1. Recall the 0-1 Knapsack Problem described in class. A thief wishes to steal \( n \) objects having values \( v_i > 0 \) and weights \( w_i > 0 \) \( (1 \leq i \leq n) \). His knapsack, which will carry the stolen goods, holds at most a total weight \( W \). Let \( x_i \in \{0,1\} \) to be the indicator variable for this problem, i.e. for \( 1 \leq i \leq n \), \( x_i = 1 \) if object \( i \) is taken, \( x_i = 0 \) if object \( i \) is not taken. The thief’s goal is then to maximize the total value \( \sum_{i=1}^{n} x_i v_i \), subject to the capacity constraint \( \sum_{i=1}^{n} x_i w_i \leq W \).

   a. Write pseudo-code for a dynamic programming algorithm which solves this problem. Your algorithm should take as input the value and weight arrays \( v[1..n] \) and \( w[1..n] \), and the weight limit \( W \). It should generate a table \( V[1..n; 0..W] \) of intermediate results. Each entry \( V[i, j] \) is the maximum value of the objects which can be stolen from the set \( \{1,\ldots,i\} \) \( (1 \leq i \leq n) \) using a knapsack having capacity \( j \) \( (0 \leq j \leq W) \). Your algorithm should return the maximum possible value of the goods which can be stolen from the full set of objects, i.e. the value \( V[n,W] \). (Alternatively you may write your algorithm to return the whole table.)

   b. Write an algorithm which given the filled table generated in part (a), prints out a list of exactly which objects are to be stolen.

2. **Canoe Rental Problem.** There are \( n \) trading posts numbered 1 to \( n \) as you travel downstream. At any trading post \( i \) you can rent a canoe to be returned at any of the downstream trading posts \( j > i \). You are given an array \( R[i, j] \) which gives the cost of a canoe which is picked up at post \( i \) and dropped off at post \( j \). for \( 1 \leq i < j \leq n \). Assume that \( R[i, i] = 0 \) and that you can’t take a canoe upriver (so perhaps \( R[i, j] = \infty \) when \( i > j \)). Your problem is to determine a sequence of rentals which start at post 1 and end at post \( n \), and which has a minimum total cost. As usual there are really two problems: determine the cost of a cheapest sequence, and determine the sequence itself.

Design a dynamic programming algorithm for both problems. What are the dimensions of your table and what are its entries? How is it initialized and what is the recurrence which determines table entries? Why is the principle of optimality satisfied in this problem? Write algorithms which fill the table given the cost array \( R \), and which determine an optimal sequence given the filled table. Determine the asymptotic run time of your algorithms.