Design and Assumptions

Design:

The code implements a simple Minesweeper game in C#. It handles user inputs for:

1. **Board Size**: Getting the size of the game board.
2. **Number of Mines**: Determining how many mines to place.
3. **Cell Coordinates**: Parsing and validating cell coordinates for revealing parts of the board.

Assumptions for UserInputManager Class:

Methods:

- GetBoardSize(): Asks for a board size between 2 and 10.
- GetNumberOfMines(int boardSize): Asks for a number of mines between 1 and 35% of the board's total cells.
- TryParseCoordinates(string input, int boardSize, out int row, out int column): Converts and validates cell coordinates from user input.

Assumptions:

1. **Board Size**: The board size is between 2 and 10 for simplicity.
2. **Number of Mines**: Mines are between 1 and 35% of the total cells to ensure the game remains playable.
3. **Coordinate Format**: Coordinates are in the format "A1", with rows labeled 'A', 'B', etc., and columns numbered starting from 1.

Assumptions for MinesweeperGame Class:

1. **Valid Input Range**: The game assumes that the board size is between 2 and 10. This constraint keeps the game manageable and ensures a balance between simplicity and challenge.
2. **Mine Count Limitation**: The number of mines must be between 1 and 35% of the total number of cells on the board. This prevents the game from becoming too easy (with very few mines) or too difficult (with too many mines).
3. **Coordinate Format**: User inputs coordinates in the format "A1", where:

   - The letter (e.g., 'A') represents the row.
   - The number (e.g., '1') represents the column.
   - Rows are labeled alphabetically starting from 'A'.
   - Columns are labeled numerically starting from 1.

4. **User Interaction**: The game is played through a command line interface, and user inputs are read from the console.

Summary of SOLID Principles Applied:

1. **SRP (Single Responsibility Principle)**:

   - Program handles the main application loop.
   - MinesweeperGame handles the game logic.
   - UserInputManager handles user input and validation.
   - Grid handles the game grid and mine placement logic.
   - Cell represents an individual cell in the grid.

2. **OCP (Open/Closed Principle)**:

    - IMinesweeperGame and IGrid interfaces allow for extension with new game or grid types without modifying existing code.

3. **LSP (Liskov Substitution Principle)**:

    - MinesweeperGame and Grid classes can be substituted for IMinesweeperGame and IGrid interfaces, respectively, ensuring consistent behavior.

4. **ISP (Interface Segregation Principle)**:

    - IMinesweeperGame and IGrid interfaces ensure that classes only implement the methods they need.

5. **DIP (Dependency Inversion Principle)**:

    - MinesweeperGame depends on the IGrid interface rather than a concrete Grid class, allowing for easier substitution and testing.

Running the Minesweeper Game

Environment Requirements:

- **Operating System**: Windows or Linux.
- **.NET Framework**: Ensure that .NET is installed on your machine.