1. Consider the first approach to the longest common subsequence problem, where we solved the problem using the forward approach. Now, solve it using the backward approach. Your answer must include: the definition of \( L(i,j) \), the definition of \( \phi(k,j) \), the recurrence, boundary conditions, and the goal.

2. Implement the third approach of dynamic programming to the longest common subsequence problem. Your algorithm needs to print the actual longest common subsequence. Specifically, write two procedures: (1) a non-recursive procedure to compute \( L(i,j) \), \( 1 \leq i,j \leq n \), and (2) a recursive procedure \( \text{Longest}(i,j) \) such that \( \text{Longest}(1,1) \) will print the longest common subsequence.

3. Consider the all-pair shortest paths problem. Suppose the global arrays \( D[1..n, 1..n] \) and \( P[1..n, 1..n] \), \( 1 \leq k \leq n \), have been computed as in Floyd’s algorithm. Write a recursive procedure \( \text{Path}(i,j) \) such that a call to \( \text{Path}(i,j) \) will print the shortest path from \( i \) to \( j \). Note: a path is a sequence of vertices. You may print a vertex more than once (e.g., it is OK to print a path \( (a,b,c,d) \) as \( (a,a,b,c,c,c,d) \).)