**Operating Systems and Systems Programming**

**Tutorial – 6: Deadlock**

**Solution1.**

Yes, both processes will enter deadlock at second statement.

**Solution2.**

In deadlock prevention method, either process P will acquire both R1 or R2 resources or process Q will acquire both resources at a time. None of the processes shall acquire one resource and wait for another resource which is not available.

**Solution3.**

Safe sequence: A, C, B

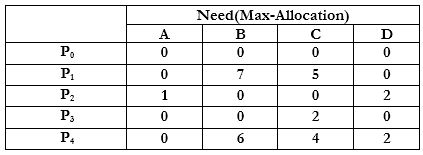
If C should have a claim of 9 instead of 7, there is no safe sequence.

**Solution4.**

|  |  |
| --- | --- |
| **P0** | **P1** |
| **Wait(A)** | **Wait(B)** |
| **Wait(B)** | **Wait(A)** |
| **Signal(B)** | **Signal(A)** |
| **Signal(A)** | **Signal(B)** |

**Solution5.**

Need matrix is calculated by subtracting Allocation Matrix from the Max matrix



To check if system is in a safe state

* The Available matrix is [1520][1520].
* A process after it has finished execution is supposed to free up all the resources it hold.
* We need to find a safety sequence such that it satisfies the criteria: need Need≤ Available

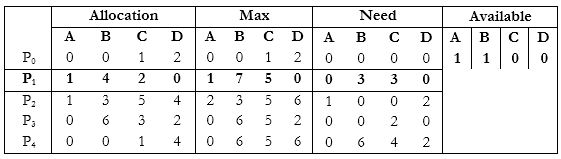
Since Need(P0)≤ Available, we select P0.[Available]=[Available]+[Allocation(P0)] Available=[1520]+[0012]=[1532] Need(P2) ≤ Available →→Available=[1 5 3 2]+[1 3 5 4]=[2 8 8 6]

* Need(P3) ≤ Available→→ Available=[ 2 8 8 6 ]+[ 0 6 3 2 ]=[2 14 11 8 ]
* Need(P4) ≤ Available→→Available=[ 2 14 11 8 ]+[0 0 1 4 ]=[ 2 14 12 12 ]
* Need(P1) ≤ Available→→Available=[ 2 14 12 12 ]+[ 1 0 0 0 ]=[ 3 14 12 12]
* Safe Sequence is <p0,p2,p3,p4,p1>

**A request from process P1 arrives for (0,4,2,0)**

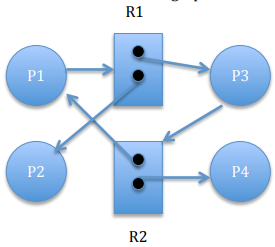
* System receives a request for P1 for Req(P1)[0420]

First we check if Req(P1) is less than Need(P1)Need(P1)→[0420]<[0750]istrue

* Now we check if Req(P1) is less than Available→[0420]<[1520]istrue.
* So we update the values as:
  + Available=Available−Request=[1520]−[0420]=[1100] Allocation=allocation(P1)+Request=[1000]+[0420]=[1420] Need=Need(P1)−Request=[0750]−[0420]=[0330]
  + 
* This is the modified table
* On verifying, we see that the safe sequence still remains the same .The system continues to remain in a safe state.

**Solution6.**

**a)**

****

**b)** P1 R1 P3 R2 P1

c) No. There is a cycle, but no deadlock. P2 and P4 have all resources for completing. P2 P4, P1 P3

**Solution7**.

 a. The four necessary conditions for a deadlock are

(1) mutual exclusion;

(2) hold-and-wait;

(3) no preemption; and

(4) circular wait.

The mutual exclusion condition holds since only one car can occupy a space in the roadway. Hold-and-wait occurs where a car holds onto its place in the roadway while it waits to advance in the roadway. A car cannot be removed (i.e. preempted) from its position in the roadway. Lastly, there is indeed a circular wait as each car is waiting for a subsequent car to advance. The circular wait condition is also easily observed from the graphic.

b. A simple rule that would avoid this traffic deadlock is that a car may not advance

**Solution8.** Given tape drive = 6 and each process may need 2 drive.

When we give 1 drive to 1 process then total process will be 6 but in this case there will definitely deadlock occur because every process contain 1 drive and waiting for another drive which is hold by other process therefore when we reduce 1 process then system to be deadlock free.

Hence maximum value of n = 6 – 1 = 5.

**Solution9.** Consider the worst case- all processes acquire maximum resources but still not able to finish. So, the resources available must be 1 less than the maximum need, for each of the processes (this ensures none of them can finish).  
We are given maximum need is always less than m + n. As per our condition for deadlock, resources available must be 1 less than maximum need for each of 'm' processes => totally the resources available must be less than m + n - m = n.  
But 'n' is the available number of resources and hence no deadlock can occur.

**Solution10.** Given, Number of processes (P) = 3 & Number of resources (R) = 4

Since deadlock-free condition is:

R ≥ P (N − 1) + 1

Where R is total number of resources, P is the number of processes, and N is the max need for each resource.

4 ≥ 3(N − 1) + 1

3 ≥ 3(N − 1)

1 ≥ (N − 1)

N ≤ 2

Therefore, the largest value of K that will always avoid deadlock is **2**.