CLASS TEST

Sl. : 02_SK_BCDE_CS_090920

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

Delhi | Bhopal | Hyderabad | Jaipur | Lucknow | Pune | Bhubaneswar | Kolkata | Patna

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

OPERATING SYSTEM

COMPUTER SCIENCE & IT

Date of Test : 09/09/2022

ANSWER KEY ➤

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

DETAILED EXPLANATIONS

1. (d)

(a) Round robin works on time quantum, after certain period of time every process gets the CPU unit for its completion, hence it's most suitable.

(b) Since OS is multiuser and multiprocessing, hence security is the primary concern so that user processes and Kernel processes can be isolated.
Hence two modes are required.

(c) When CPU temperature is too high, the BIOS initiate an interrupt. OS given top priority to this interrupt.

(d) Address translation table need to be changed when switching context from process A to process B.

2. (d)

- Switching between two user level threads only require procedure calls not context switching.
- All Kernal threads operations are implemented in Kernal, and OS schedules all threads in the system.
- Since user level threads are transport to Kernal, hence are not scheduled independently and hence are not given independent time slice.
- Threads do share the code segment.

3. (d)

Since 4 distinct page numbers are only to be accessed. Hence the best condition i.e., the condition with minimum number of page faults will be accessing all those elements repeatedly that are in the frame already, which will give maximum 4 page faults.
If, considered the worst case, it will be on every iteration, we are accessing the same element that has been removed from the frame, which will give 52 page faults.

4. (c)

1. CPU senses interrupt request line after every instruction.
2. Nearest cylinder next disk scheduling strategy gives the best throughput but the only problem is it can lead to starvation.
3. Using large file block size in a fixed block size file system leads to better disk throughput but poor disk space utilization.

5. (c)

Access time: total time needed to access the data

Access time = seek time + rotational latency + data transfer time

Seek time: Time taken to move the head to the correct cylinder that contains desired sector.

Rotational latency: Time taken to move the head to the desired sector within the cylinder.

Data transfer time : Time taken to transfer the actual data.

6. (a)

The value of Registers, Program counter and Stack pointers will be changed. Memory management information does not change.

7. (b)

- Contiguous file allocation results in external fragmentation because it could be the case that there are many small unallocated fragments between files and they can't be used to allocate a file whose size is greater than the size of any fragment but less than total size of free space.
- Linked allocation supports sequential access to disk block but not random access.
- Linked allocation does not suffer from external fragmentation but contain internal fragmentation.
- In indexed file allocation, separate block is used for each file as an index to store only block pointers.

8. (b)

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.
- Circular wait can be solved by numbering the resources in some order.

9. (a)

Linked allocation does not support direct access but indexed and contiguous allocation support direct access.

10. (d)

- Scheme 1 protocol ensures that hold and wait condition never occurs in the system.
- Scheme 2 ensures that there will be preemption of resources that have already been allocated.
- Scheme 3 ensures the circular wait condition.

11. (c)

1. Computer ( ) → p (mutex) → mutex = 0
   p (Q) → Q = 0
2. Science ( ) → p (Q) → process sleep
3. Computer ( ) → p (R) → R = 0
   v(Q) → Q = 1, science ( ) awake
4. Science → p(Q); Q = 0; p(R) → process sleep
5. Computer → v(mutex) → mutex = 1
   p(Q) → process sleep

Hence a deadlock.

12. (d)

	X	*Y*	*Z*	*W*
P_0	2	2	2	2
P_1	3	2	0	0
P_2	0	3	2	4
P_3	2	5	0	2
P_4	2	0	0	1

Since available is a 0 0 b, let's suppose a takes value 2 and b takes the value 1.

Available = 2 0 0 1

P_4 → Complete → Avail = (0000 + 6214) = 6214

P_1 → Complete → Avail = (6214) – (3200) = (3014) + (3512) = (6526)

P_0 → Complete → Avail = (6526) – (2222) = (4304) + (3242) = (7546)

P_2 → Complete → Avail = (7546) – (0324) = (7222) + (2775) = (9, 9, 9, 7)

P_3 → Complete → Avail = (9997) – (2502) = 7495

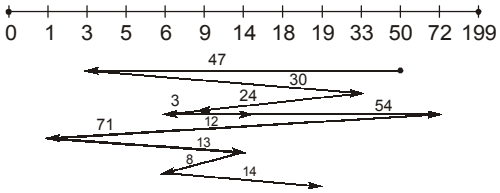
Hence, the system is in a safe state will value of a as 2 and value of b as 1.

13. (c)

Given function compare and swap is like test and set. Or, we can say that test and set is just a special case of compare and swap, which maintain mutual exclusion and is deadlock free.

14. (c)

- FCFS



Total time = 47+30+27+12+54+71+13+8+14 = 276 msec

15. (d)

- The total size of address space in a virtual memory system is limited by the available secondary storage.
- Best fit technique can also suffer from fragmentation.
- Locality of reference implies that the page reference being made by a process is likely to be the page used in the previous page reference.
- In a system with virtual memory context switch includes extra overhead in switching of address space.

16. (d)

The output is 'TGE'. So, to print 'T', we must give a value of 1 to semaphore b and should block rest three processes.

Now, process 3, after printing T, will give signal to semaphore a, which will wake up process 1 and will print 'G' and given signal to semaphore 'b' and 'c'. On giving signal to semaphore 'c', process '2' will get awake. But 'a' should not be printed in the output hence 'c' should be given value '–1'.

Process 4 will also awake after process 3 on signal 'a', but it will again be blocked by wait (b).

17. (b)

Given code correctly implements mutual exclusion but producer and consumer may enter the deadlock. Deadlock may occur; if consumer executes first down (mutex) and then down (full), consumer goes to block mode. Now producer executes down (empty) and then down (mutex), producer goes to block mode. Deadlock may occur due to the above executions when buffer is initially empty.

18. (a)

Mutual exclusion will always hold for these processes since only 1 process can enter the critical section at a time. Consider a scenario where all the processes execute first step i.e. turn [i] = true, so for all the process turn [i] will be true. Now, 'if' condition will be true for each process, so every process will sleep, which will result in deadlock condition.

19. (d)

Considering each statements :

S_1 : With Kernel threads, a thread can block on a semaphore and the Kernel can run some other thread in the same process. Consequently, there is no problem using semaphores. With user - level threads, when one thread blocks on a semaphore, the Kernel thinks the entire process is blocked and does not run it ever again. Consequently, the process fails.

S_2 : It does not lead to race conditions (nothing is even lost), but it is effectively busy waiting.

20. (c)

$$\begin{aligned}
\text{Page size} &= 8\text{ KB} \Rightarrow 13\text{ bit offset}\\
\text{Number of frame bits} &= 32 - 13 = 19\text{ bits}\\
\text{Page table entry} &= \text{Valid} + \text{Translation (frame bits)}\\
&= 1 + \text{Frame bits}\\
&= 1 + 19 = 20\text{ bits}\\
\text{Page table size} &= 20\text{ Mbytes}\\
\text{Number of pages} &= \text{Number of page table entries}\\
&= \frac{20\text{ Mbytes}}{20\text{ Bits}} = 8\text{ M} = 2^{23}\text{ pages}
\end{aligned}$$

$\therefore$ 23 bits needed for page and 13 bits offset
Length of virtual address = 23 + 13 = 36 bits.

21. (b)

	$(R_1$	R_2	$R_3)$
Total =	(12	9	12)
Allocated =	(10	8	11)
Available =	(2	1	1)

P_3 or P_4 can satisfy its need.

Available =	(2	1	1)
$P_3 \rightarrow$	(5	4	3)
	7	5	4

P_1 or P_2 or P_3 can satisfy.
It implies more than one safe sequence exist in the system.

Process	Need		
	R_1	R_2	R_3
P_1	2	2	1
P_2	4	1	0
P_3	1	0	0
P_4	2	0	0

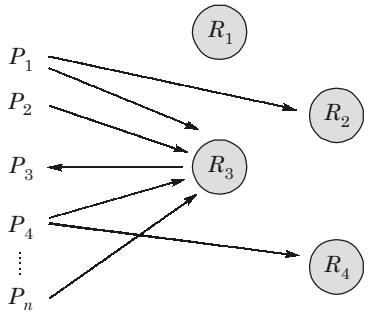
22. (a)

Only triple indirect blocks are responsible for maximum file size.

$$\begin{aligned}
\text{Max size} &= \left(\frac{\text{DB size}}{\text{DBA}}\right)^3 \times \text{DB size}\\
&= \left[\frac{4\text{ KB}}{4\text{ B}}\right]^3 \times 4\text{ KB} = \left[\frac{2^{12}\text{ B}}{4\text{B}}\right]^3 \times 4\text{ KB}\\
&= [2^{10}]^3 \times 4\text{ KB} = 2^{30} \times 2^{10} \times 4\text{ B}\\
&= 4 \times 2^{40}\text{ B}\\
&= 4\text{ TB}
\end{aligned}$$

23. (a)

- Given policy is one of the deadlock prevention policy so, deadlock not possible since cycle cannot be possible.
- Starvation is possible, consider the following case:



Since every process request for resource R_3, it might be the case process 1 get starve.

24. (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.

25. (c)

For variable '*a*' :

I. (i) $a = 2$
(ii) $a = 2 + 6 = 8$
Sequence = {1, 2, 3, 4} or {1, 3, 2, 4}

II. (iii) $a = 0 + 6 = 6$
(iv) $a = 2$
Sequence = {3, 4, 2, 1}

Hence possible value are {2, 8}.

For variable '*b*' :

I. (i) $b = 0 + 2 = 2$
(ii) $b = 4$
Sequence = {1, 2, 3, 4}

II. (i) $b = 4$
(ii) $b = 4 + 2 = 6$
Sequence = {3, 1, 2, 4} or {3, 4, 1, 2} or {1, 3, 2, 4}

III. (i) $b = 4$
(ii) $b = 4 + 8 = 12$
Sequence = {3, 1, 4, 2}

Hence possible value are {4, 6, 12}.

26. (d)

Locking of resource can create:

1. **Starvation:** P_1 wants resource R_1, which is locked by process P_2, when process P_2 completes another process P_3 locks resource R_1 before process P_1, it will happen for indefinite time, so process P_1 is in starvation.
2. **Inconsistent data:** This problem occur when one process fail in between and another dependent process read updated value of failed process.
3. **Deadlock:** When two process both want two resources to complete but currently lock one-one resources. This will create deadlock since both are waiting for resource to be free.

27. (c)

- Kernal level threads are designed as independent threads, so each thread can be scheduled separately.
- Kernal level threads have more context than user level threads, so switching between Kernal level threads is slower.
- Kernal level threads are designed as independent, so blocking one threads does not stop entire process, which is not the case with user level threads.
- Kernal level threads are independent so can be run simultaneously on different processors.

28. (b)

- To get the minimum value of count.

Tally1( )	Tally2( )	Tally3( )
Iteration 1. I: count = 0, $R_1 = 0$		
II: count = 0, $R_1 = 0$		
	Iteration 1. count = 1, $R_2 = 1$	
	2. count = 2, $R_2 = 2$	
	3. count = 3, $R_2 = 3$	
	4. count = 4, $R_2 = 4$	
	5. count = 5, $R_2 = 5$	
		Iteration 1. count = 6, $R_3 = 6$
		2. count = 7, $R_3 = 7$
		3. count = 8, $R_3 = 8$
		4. count = 9, $R_3 = 9$
Iteration 1. III: count = 1, $R_1 = 1$		
		Iteration 5. I: count = 1, $R_3 = 1$
		II. count = 1, $R_3 = 2$
Iteration 2. count = 2, $R_1 = 2$		
3. count = 3, $R_1 = 3$		
4. count = 4, $R_1 = 4$		
5. count = 5, $R_1 = 5$		
		Iteration 5. III: count = 2, $R_3 = 2$

Time

- To get the maximum value, execute P_1, P_2 and P_3 completely in sequence.

$\therefore$ Count = 15.

29. (a)

$$\text{Waiting Time} = \text{Turn Around Time} - \text{Burst Time}$$

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

Process	Waiting Time
P_0	0
P_1	0
P_2	3
P_3	0
P_4	2

$$\text{Average Waiting Time} = \frac{0+0+3+0+2}{5} = 1\,\text{ms}$$

30. (d)

- Pages that are shared between two or more processes can be swapped out to disk when demand paging is applied and we have to swap in new pages and main memory is full.
- The operating system automatically loads pages from disk when necessary when it is needed
- The translation look aside buffer is a hardware data structure.

■■■■