1. **Championship Showdown**

   What excitement! The Champaign Spinners and the Urbana Dreamweavers have advanced to meet each other in the World Series of Basketweaving! The World Champions will be decided by a best of $2n-1$ series of head-to-head weaving matches, and the first to win n matches will take home the coveted Golden Basket (for example, a best-of-7 series requires four match wins, but we will keep the generalized case). We know that for any given match there is a constant probability p that Champaign will win, and a subsequent probability $q = 1-p$ that Urbana will win.

   Let $P(i,j)$ be the probability that Champaign will win the series given that they still need i more victories, whereas Urbana needs j more victories for the championship. $P(0,j) = 1$, $1 \le j \le n$, because Champaign needs no more victories to win. $P(i,0) = 0$, $1 \le i \le n$, as Champaign cannot possibly win if Urbana already has. $P(0,0)$ is meaningless. Champaign wins any particular match with probability p and loses with probability q, so

   $$P(i,j) = p \cdot P(i-1,j) + q \cdot P(i,j-1)$$

   for any $i \ge 1$ and $j \ge 1$.

   Create and analyze an $O(n^2)$-time dynamic programming algorithm that takes the parameters n, p, and q and returns the probability that Champaign will win the series (that is, calculate $P(n,n)$).

2. **Making Change**

   Suppose you are a simple shopkeeper living in a country with n different types of coins, with values $1 = c[1] < c[2] < \cdots < c[n]$. (In the U.S., for example, $n = 6$ and the values are $1, 5, 10, 25, 50$, and 100 cents.) Your beloved benevolent dictator, El Generalissimo, has decreed that whenever you give a customer change, you must use the smallest possible number of coins, so as not to wear out the image of El Generalissimo lovingly engraved on each coin by servants of the Royal Treasury.

   Describe and analyze a dynamic programming algorithm to determine, given a target amount A and a sorted array $c[1..n]$ of coin values, the smallest number of coins needed to make A cents in change. You can assume that $c[1] = 1$, so that it is possible to make change for any amount A.

3. **Knapsack**

   You are a thief, who is trying to choose the best collection of treasure (some subset of the n treasures, numbered 1 through n) to steal. The weight of item i is $w_i \in \mathbb{N}$ and the profit is $p_i \in \mathbb{R}$. Let $C \in \mathbb{N}$ be the maximum weight that your knapsack can hold. Your goal is to choose a subset of elements $S \subseteq \{1, 2, \ldots, n\}$ that maximizes your total profit $P(S) = \sum_{i \in S} p_i$, subject to the constraint that the sum of the weights $W(S) = \sum_{i \in S} w_i$ is not more than C.

   Give an algorithm that runs in time $O(Cn)$.