# Homework #3 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ \_\_\_\_\_

•Grading: 3 = correct

2 = almost

1 = an attempt

0 = nothing

•Score: Points / Possible

# (38 points) (Name) (Section)

**Chapter 5 – Mutual Exclusion**

**Chapter 6 – Deadlock / Starvation**

|  |  |
| --- | --- |
| Questions: | Answers: |
| 1. (6.1) (4 points) Show how the four conditions of deadlock might apply to the following figure:  Intersection |  |
| 2. (6.3) (15 points) Consider the following snapshot of a system. (There are no outstanding unsatisfied requests for resources.)  |  |  |  |  | | --- | --- | --- | --- | | Available | | | | | r1 | r2 | r3 | r4 | | 2 | 1 | 0 | 0 |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | |  | Current Allocation | | | | Maximum Demand | | | | |  | r1 | r2 | r3 | r4 | r1 | r2 | r3 | r4 | | P1 | 0 | 0 | 1 | 2 | 0 | 0 | 1 | 2 | | P2 | 2 | 0 | 0 | 0 | 2 | 7 | 5 | 0 | | P3 | 0 | 0 | 3 | 4 | 6 | 6 | 5 | 6 | | P4 | 2 | 3 | 5 | 4 | 4 | 3 | 5 | 6 | | P5 | 0 | 3 | 3 | 2 | 0 | 6 | 5 | 2 |  a. Compute what each process still might request and display in the columns labeled “still needs”.b. Is this system currently in a safe or unsafe state? Why?c. Is this system currently deadlocked? Why or why not?d. Which processes, if any, are or may become deadlocked?e. If a request from P3 arrives for (0,1,0,0), can that request be safely granted immediately? In what state (deadlock, safe, unsafe) would immediately granting that whole request leave the system? Which processes, if any, are or may become deadlocked if this whole request is granted immediately? |  |
| 3. (6.2) (4 points) Apply the deadlock detection algorithm of Section 6.4 to the following data and show the results:Available = | 2 1 0 0 || 2 0 0 1 | | 0 0 1 0 |Request = | 1 0 1 0 | Allocation = | 2 0 0 1 || 2 1 0 0 | | 0 1 2 0 | |  |
| 4. (6.3) (6 points) Consider a system with a total of 150 units of memory, allocated to three processes as shown:  |  |  |  | | --- | --- | --- | | Process | Max | Hold | | 1 | 70 | 45 | | 2 | 60 | 40 | | 3 | 60 | 15 |  Apply the banker’s algorithm from Section 6.3 to determine whether it would be safe to grant each of the following requests. If yes, indicate a sequence of terminations that could be guaranteed possible. If no, show the reduction of the resulting allocation table.a. A fourth process arrives, with a maximum memory need of 60 and an initial need of 25 units.b. A fourth process arrives, with a maximum memory need of 60 and an initial need of 35 units. |  |
| 5. (6.5) (9 points) Consider the following ways of handling deadlock:(1) banker’s algorithm,(2) detect deadlock and kill thread, releasing all resources,(3) reserve all resources in advance,(4) restart thread and release all resources if thread needs to wait,(5) resource ordering, and(6) detect deadlock and roll back thread’s actions.a. One criterion to use in evaluating different approaches to deadlock is which approach permits the greatest concurrency. In other words, which approach allows the most threads to make progress without waiting when there is no deadlock. Give a rank order from 1 to 6 for each of the ways of handling deadlock just listed, where 1 allows the greatest degree of concurrency. Comment on your ordering.b. Another criterion is efficiency; in other words, which requires the least processor overhead. Rank order the approaches from 1 to 6, with 1 being the most efficient, assuming that deadlock is a very rare event. Comment on your ordering.c. Does your ordering from (b) change if deadlocks occur frequently?Note: Although the answers are somewhat subjective, your most and least rankings should be correct. In any case, justify your answers. |  |