CS 162 Section 3 - Solutions

Short Answer

1. List the four requirements for deadlock and explain each.

   Mutual exclusion - only one thread at a time can use a resource
   Hold and wait - Thread holding at least one resource is waiting to acquire additional resources held by other threads
   No preemption - resources are released only voluntarily by the thread holding the resource, after thread is finished with it
   Circular wait - there exists a set $\{T_1, ..., T_n\}$ of waiting threads
   - » T_1 is waiting for a resource that is held by T_2
   - » ...
   - » T_n is waiting for a resource that is held by T_1

2. Explain two approaches to avoiding deadlock.

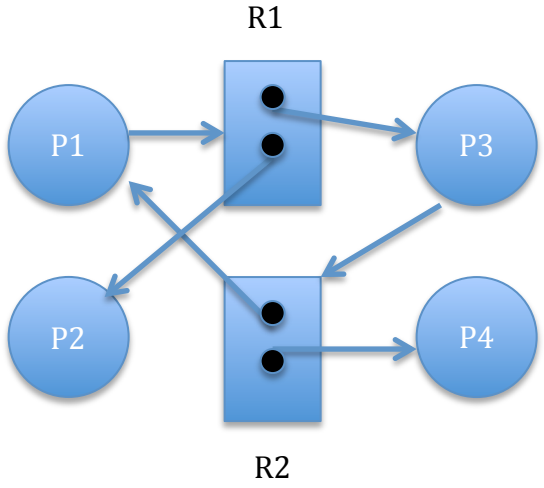
   - Ordering on resource acquisition (example: dining philosophers)
   - Using banker's algorithm

Resource Allocation Graphs

Consider a system with four processes P1, P2, P3, and P4, and two resources, R1, and R2, respectively.
Each resource has two instances. Furthermore:

- P1 allocates an instance of R2, and requests an instance of R1
- P2 allocates an instance of R1, and doesn't need any other resource
- P3 allocates an instance of R1 and requires an instance of R2
- P4 allocates an instance of R2, and doesn't need any other resource

Draw the resource allocation graph.



Is there a cycle in the graph? If yes name it.

P1 R1 P3 R2 P1

Is the system in deadlock? If yes, explain why. If not, give a possible sequence of executions after which every process completes.

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

Resource Allocation Tables

Consider the following snapshot of a system with five processes (P1, P2, P3, P4, P5) and four resources (R1, R2, R3, R4). There are no current outstanding queued unsatisfied requests.

Currently Available Resources

R1	R2	R3	R4
2	1	2	0

	Current Allocation				Max Need				Still Needs			
Process	**R1**	**R2**	**R3**	**R4**	**R1**	**R2**	**R3**	**R4**	**R1**	**R2**	**R3**	**R4**
P1	0	0	1	2	0	0	3	2	0	0	2	0
P2	2	0	0	0	2	7	5	0	0	7	5	0
P3	0	0	3	4	6	6	5	6	6	6	2	2
P4	2	3	5	4	4	3	5	6	2	0	0	2
P5	0	3	3	2	0	6	5	2	0	3	2	0

Is this system currently deadlocked, or can any process become deadlocked? Why or why not? If not deadlocked, give an execution order.

Not deadlocked and will not become deadlocked.

Using the Banker's algorithm: P1, P4, P5, P2, P3.

If a request from a process P1 arrives for (0, 4, 2, 0), can the request be immediately granted? Why or why not? If yes, show an execution order.

No, the request is invalid as it would exceed the maximum need that P1 specified at the time of its creation.

If a request from a process P2 arrives for (0, 1, 2, 0), should the request be immediately granted? Why or why not? If yes, show an execution order.

No, the request is valid but if granted, the resulting Currently Available Resources would be (2, 0, 0, 0) and there is no sequence of process executions that would allow the completion of all processes. This is an UNSAFE state.