Task 2:
Traveling Salesman Problem
Create a function int FindMinCost(int[,] graph)
that takes a 2D array representing the graph where graph[i][j] is the cost to travel from city i to city j. The function should return the minimum cost to visit all cities and return to the starting city. Use dynamic programming for this solution.
ANS:

```
package com.Day19;
import java.util.Arrays;
public class TravelingSalesman {
  public static int FindMinCost(int[][] graph) {
    int n = graph.length;
    int VISITED_ALL = (1 << n) - 1;
    int[][] dp = new int[n][1 << n];

    // Initialize dp array with -1 (indicating not computed)
    for (int[] row : dp) {
      Arrays.fill(row, -1);
    }
    return tsp(graph, 1, 0, dp, VISITED_ALL);
  }
  private static int tsp(int[][] graph, int mask, int pos, int[][] dp, int VISITED_ALL) {
    // Base case: all cities have been visited
    if (mask == VISITED_ALL) {
      return graph[pos][0]; // return to starting city
    }
    // Check if the answer is already computed
    if (dp[pos][mask] != -1) {
      return dp[pos][mask];
    }
    int minCost = Integer.MAX_VALUE;
    // Try to go to any other city that is not yet visited
    for (int city = 0; city < graph.length; city++) {
      if ((mask & (1 << city)) == 0) {
        int newCost = graph[pos][city] + tsp(graph, mask | (1 << city), city, dp, VISITED_ALL);
        minCost = Math.min(minCost, newCost);
```

```
            }
        }
        // Save the result in dp array and return
        dp[pos][mask] = minCost;
        return minCost;
    }
    public static void main(String[] args) {
        int[][] graph = {
            {0, 10, 15, 20},
            {10, 0, 35, 25},
            {15, 35, 0, 30},
            {20, 25, 30, 0}
        };
        System.out.println("The minimum cost to visit all cities and return to the starting city is: " + FindMinCost(graph));
    }
}
```

OUTPUT:

```
The minimum cost to visit all cities and return to the starting city is: 80
```