0 Instructions

- Place your **NAME**, **NETID** and **CUID** at the front of your booklet. Your CUID is the 6-number code on your ID card. If you do not have a CUID, put down your Social Security Number.
- Answers should be **clear** and **concise**. Unnecessarily long and/or obscure answers will lose credit.
- If you have questions, raise your hand, and we will come around to answer them.
- If you have only a partial solution for a problem and you can't figure out the rest, **put down the partial solution and explain (briefly) why it does not work**. That will get you more credit than a broken solution by itself.
- Partial credits will be given for incomplete answers.
- Use your time wisely. You may want to skip hard problems first and the come back after you have finished others.

1 Process Scheduling and Synchronization

1. Consider a simple priority-based CPU scheduling algorithm, which always picks the process with the highest priority in the ready queue. Explain briefly how such a scheme could lead to starvation, and present a small modification to the scheme to ensure no starvation. (**5 points**)

2. A student comes up with the following algorithm (slightly modified from the solution presented in the textbook) for solving the critical section problem for two processes P_0 and P_1. We use P_i and P_j, where i is either 0 or 1, and $j = 1 - i$ is the index of the other process. Initially, *flag[0]* and *flag[1]* are set to *false*.

   Entry code for process P_i:

   $\quad$ *flag*[i] := *true*;

   $\quad$ *turn* := i;

   $\quad$ while (*flag*[j] and *turn* $= j$) do *no-op*;

   Exit code for process P_i:

   $\quad$ *flag*[i] := *false*;

   Does this solution satisfy *Mutual Exclusion*, *Progress*, and *Bounded Waiting*, the three properties of critical sections? Provide a proof for every property that is satisfied, and give a counterexample for every requirement that is not. (**15 points**)

3. Using a monitor, write a solution to the bounded-buffer problem, where a producer repeatedly stores the item it produces into a buffer of n slots when there are empty slots, and a consumer repeatedly retrieves an item, when there is any available, from the buffer and process that item.

   Define a monitor type for the solution. The definition should include the declaration of necessary local and conditional variables, entry and local procedures, and the initialization code. Then, provide the pseudo code for both the producer and the consumer (ignore how an item is produced or consumed). (**15 points**)

2 Deadlock Prevention, Avoidance, and Detection

1. Consider an electronic fund transfer system, where there are a large number of concurrent processes, each works as follows: a process first gets an input that specifies the account number from which the money comes (i.e., the origin of the transfer), the account number to which the money goes (i.e., the destination of the transfer), and the amount to be transferred; the process locks the origin account and the destination account; the process then does the transfer and releases the locks when the transfer is complete.

   There is a danger for deadlocks in such a system. Assuming that lock preemption is not allowed, that a process does not release locks until the transaction is complete, and that a process can only lock one account at a time, propose a solution that can PREVENT deadlocks in such a system. Please describe your scheme in detail and prove why such a scheme prevents deadlocks. (**10 points**)

2. For the electronic fund transfer system described in the previous question, assume that no deadlock prevention or deadlock avoidance mechanisms have been incorporated, describe a deadlock detection algorithm (assume

that the algorithm has obtained information about which account(s) a process has locked and which account(s) a process requests to lock). Describe the algorithm in detail and explain why it works. (**10 points**)

3. To avoid deadlock, the Banker's algorithm always makes sure that the system stays in a *safe* state and never enters an *unsafe* state. (**15 points**)
   (a) If a system is in an unsafe state, is deadlock bound to happen? Explain why.
   (b) Consider a system that is initially in a safe state. A new process is now admitted into the system. If the maximum requirement for resources by this process can be satisfied by the current available resources in the system, will the admission of the process causes the system to enter an unsafe state? Explain your answer.
   (c) If a new process's maximum requirement for resources cannot be satisfied by the current available resources in the system, but can be satisfied by the the maximum available resources in the system (i.e., the current available resources in the system plus all the resources that have been allocated to other processes), will the admission of the process causes the system to transit from a safe state to an unsafe state? Explain your answer.

3 Memory Management

1. Paging and Segmentation.(**15 points**)
   (a) Compare paging with segmentation in terms of characteristics of partitions (i.e., pages or segments), fragmentation, and protection.
   (b) Briefly explain how the two mechanisms can be combined for improvement and list the advantages of the hybrid scheme.
   (c) Are there any disadvantages of the hybrid scheme with respect to pure paging or pure segmentation? List at least one and propose a way to alleviate the problem. Explain your answers.

2. Interaction between memory management and process management. For a paging system, consider the following two ways to implement the page table: (1) implement the page table as a set of dedicated registers and (2) store the page table in the memory and have a dedicated page-table register point to the page table. (**15 points**)
   (a) For each implementation, explain what information has to be stored in a PCB (process control buffer) and what need to be done (related to memory management) during context switch. Which scheme would likely lead to faster context switch? Explain why.
   (b) Which of the two would likely lead to faster memory access? Explain why?
   (c) In scheme (2), if TLB (Translation Lookaside Buffer) is used as a cache for mappings from a page number to a frame number, what needs to be done during context switch in order to achieve memory protection? Explain your answer.