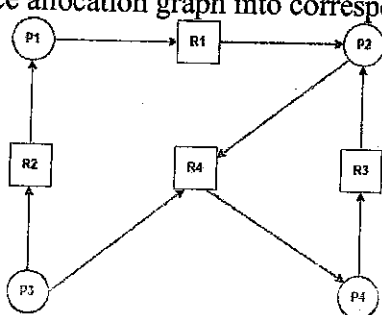


Note: (a) All questions are compulsory.

Q.N o	Questions	C O	M ar ks																																			
1	<p>An operating system uses the Banker's algorithm for deadlock avoidance when managing the allocation of three resource types A, B, and C to three threads T_1, T_2 and T_3. The table given below presents the current system state. Here, the Max matrix shows the maximum number of resources of each type required by each thread during its execution and the Allocation matrix shows the current number of resources of each type allocated to each thread. There are 3 units of type A, 2 units of type B and 2 units of type C still available.</p> <table><tr><th>Thread</th><th colspan="3">Max</th><th colspan="3">Allocation</th></tr><tr><th></th><th>A</th><th>B</th><th>C</th><th>A</th><th>B</th><th>C</th></tr><tr><td>T_1</td><td>8</td><td>4</td><td>3</td><td>0</td><td>0</td><td>1</td></tr><tr><td>T_2</td><td>6</td><td>2</td><td>0</td><td>3</td><td>2</td><td>0</td></tr><tr><td>T_3</td><td>3</td><td>3</td><td>3</td><td>2</td><td>1</td><td>1</td></tr></table> <p>(a) Find out the total number of instances of each resource type? (b) Is the system currently in a safe state? If yes, find out the safe sequence. If a request from thread T_1 arrives for (0, 0, 2), can the request be granted immediately?</p>	Thread	Max			Allocation				A	B	C	A	B	C	T_1	8	4	3	0	0	1	T_2	6	2	0	3	2	0	T_3	3	3	3	2	1	1	3	(1+ 2+ 2)
Thread	Max			Allocation																																		
	A	B	C	A	B	C																																
T_1	8	4	3	0	0	1																																
T_2	6	2	0	3	2	0																																
T_3	3	3	3	2	1	1																																
2	<p>Discuss the importance of wait-for graph in deadlock handling mechanism. Convert the given resource allocation graph into corresponding wait-for graph.</p>  <p>Fig 1. Resource Allocation Graph</p>	3	(1+ 1)																																			
3	<p>Two processes X and Y need to access a critical section. Consider the following synchronization construct used by both the processes.</p>	2	(3)																																			

Process X	Process Y
<pre>/* other code for process X */ while (true) { varP = true; while (varQ == true) { /* Critical Section */ varP = false; } } /* other code for process X */</pre>	<pre>/* other code for process Y */ while (true) { varQ = true; while (varP == true) { /* Critical Section */ varQ = false; } } /* other code for process Y */</pre>

Here *varP* and *varQ* are shared variables and both are initialized to false. Which one of the following statements is true?

- A. The proposed solution prevents deadlock but fails to guarantee mutual exclusion
- B. The proposed solution guarantees mutual exclusion but fails to prevent deadlock
- C. The proposed solution guarantees mutual exclusion and prevents deadlock
- D. The proposed solution fails to prevent deadlock and fails to guarantee mutual exclusion

- 4 Consider the methods used by processes P1 and P2 for accessing their critical sections whenever needed, as given below. The initial values of shared Boolean variables S1 and S2 are randomly assigned. Find the following and provide appropriate explanation.

- 1. Mutual Exclusion
- 2. Progress
- 3. Bounded wait

Method used by P1	Method used by P2
while (S1 == S2);	while (S1 != S2);
Critical Section	Critical Section
S1 = S2;	S2 = not(S1);

- 5 Consider the following solution to the producer-consumer synchronization problem. The shared buffer size is N. Three semaphores *empty*, *full* and *mutex* are defined with respective initial values of 0, N and 1. Semaphore *empty* denotes the number of available slots in the buffer, for the consumer to read from. Semaphore *full* denotes the number of available slots

in the buffer, for the producer to write to. The placeholder variables, denoted by P, Q, R, and S, in the code below can be assigned either *empty* or *full*. The valid semaphore operations are: *wait ()* and *signal ()*.

Producer:	Consumer:
<pre>do{ wait(P); wait(mutex); //Add item to buffer signal(mutex); signal(Q); }while(1);</pre>	<pre>do{ wait(R); wait(mutex); //Consume item from buffer signal(mutex); signal(S); }while(1);</pre>

Which one of the following assignments to P, Q, R and S will yield the correct solution?

- a** P: empty, Q: empty, R: full, S: full
- b** P: full, Q: empty, R: empty, S: full.
- c** P: empty, Q: full, R: full, S: empty

6 Synchronization in the classical readers and writers problem can be achieved through use of semaphores. In the following incomplete code for readers-writers problem, two binary semaphores *mutex* and *wrt* are used to obtain synchronization

```
wait (wrt)
writing is performed
signal (wrt)
wait (mutex)
readcount = readcount + 1
if readcount = 1 then S1
S2
reading is performed
S3
readcount = readcount - 1
if readcount = 0 then S4
signal (mutex)
```

The values of S1, S2, S3, S4, (in that order) are

- a** signal (mutex), wait (wrt), signal (wrt), wait (mutex)
- b** signal (wrt), signal (mutex), wait (mutex), wait (wrt)

2 (3)

	C wait (wrt), signal (mutex), wait (mutex), signal (wrt)		
7	A shared variable x , initialized to zero, is operated on by four concurrent processes A, B, C, D as follows. Each of the processes A and B reads x from memory, increments by one, stores it to memory and then terminates. Each of the processes C and D reads x from memory, decrements by two, stores it to memory, and then terminates. Each process before reading x invokes the P operation (i.e. wait) on a counting Semaphore S and invokes the V operation (i.e. signal) on the semaphore S after storing x to memory. Semaphore S is initialized to two. What is the maximum possible value of x after all processes complete execution?	3	(4)
8	A counting semaphore S is initialized to 20. Then, 12 P operations and 8 V operations are performed on S. Find the final value of S?	3	(2)

ALL THE BEST