**CSCI 3431: Operating System**

**Fall 2018**

**Date: October 24, 2018**

**Due: October 31, 2018**

**Home Work #7**

**Name: Sabbir**

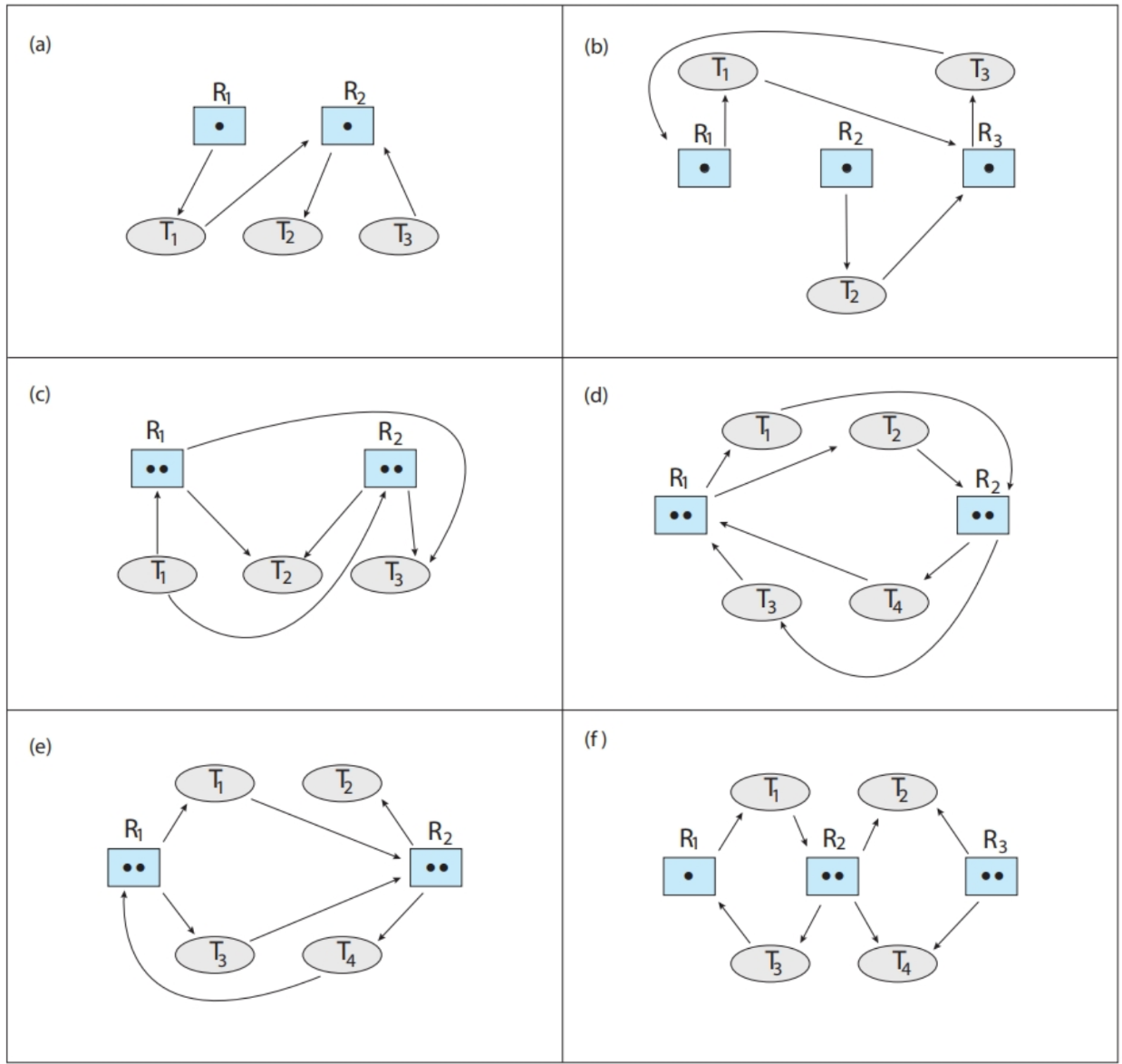
**A#:A00394768**

*Please write your solutions in this file and upload it to moodle*

**Description:** This homework is designed to make you practice deadlock problem solving and programming.

NOTE: you can use C/C++/Java for this assignment.

1. Which of the six resource-allocation graphs shown in Figure 1 illustrate deadlock? For those situations that are deadlocked, provide the cycle of threads and resources. Where there is not a deadlock situation, illustrate the order in which the threads may complete execution. (**30 points**)



Ans: Graph A is a not deadlock -> no circular dependency.

Graph B is a deadlock - > T1 and T2 are requesting R3, which is held by T3.

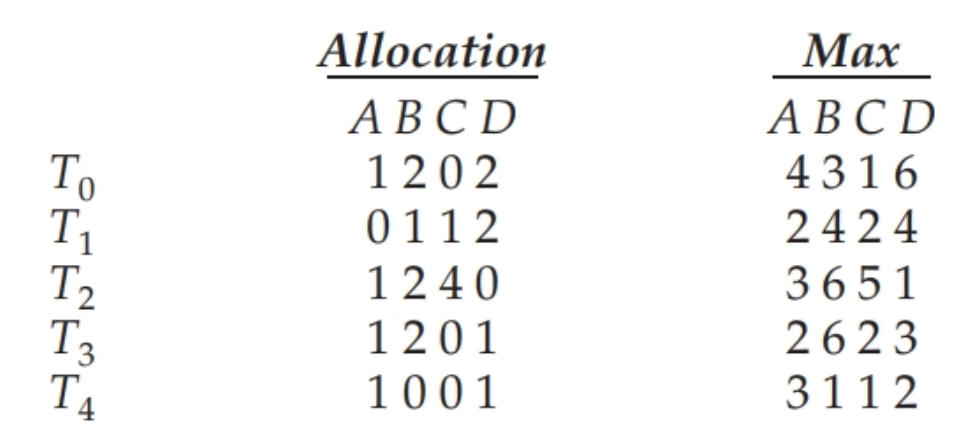
Graph C is a deadlock -> R1 has 2 resources held in T2 , T3 and T1 are requesting a resource to R1. Same for R2.

Graph D is not a deadlock -> no process is waiting for a resource already taken by another process.

Graph E is a deadlock -> there is a circular wait between R1 allocating a resource to T3, who is asking to R2, who is in turn giving to T4 and which is requesting to R1.

Graph F is not a deadlock -> there is not circular wait between R1, T1, R2 and T3. The other circle has no deadlock at all.

1. Consider the following snapshot of a system:



Using the banker’s algorithm, determine whether or not each of the following states is unsafe. If the state is safe, illustrate the order in which the threads may complete. Otherwise, illustrate why the state is unsafe.

a. Available = (2,2,2,3)

b. Available = (4,4,1,1)

c. Available = (3,0,1,4)

d. Available = (1,5,2,2)

(**20 points**)

Ans: a)

T4 T0 T1 T2 T3

b)

T2 T4 T1 T3 T0

c)

The state is unsafe since the resources available are less than what is needed by each thread.

d)

T3 T4 T0 T1 T2

1. A single-lane bridge connects the two Vermont villages of North Tunbridge and South Tunbridge. Farmers in the two villages use this bridge to deliver their produce to the neighboring town. The bridge can become deadlocked if a northbound and a southbound farmer get on the bridge at the same time. (Vermont farmers are stubborn and are unable to back up.) Using semaphores and/or mutex locks, design an algorithm (can be in pseudocode) that prevents deadlock. Initially, do not be concerned about starvation (the situation in which northbound farmers prevent southbound farmers from using the bridge, or vice versa). Modify your solution so that it is starvation free. Attach the code as well as a snap shot of the execution.

NOTE: if the solution is implemented in code, please print some relevant information, e.g., sizes of queues on both sides of the bridge. Example:

[South: 12, North: 10]

[South: 11, North: 9]

[South: 11, North: 3]

**Marking:**

* **20** points for non starvation-free solution (10 points if implemented in pseudocode or 20 points for implementation in C/C++/Java)
* **30** points for starvation-free solution (15 points if implemented in pseudocode or 30 points for implementation in C/C++/Java)

References:

Operating System Concepts, Silbershatz, Gegne and Galvin.