CSI1002 Operating System Principles

Set-2

Name : Prashanth.S

Roll : 19MIDO020

3) Lamport's bakery algorithm:

1)

a) At time T_0. Is the system safe?

* P_5 requests NO resources. So P_5 uses 1 instance of R_4 and executes.

* So P_3 requests R_3 and holds R_3 and executes successfully

* P_2 requests R_3 and holds R_2 and executes successfully.

* P_1 requests R_2 and holds R_1 and executes successfully.

* At last P_4 requests 3 instance of R_1 and holds 1 instance of R_2 and R_4.

* So all the process doesn't enter into deadlock.

Since there is no cycle, NO deadlock.

B) T_1 : P_5 releases R_4

T_2 : Request for R_4 by P_2 is granted

Both P_2 and P_3 requests R_4 and P_4 holds R_4

Cannot able to proceed further.

So the condition goes into deadlock.

C) At time t_0 ⇒ No deadlock P_3 contributes to deadlock.

At time t_1 ⇒ P_2 and P_3 contributes to deadlock.

D) At time t_D When P_4 is terminated, it will recover from deadlock

$T_0 \Rightarrow P_4$ (critical)

2	1	3	10
0	1	2	3

3)

P_0	P_1	P_2
T_3 i.1	T_1 i.1	T_8 i.1
T_4 i.2	T_2 i.2 (loss)	T_9 i.2
T_5 i.3 (loss	T_6 i.3	T_{10} i.3 loss
T_{11} : processor	T_7 i.4 (loss)	

a) Num for $P_0 \Rightarrow 2$

$P_1 \Rightarrow 1$

$P_2 \Rightarrow 3$

b) P_1 will the critical section first, since it has the lowest token value.

9) P_0 will be second since it has the least token after the execution of P_1

10) Yes Mutual exclusion is achieved.

19MID0020

2)

count ⇒ 10

T_1	10
S_1	10
T_2	50
S_2	5
S_3	5
T_3	50
U_1	50
S_1	50
U_2	55
T_1	50
T_2	250
S_2	45
U_3	55

count	req-1	req-2	req-3
~~10~~ 5	10	10	~~50~~ 55
~~50~~ 55	~~5~~ 50	~~50~~ 250	

```
boolean TestAndSet ( boolean * target)

{ boolean rv = *target;
  *target = TRUE;
  return rv;
}

boolean waiting [n];
boolean lock;
lock = False;
waiting [i] = False;

do
{
    waiting [i] = TRUE;
    key = TRUE;
    while ( waiting [i] && key)
        key = TestAndSet (& lock) ;
    waiting [i] = FALSE;

  { Critical section ; }
```

```
while ((j != i) && (!waiting[j]))
        j = (j+1) % n;
if (j == i)
        lock = False;
else
        waiting[j] = FALSE;
} while (true)
```