BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE, PILANI
Second Semester 2003-2004

Course Title : OPERATING SYSTEMS **Course No *CS C372 & IS C362***
Component : Test II (Regular) ***Open Book Component***
Weightage : 12% **Max Marks: 12** **Date : 09-04-2004**

Note: Attempt all the Questions. Start each answer from a fresh page.

Question #1 **(2 Marks)**

There are a lot of similarities between semaphores and condition variables

a. What is the major difference between semaphores and condition variables?

Semaphores have a value; condition variables do not.
Semaphores provide mutual exclusion and synchronization; condition variable can be used only for synchronization.
Semaphores can't wait inside a critical section; condition variables can (inside a lock).

b. Describe a situation where a semaphore P and V behaves differently than a condition variable wait and signal.

If s is a semaphore, and one process does V(s), and then later another process does P(s), that second process does not wait at the P.
If c is a condition variable, and one process does signal(c), and then later another process does wait(c), that second process waits until yet another signal(c) is done.

Question #2 **(2 Marks)**

Consider 2 processes P0 and P1 running concurrently with race condition. Calculate the maximum and minimum value of x.

P0	P1
for(int i=0; i<1000; i++) {	for(int j=0; j<2000; j++) {
x++;	x-- ;
x-- ;	x++;
x++;	}
}	

Max x value is 4000
Min x value is -2999

Question #3 **(3 Marks)**

A file is to be shared among different processes, each of which has a unique number. The file can be accessed simultaneously by several processes, subject to the following constraint: The sum of all unique numbers associated with all the processes currently accessing the file must be less than n. Write a program to coordinate the access to the file using Semaphore.

```
wait_no = 0;
queue=0,mutex = 1;

wait (mutex);
if( sum + num(i) < n)
{
        sum = sum + num(i);
        signal(mutex);
}
else
{
        wait_no++;
        signal(mutex);
        wait(queue);
}

//Access the File

wait(mutex);
if(wait_no>0)
        signal(queue);
sum = sum – num(i);
signal(mutex);
```

Question #4 **(1 Marks)**

A computer has 16 tape drives, with n processes competing for them. Each process may need 5 tape drives and are holding two currently. What can be the minimum value of n that makes the system deadlock?

Answer **7**

Question #5 **(2 Marks)**

Consider the following snap shot of a system. There are no current outstanding queued unsatisfied requests.

Available			
R1	R2	R3	R4
2	1	0	0

	Current Allocation					**Maximum Demand**			
Process	**R1**	**R2**	**R3**	**R4**		**R1**	**R2**	**R3**	**R4**
P1	0	0	1	2		0	0	1	2
P2	2	0	0	0		2	7	5	0
P3	0	0	3	4		6	6	5	6
P4	2	3	5	4		4	3	5	6
P5	0	3	3	2		0	6	5	2

a. Is this system currently in safe or unsafe? Why?
b. If safe write down the safe sequence. If unsafe mention the processes in unsafe state.
c. If a request from P3 arrives for (0,1,0,0), can that request be safely granted immediately? In what state (deadlock, safe, unsafe) would immediately granting that whole request leave the system? Which processes, if any, are or may become deadlocked if this whole request is granted immediately?

Need matrix

0 0 0 0
0 7 5 0
6 6 2 2
2 0 0 2
0 3 2 0

a. System is safe

b. Running the banker's algorithm, we see processes can finish in the order p1, p4, p5, p2, p3.

c. Change available to (2,0,0,0) and p3's row of "still needs" to (6,5,2,2). Now p1, p4, p5 can finish, but with available now (4,6,9,8) neither p2 nor p3's "still needs" can be satisfied. So it is not safe to grant p3's request.

Question #6 **(2 Marks)**

A 1-Mbyte block of memory is allocated using the buddy system.

a. Show the result of the following sequence in a figure. Request A (65K), request B(210K), request C(39K), request D(120K), request E(64K), request F(196K).

b. Show the binary tree representation after F and calculate total internal fragmentation due to this allocation.

Answer

a.

A	**C**	**E**	**B**	**D**		**F**

b. Total internal fragmentation

Request A (128K – 65K) = 63K
Request B (256K – 210K) = 46K
Request C (64K – 39K) = 25K
Request D (128K – 120K) = 8K
Request E (64K – 64K) = 0K
Request F (256K – 196K) = 60K

Total internal fragmentation (63 + 46 + 25 + 8 + 0 + 60) = 202K

Binary Tree Reprasentation