Due start of class Monday, November 1, 2010.

**Problem 1.** Suppose you are given an $n \times n$ chess board. You may place your king on *any* of the $n$ bottom squares and move it to *any* of the $n$ top squares. Your king may only move up. So from any square it can only move to the square immediately above it, immediately above it to the left (assuming it is not already in the leftmost column), or immediately above it to the right (assuming it is not already in the rightmost column).

Let $v(x, y)$ be the value of moving the king from square $x$ to square $y$ (which may be negative). The value of a path is the sum of the values of the individual moves.

The goal is to find the maximum value of a path from any one bottom square to any one top square.

The goal of this assignment is to find an efficient dynamic programming algorithm for solving this problem.

HINT: Let $V[x]$ be the maximum value of any path from a bottom square to square $x$.

(a) Write down a recurrence for $V[x]$. (You may want to write $x$ as an ordered pair.)

(b) Based on your recurrence, give a recursive program for finding value of the “best” path. What can you say about its running time?

(c) Show how to MEMOIZE the recursive algorithm. What is the running time?

(d) Based on your recurrence, give an efficient dynamic programming algorithm for finding the value of the “best” path. What is the running time?

(e) Modify your algorithm to actually find the “best” path (not just its value).

**Problem 2.** Give a dynamic programming algorithm for solving the knapsack problem that also finds the optimal packing. (See Kleinberg and Tardos, pages 271-272.)

**Problem 2.** Do Exercise (4) on pages 315-316 of Kleinberg and Tardos.