Name- Ashutosh soni
Id- 2018UCP1505

Que 1.

(a) Parallel & concurrent Programming

Parallel Prog.	Concurrent Prog.
1) In parallel programming single task divided into multiple single independent tasks which can be performed simultaneously	1). In concurrent Programming multiple tasks performed simultaneously with shared resourcess.
2) Programming as the simultaneous execution of (possibly related) computations	2) Programming as the composition of independently executing process
3) In this type of programming tasks &. literally run at the same time.	3). In this type of programming two tasks can start and run and complete in overlapping time periods.

Scanned by CamScanner

Name-Ashutosh soni
Id-2018UCP1505.

(b). Strong & weak Semaphore.

Strong semaphore	Weak semaphore
1) A semaphore whose definition includes the policy of first in first out (FIFO) queue.	1) A semaphore that doesnot specify the order in which the processes are removed from the queue.
2) As process have sequence so chances of starvation is less or we say because of this no starvation happens.	2). As process have no sequence means came out of queue is arbitrary order so chances of starvation arise.

(c) Deadlock & Livelock.

Deadlock	Livelock.
→ In this case, nothing whatsoever is being computed. here or this term used for frozen computation.	→ This is a scenario where several processes are actively executing statements, but nothing useful gets done is called livelock.
• Here the states are frozen.	→ Here the states changes but nothing useful will happen.
→ It is like "Me first-Me first". but noone get entry.	→ It is like "You first, You first" but noone get enter.

Name - Ashutosh Soni
Id - 2018UCP1505

(d). Data & Task Parallelism.

Data Parallelism	Task Parallelism.
1) Same tasks are performed on different subsets of same data.	1) Different tasks are performed on the same or different data.
2). Synchronous computation is performed	2). Asynchronous computation is performed.
3). Amount of parallelism is proportional to the input size.	3). Amount of Parallelism is proportional to the number of independent tasks is performed.

(e). Message Passing & Shared Memory Paradigm.

Message Passing	Shared Memory.
1) It is typically used in a distributed environment where communicating process reside on remote machines connected through a network.	1) It is used for communication between processes on a single processor or multiprocessor systems where the communicating processes reside on the same machine as the communicating processes share a common address space.
2) It is useful for sharing small amounts of data as conflicts need not to be resolved.	2) Here the processes need to ensure that they are not writing to the same location simultaneously
3) Relatively slower communication strategy	3) Faster communication strategy.

Name-Ashutosh Soni
Id-2018UCP1505

Ans(2).

(a).

- **Algorithm for stimulation of Monitors using semaphore :**

monitor sem.
    integer. S ← K.
    condition notZero.
    operation. wait
        if S=0.
            waitC(notzero)
        S=S-1.

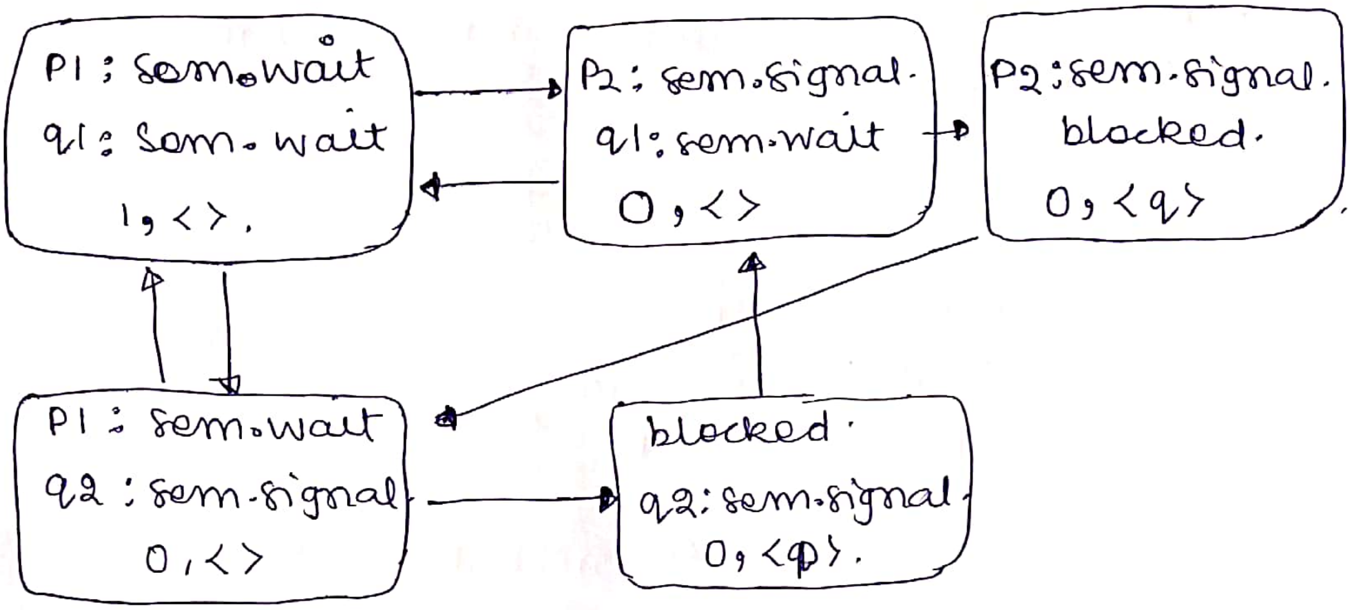
    operation signal.
        S=S+1.
        signal C(not zero).

p	q.
loop forever non-critical section. p1: sem.wait critical section sem.signal.	loop forever non critical section q1: semwait critical section. sem.signal.

Name- Ashutosh soni

Id- 2018UCP1505

- State diagram for the semaphore simulation.



Now, wait & signal gave the same functionalality of waiting & process to solve the problem of critical section.

If the wait signal is not there then the process got blocked & enter into the queue & on getting the signal process got unblocked & then the same happens for both the process, & the process will carry on.

Name-Ashutosh Soni
Id-2018UCP1505

wait C (cond).

append p to cond.

p.state ← blocked.

monitor.lock ← release

signal C (cond).

if cond ≠ empty.

remove head of cond & assign to q

q.state ← ready.

Name-Ashutosh Soni

Id-2018UCP1505.

Ans 2.

(b).

Various methods to handle critical section problem.

1) using global or local variable.
2) using semaphore.
3) using monitors.

1) using global or local variable.

- they are simple just initialize the variable and handle the problem.

2) using semaphore.

→ first initialize the semaphore.

sem_t a;.

sem_init (&a, 0, 1);.

then by using sem_wait (&a) &

sem_post (&a) as ~~used~~ for.

waiting & signal accordingly they are used in order to handle critical section.

Scanned by CamScanner

Name - Ashutosh Soni
Id - 2018UCP1505

binary semaphore. S ← (1, φ).

p	q
loop forever	loop forever
p1: non critical section	q1: non critical section
p2: wait(S).	q2: wait(S)
p3: critical section	q3: critical section
p4: signal(S).	q4: signal(S).

3) using Monitors.

p	q
loop forever	loop forever
p1: non critical section	q1: non critical section
p2: ~~wait~~ Mona.wait	q2: mona.wait
p3: critical section	q3: critical section
p4: ~~so~~ Mona.signal.	q4: Mona.signal.

above monitor is Mona.

this is how critical section problem is handled using monitor where wait & signal are operations of monitors.

Name-Ashutosh soni
Id-2018UCP1505

Que 3
(a) ans =)
The given algorithm is the forth attempt of Dekker's algorithm. inorder to solver the problem of critical section..
The above algorithm is.
- free from deadlock
- & holds mutual exclusion principle

but
starvation may occur in the above algorithm.

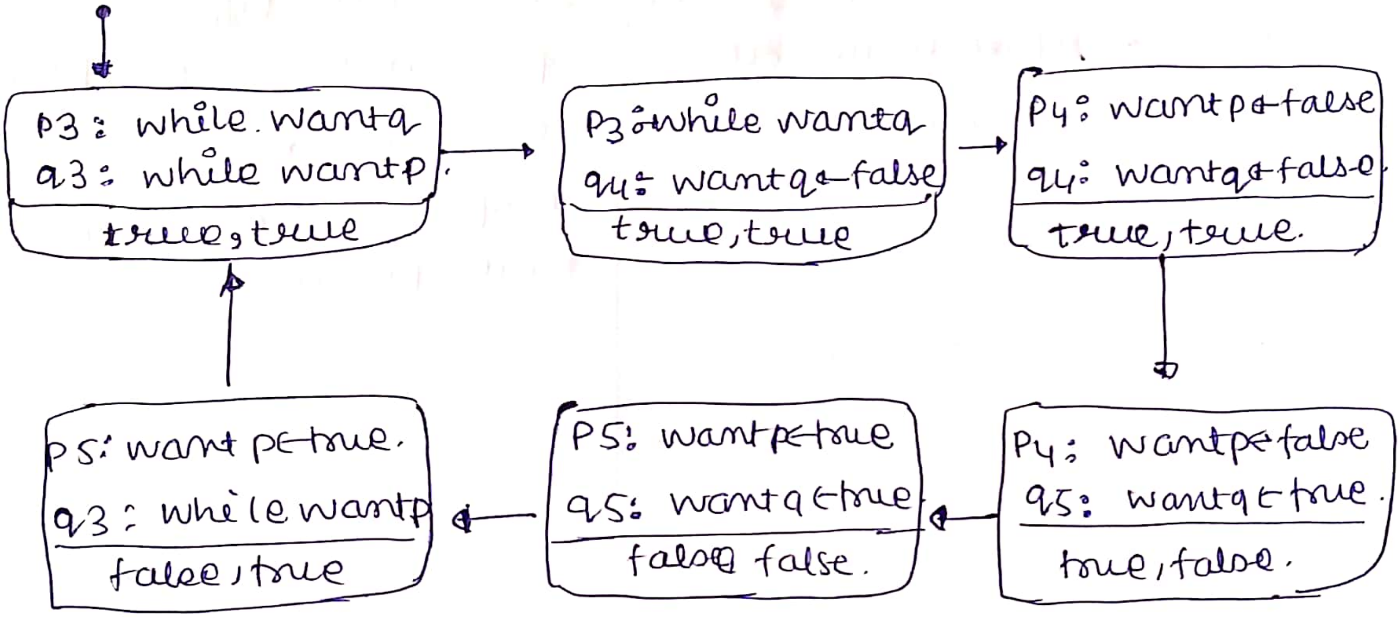
Prove for all the correctness specifications:

1). Holds Mutual Exclusion Principle:
As Arbitrary. interleaving takes place in above algorithm. & waiting for. want of other section to be false in order to enter into the. critical section & finally only one thread may get into the critical section.
only one process present in critical section shows that the above Algorithm is ~~for~~ holding Mutual exclusion principle;

2) Free from deadlock:-
As in above algorithm if sections are. waiting for enter into the critical ~~section~~ section then. one of the section or process eventually get. succed. in the attempt to enter into the critical section so the Algorithm is free from deadlock.

Scanned by CamScanner

3) <u>state diagram for above algorithm:-</u>



This above state diagram shows that the algorithm might get starved. & above case is the case of starvation. So the above algorithm is not free from starvation or say may get starved.

Name-Ashutosh Soni

Id-2018UCP1505

(b)

Algorithm:

```
Semaphore. Room = 4.
Semaphore fork (5) := ((5) 1).

process. philospher (i := 0 to 4) {.
    while (1) {

        Think();            //thinking not a CS.
        wait (Room);
        wait (fork (i));.
        wait ( fork (i+1) mod 5 );

        Eat();              // Eating is the CS.

        signal ( fork(i));
        signal ( fork(i+1) mod 5)};.
        signal (Room);.
    }
}.
```

→ As we limit the no. of ~~ph~~ rooms to be 4 so. finally or eventually one of the philospher get into the critical section.

Above Algorithm satisfies all the liveness property these are Mutual exclusion, free from deadlock & free from starvation.