Algorithms
Computer Science 140 & Mathematics 168
Instructor: B. Thom
Fall 2004
Homework 15a
Due on Thursday, December 9, 2004 (beginning of class)

1. **[25 Points] The Disk Storage Problem!** First the gratuitious story. You have been hired by Moon Microsystems to work on their new operating system, Lunaris. When Lunaris does a backup, it typically uses two large backup disks and a tape drive. Since disks have lower access time than tape, it is desirable to store as many files as possible on disk and the remainder will go on tape. Let $F = \{f_1, \ldots, f_n\}$ denote the n files to be backed up and let ℓ_i denote the length of file i. Without loss of generality, let $\ell_1 \leq \ell_2 \leq \ldots \leq \ell_n$.

   There are two identical disks, each with storage capacity of L. A file stored on disk cannot be broken up - it must be stored entirely on one of the two disks. The Disk Storage Optimization Problem is to store the maximum number of files from F onto the disks.

   Your boss has suggested that you implement a greedy algorithm for the Disk Storage Optimization Problem: Since the files in F appear in non-decreasing order of size, simply go through F from shortest file to longest file, putting as many of the files on the first disk as possible. When the first disk fills up, continue iterating through F putting as many of the remaining files as possible on the second disk.

   (a) Give an example that demonstrates that the greedy algorithm does not necessarily find an optimal solution. That is, give a set of specific file sizes, show the solution found by the greedy algorithm, and then the optimal solution. Show that the solution found by the greedy algorithm is not optimal.

   (b) State the decision problem corresponding to this optimization problem. Then prove that the decision problem is NP-complete. Use a reduction from one of: Vertex Cover, 3SAT, Partition, Traveling Salesman, Clique, or Undirected Hamiltonian Cycle.

   (c) Prove that the greedy algorithm is an approximation algorithm which finds solutions that are at most 1 smaller than optimal. This is really neat, since all of the approximation algorithms we've seen so far were some *multiplicative* factor worse than optimal (typically 2 times optimal). Here, the algorithm finds solutions which are an *additive* amount worse than optimal!