~

Processes	Arrival Time	Completion Time	Turn Around Time
1	0	10	10
2	18	44	26
3	12	26	14
4	20	41	21
			Average = $\frac{71}{4}$ = 17.75

SJF

P_1		P_3	P_4	P_2

0 10 12 20 29 44

Processes	Arrival Time	Completion Time	Turn Around Time
1	0	10	10
2	18	44	26
3	12	20	8
4	20	29	9
			Average $= \frac{53}{4} = 13.25$

SRTF

P_1		P_3	P_4	P_2

0 10 12 20 29 44

Processes	Arrival Time	Completion Time	Turn Around Time
1	0	10	10
2	18	44	26
3	12	20	8
4	20	29	9
			Average $= \frac{53}{4} = 13.25$

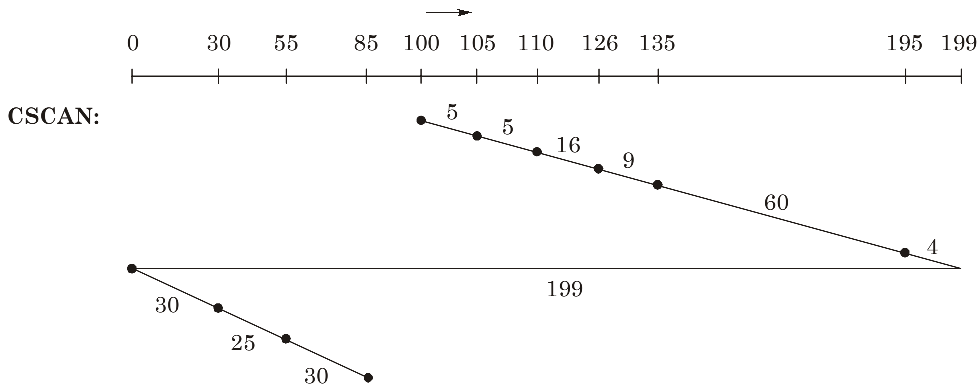
27. (d)

- Since 430 < 600, So , Physical address: 219 + 430 = 649
- Since 10 < 14 , So, Physical address: 2300 + 10 = 2310
- Since 500 >100, So, illegal reference and traps to operating system.
- Since 400 < 580 , So, Physical address: 1327 + 400 = 1727
- Since 112 > 96, So, illegal reference and traps to operating system

28. (d)

$$\text{Disk size} = 40 \text{ GB}$$
$$\text{Block size} = 1 \text{ KB}$$
$$\text{Number of blocks} = \frac{40 \times 2^{30}}{8 \times 2^{10}}$$
$$= 5 \text{ MB}$$

29. (b)



Total distance traversed by R/W head = (105 – 100) + (110 – 105) + (126 – 110) + (135 – 126) + (195 – 135) + (199 – 195) + (199 – 0) + (30 – 0) + (55 – 30) + (85 – 55)

= 5 + 5 + 16 + 9 + 60 + 4 + 199 + 30 + 25 + 30

= 383

30. (b)

Initially, there is compulsory miss from page number 1 to 30, at 31 there is miss page and page 1 is replaced and so on. Total 80 page fault.

On second access of 80 there is hit from 80 to 51. After 50 to 1 there are total 50 page faults.

Total page fault = 80 + 50 = 130

■■■■