Dr. P.K Gupta

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

TEST -2 EXAMINATION - OCTOBER 2019

B.Tech 5th Semester

COURSE CODE: 10B11CI511 **MAX. MARKS: 25**

COURSE NAME: Operating Systems

COURSE CREDITS: 04 **MAX. TIME: 1Hr 30 Min**

Note: *All questions are compulsory. Carrying of mobile phone during examinations will be treated as case of unfair means.*

SECTION A (4x1=4)

Q1.

a. Are binary semaphore and mutex same? Explain

b. What private (thread-private) and public (shared between cooperating threads) resources are allocated when a thread is created? How do they differ from those allocated when a process is created?

c. "*An application programmer has to worry about deadlock occurring involving the CPU as a resource*" Justify the validity of this statement.

d. How interrupts can be used to implement critical sections?

SECTION B (3x3=9)

Q2.

a. Given six memory partitions of 300 KB, 600 KB, 350 KB, 200 KB, 750 KB, and 125 KB (in order), how would the first-fit, best-fit, and worst-fit algorithms place processes of size 115 KB, 500 KB, 358 KB, 200 KB, and 375 KB (in order)? Rank the algorithms in terms of how efficiently they use memory.

b. A particular abstraction only allows a maximum of 10 threads to enter the "room" at any point in time. Further threads attempting to enter the room have to wait at the door for another thread to exit the room. How could one implement a synchronization approach to enforce the above restriction?

c. Let m[0]…m[4] be mutexes (binary semaphores) and P[0] …. P[4] be processes. Suppose each process P[i] executes the following:

```
wait (m[i]); wait(m[(i+1)mod 4]);
------
release (m[i]); release (m[(i+1)mod 4]);
```

Is the above code will turn Deadlock? Justify your answer with suitable reasons.

SECTION C (3x4=12)

Q3. Implement a counting semaphore x using binary semaphores and no other synchronization construct (no atomic read modify-write, no disabling interrupts, no access to PCBs, no wait/wakeup, etc.) and no busy waiting. Specifically, supply three chunks of code: one for x's initialization, one for P(x), one for V(x).

Q4. The following function is called by multiple threads (potentially concurrently) in a multi-threaded program. Identify the critical section(s) that require(s) mutual exclusion. Describe the race condition or why no race condition exists. How would you synchronize the following code?

```
int i;
void synchro()
{
    int j;
    /* random stuff*/
    i = i + 1;
    j = j + 1;
    /* more random stuff */
}
```

Q5. Consider the following snapshot of a system

Process	Allocation	Max	Available
	A B C D	A B C D	A B C D
P0	0 0 1 2	0 0 1 2	1 5 2 0
P1	1 0 0 0	1 7 5 0	
P2	1 3 5 4	2 3 5 6	
P3	0 6 3 2	0 6 5 2	
P4	0 0 1 4	0 6 5 6	

Answer the following questions using the banker's algorithm:

(a) Is the system in a safe state? Write the safe sequence with proper explanation. **[2]**

(b) If a request from process P1arrives for (0, 4,2,0), can the request be granted immediately? Show your processing steps. **[2]**