Problem 1 –

Game Scenario: "Battle Royale - Last Player Standing"

Description:

The game takes place on a 2D grid representing a battlefield. Multiple players participate in a Battle Royale style competition, where they must outmaneuver and outsmart their opponents to be the last player standing. Each player has health points (HP), and the objective is to reduce the HP of other players to zero while keeping their own HP positive.

Requirements:

- Create a 2D grid of a fixed size (e.g., 10x10) to represent the battlefield. Each cell in the grid can be either empty or occupied by a player.
- Each player is a separate entity with attributes such as position (x, y), HP, and a unique identifier (ID).
- Each player will have their own thread to simulate their movements and actions independently. • Players can move around the grid one step at a time, either up, down, left, or right, but cannot move to a cell occupied by another player or outside the grid boundaries.
- Players can also attack other players who are within a certain range, which reduces the target player's HP. The attack range can be limited to adjacent cells or a fixed number of cells away. • The game should display the current state of the battlefield after each turn, showing the positions and HP of each player.
- The game should continue until only one player remains with positive HP, and that player will be declared the winner.

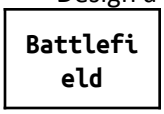
Implementation:

- Player Class:

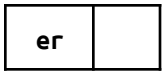
  The class should have methods to handle player movements, attacks, and other actions.

- Battlefield Class:

  Design a class to manage the 2D grid and handle interactions between players.



The class will have a 2D array to represent the grid, where each cell can hold a pointer



to a object or be empty.

Implement methods in the class to update player positions, handle attacks, and

check for collisions.

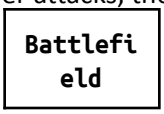
• Player Movement:

Each player's thread will execute a method to move the player randomly in one of the four cardinal directions (up, down, left, right).

Ensure that the player does not move to an occupied cell or move outside the grid boundaries. • Player Attacks:

Players will also have a method to attack other players within the attack range.

When a player attacks, the class will check if there are any other players within the



attack range and reduce their HP accordingly.

Implement proper synchronization to handle attacks on multiple players at the same time. • Game Loop:

Create a game loop that continues until only one player remains or until all but one player have been eliminated.

In each iteration of the game loop, update player movements and resolve any attacks that occur.

Display the current state of the battlefield after each turn, showing the positions and HP of each player.

• Determine the Winner:

Continuously monitor the number of players remaining with positive HP.

Once only one player remains with positive HP, declare that player as the winner of the Battle Royale.

This implementation will create a challenging and dynamic game where players must strategize their movements and attacks to outmaneuver opponents and be the last player standing. Remember to handle synchronization and thread safety carefully to avoid race conditions and ensure the game runs smoothly. As the game progresses, players will be eliminated one by one until only one player emerges victorious, creating an exciting and complex gaming experience.