A startup is developing a simple grid-based adventure game where an intelligent character must learn how to reach a treasure without prior knowledge of the environment. The game environment is represented as a 4 × 4 grid, where The agent starts at position (0, 0), The goal (treasure) is located at (3, 3) and There are dangerous obstacle cells at (1, 1) and (2, 2). The agent can take the following actions: Move Up, Down, Left, Right. Each action leads to a new state based on the grid boundaries.

Reward Policy: +10 → Reaching the treasure, -5 → Entering an obstacle cell, -1 → Every movement step. The company wants the agent to learn the optimal path automatically using Q-learning, so that it reaches the goal in the least number of steps while avoiding obstacles. Develop a Python program to implement Q-learning for this problem.

1. Model the Environment
    - Represent the grid as states
    - Define possible actions for each state
    - Ensure the agent does not move outside the grid

2. Initialize Q-Learning Components
    - Create a Q-table initialized with zeros
    - Set parameters:
        - Learning rate (α)
        - Discount factor (γ)
        - Exploration rate (ε)

3. Implement Q-Learning Algorithm
    - Train the agent for multiple episodes (e.g., 100–200)
    - Use epsilon-greedy strategy:
        - Explore (random action)
        - Exploit (best Q-value action)
    - Update Q-values using the Q-learning formula

4. Train the Agent
    - Simulate episodes until the agent reaches the goal
    - Update Q-table after every step

5. Display Results
    - Print the final Q-table
    - Show the optimal path from start to goal
    - Display total reward obtained