1. Gaming: Player Profile Management

Objective: Manage player profiles, game statistics, and achievements using callback functions, error handling, async/await, and ES6 features.

Part 1: Basic Features

Instructions:

1. **Create a class Player** with properties `name`, `level`, `score`, and `achievements`.
2. **Implement methods to:**
    - Update the player's profile (with error handling).
    - Add a new achievement (with callback function).
    - Fetch player data from a simulated API (using async/await).
    - Save updated player data to a simulated API (using async/await).
3. **Use callback functions** to display updated profile and achievements after fetching and updating data.
4. **Add error handling** to manage simulated API request failures.

Part 2: Advanced Features

Objective: Enhance the player profile management system using advanced ES6 features, including modules, EventEmitter, and streams.

Instructions:

1. **Organize code using ES6 modules**:
    - Split the code into separate modules for Player, API calls, Logger, and Streams.
2. **Add an event-driven logging system** using EventEmitter:
    - Log significant actions such as fetching data, updating profile, and saving data.
3. **Implement stream operations**:
    - Create readable, writable, duplex, and transform streams to handle player data.
    - Use `pipe()` for efficient data flow.
4. **Enhance error handling** to manage stream operations and API request failures.