Answers to Homework #2

Due Date: February 1, 2001 **Points**: 60

1. (*10 points*) Number all the forks in the Dining Philosopher's problem and require that each philosopher request an even-numbered fork before an odd-numbered fork. Will this allocation strategy prevent deadlock and starvation? Is it a form of a well-known strategy (named in section 3.9.3)?
Answer: Yes. This solution avoids deadlock because there is no longer any symmetry in the way the philosophers pick up their forks (that is, all cannot pick up their left forks simultaneously.) Assuming philosophers pick up forks in the order each fork is requested, it avoids starvation because only one philosopher can pick up either fork; the others are constrained to pick up one particular fork, and in some cases that fork is the left one while in other cases that fork is the right one. This is an example of the heirarchical allocation rule.
2. (*15 points*) Using the definitions given in class, prove that "S is not a deadlock state" does not imply that "S is a safe state."
Answer: Let S be a state other than a deadlocked state. Then there exists no process p_i which is deadlocked in S. By definition this means there is at least one state T for which $S \rightarrow^* T$ and p_i is not blocked in T. Note the words "at least" in the above sentence; it is certainly possible that there is some state T' for which $S \rightarrow^* T'$, p_i is blocked in T', and for every state T'' reachable from T', p_i is blocked. In this case there is at least one state T' reachable from S that is a deadlock state; hence, S is not a safe state.
3. (*15 points*) Assume a system has p processes and r identical units of a reusble resource. If each process can claim at most n units of the resource, show that the system will be deadlock free if, and only if, $r \geq p(n-1)+1$ [text, problem 3.7].
Answer: ($\Rightarrow$) Assume the system is deadlock free. At least one process must have all its resource requests satisfied. In the worst case, all processes request n units of the resource. If only one process is to have its requests satisfied, each process can hold up to $n-1$ units of the resource, with one unit left over (to satisfy one of the processes). Hence there must be at least $p(n-1)+1$ units of the resource, as claimed.
($\Leftarrow$) Assume a system of the sort described above has at least $p(n-1)+1$ units of a resource. Let p processes request n resource units. Each process can acquire $n-1$ units of the rsource, and the remaining unit will be given to one of theprocesses. The others block. However, that one process can run to completion. It then releases n units. If $n \geq p-1$, all processes complete. If not, n more processes complete, and release n^2 units of the resource. This continues until all processes complete. Hence the system is deadlock free.
4. (*20 points*) Prove the Cycle Theorem (theorem 3.6).
Answer: Necessity first. Assume the graph has no cycle. We shall show first that there exists a linear ordering of the nodes such that, if there is a path from node i to node j, then i appears before j in that ordering. Once this is established, it is clear that there is a sequence of reductions that completely reduces the resource graph; by the deadlock theorem, this means that the graph is not in a deadlock state.
Let us now construct the required ordering from a digraph without cycles. In a digraph there are two possible relations based on paths: either there is a path from a node i to a node j, or there is no such path. If there is no such path, the order of the vertices in our ordering is immaterial. If such a path exists, we put i before j in our ordering. Now suppose there are three nodes n_1, n_2, and n_3 in our ordering for which n_1 precedes n_2, n_2 precedes n_3, and n_3 precedes n_1. Then there is a path from n_1 to n_2, a path from n_2 to n_3, and a path from n_3 to n_1 — in short, a cycle, contradicting our hypothesis. Hence this situation can never arise and indeed our ordering can be linear.
To see sufficiency, let us assume the serially reusable resource graph contains a cycle. Consider only those processes and resources belonging to the cycle. Each process node in the cycle must have an outgoing and incoming edge, so each process must hold a resource in the cycle as well as have an outstanding request for some resource in the cycle. Similarly, every resource in the cycle must be held by a process in the cycle (since each resource has an inventory of 1 unit.) Therefore, every process in the cycle is blocked on a resource in the cycle that is being held by some other process in the cycle. Thus, all the processes in the cycle are deadlocked.