1. Consider the following snapshot of a system:

	Allocation	***Max***	***Available***
	A B C D	*A B C D*	*A B C D*
*P*0	2 0 0 1	4 2 1 2	3 3 2 1
*P*1	3 1 2 1	5 2 5 2	
*P*2	2 1 0 3	2 3 1 6	
*P*3	1 3 1 2	1 4 2 4	
*P*4	1 4 3 2	3 6 6 5	

Answer the following questions using the banker's algorithm:

a. Illustrate that the system is in a safe state by demonstrating an order in which the processes may complete.
b. If a request from process *P*1 arrives for (1, 1, 0, 0), can the request be granted immediately?
c. If a request from process *P*4 arrives for (0, 0, 2, 0), can the request be granted immediately?

Answer:

The values of Need for processes P0 through P4 respectively are (2, 2, 1, 1), (2, 1, 3, 1), (0, 2, 1, 3), (0, 1, 1, 2), and (2, 2, 3, 3).

a. With **Available** being equal to (3, 3, 2, 1) only process P0 could run. Once process P0 runs, it releases its resources (2, 0, 0, 1) which are added to the **Available.**

   With **Available** being equal to (5, 3, 2, 2), only process P3 could run. Once process P3 runs, it releases its resources (1, 3, 1, 2) which are added to the **Available.**

   Available is (6, 6, 3, 4) which allows all other existing processes to run.

   The system is in a safe state.

   Any order that starts with <P0, P3, …> can complete. One ordering of processes that can finish is <P0, P3, P1, P2, P4>

b. The request can be granted immediately?

   This reduces in the value of **Available** to (2, 2, 2, 1). With the addition of (1, 1, 0, 0) P1's **Allocation** is (4,2, 2, 1) and **Need** reduced to (1, 0, 3, 1)

   Available is (2, 2, 2, 1). Values of **Need** for processes P0 through P4 respectively are (2, 2, 1, 1), (1, 0, 3, 1), (0, 2, 1, 3), (0, 1, 1, 2), and (2, 2, 3, 3). Follow the

instructions in part (a) to complete. Answer is yes, if you get a safe sequence as shown above.

c. Similar to part b. Follow the same steps.

2. Assume there are three resources, R1, R2, and R3 that are each assigned unique integer values 15, 10, and 25, respectively. What is a resource ordering which prevents a circular wait?

   Follow the integer order. Answer is R2, R1, and R3

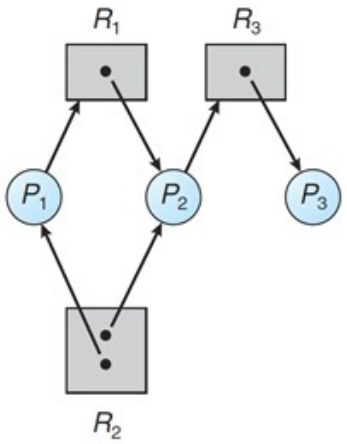
3. A resource-allocation graph is shown in Figure (a).
   a. Is the system deadlocked? Why or why not?

      No. When P3 finishes R3 will be available for P2. All processes can finish in the order P3, P2, & P1

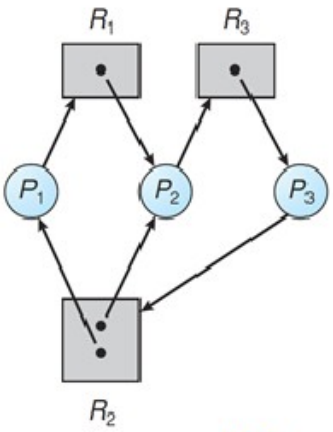
   Suppose that process P3 requests an instance of resource type R2. We add a request edge P3→ R2 to the graph as shown Figure (b).

   b. Is the system deadlocked? Why or why not?

      Yes. P1 is waiting for R1 held by P2, which is waiting for R3 held by P3, which in turn is waiting for R2 form either P1 or P2



(a)



(b)

4. Answers to all the rest of questions are directly in the slides given.