**Assignment One**

**Grade Weight 10%**

**DO NOT USE Visual Studio for this assignment**

**Use Notepad++ instead**

In this assignment, you will create a console-based simulation of a battle between two teams of warriors. Each team has two types of warriors: Soldiers and Queens. The warriors will have health, power, and damagePower, which will be used in combat. The battle will take place in a randomized world, where each warrior can move and attack during each round. The goal is to create a game where the two teams fight until one of the teams is defeated, and the winner is printed on the console.

**Problem Description:**

You need to implement the following components for the game:

1. **Warrior Class**:
   * Define a Warrior class with the following properties:
     + health: an integer that represents the health of the warrior.
     + power: an integer that represents the base power of the warrior.
     + damagePower: an integer that represents how much damage the warrior can deal during an attack.
     + x and y: the warrior's position in the game world (coordinates on a 2D grid).
   * Methods:
     + move(): Move the warrior randomly within a radius of 1 unit. The warrior should be able to move to one of the 8 adjacent cells (diagonals included).
     + attack(): The warrior will deal damage to an enemy by subtracting damagePower from the enemy’s health.
     + isAlive(): Returns true if the warrior's health is greater than 0, otherwise false.
     + getType(): Returns the type of the warrior ("Soldier" or "Queen").
2. **Soldier and Queen Classes**:
   * Create two derived classes, Soldier and Queen, that inherit from Warrior.
   * The Soldier class will have:
     + Health: 100
     + Power: 10
     + DamagePower: 15
   * The Queen class will have:
     + Health: 150
     + Power: 20
     + DamagePower: 25
   * These classes will override the attack() method if necessary (based on the specific type of damage).
3. **Team Class**:
   * Define a Team class to represent each team in the battle. Each team will consist of an equal number of soldiers and queens.
   * Team should be initialized with a random set of warriors, with each warrior placed in a random position in the world.
   * Methods:
     + addWarrior(): Adds a warrior to the team.
     + attackOpponent(): Each warrior of the team attacks an enemy warrior.
     + isTeamAlive(): Returns true if at least one warrior is alive in the team.
4. **Battle Simulator**:
   * Create a class BattleSimulator that handles the overall game logic:
     + Initialize two teams with warriors at random positions on the battlefield (grid).
     + Simulate rounds of battle, where each team will move and attack. In each round, each warrior will randomly move to an adjacent position and then attack the opponent.
     + After every round, check if one of the teams has no remaining warriors. If so, the other team wins, and the game ends.
5. **Game World**:
   * The game world should be represented as a 2D grid (size 10x10, for example).
   * Each warrior can occupy a specific coordinate (x, y) on the grid, and warriors should be placed at random positions when the game starts.
   * Ensure that warriors cannot occupy the same position on the grid (if a random position is already occupied, re-roll the position until a free one is found).
6. **Main Game Loop**:
   * In the main() function, initialize the two teams, set up the battle world, and simulate the rounds of battle.
   * Print the current status (health, position) of each warrior before and after each round.
   * Declare and print the winner of the battle when one team is completely defeated.

**Key Game Mechanics:**

* **Warrior Movement**: Each warrior can move one unit in any direction (including diagonals) at each step. For example, if a warrior is at position (5, 5), they can move to any of the following positions: (4, 4), (4, 5), (4, 6), (5, 4), (5, 6), (6, 4), (6, 5), (6, 6).
* **Attack**:
  + A warrior attacks another by reducing its opponent's health by the warrior's damagePower.
  + If a warrior’s health drops to zero or below, they are considered defeated and can no longer participate in the battle.
* **Victory Condition**:
  + A team wins when all of the opposing team's warriors are dead.
  + Once one team is defeated, print the winner and end the game.

### Requirements:

* Use **object-oriented programming** (OOP) principles to design and implement the game.
* Implement **randomization** to ensure that warriors are placed at random positions on the battlefield at the start of the game.
* Display all relevant information (warrior type, health, power, position) clearly on the console at each step of the game.
* The game should run in the console with no graphics involved.

**Submission Guidelines:**

* Submit the source code (.cpp and .h files) along with a README file explaining how to compile and run the program.
* Ensure that your code is well-commented and follows good programming practices (e.g., clear variable names, modular structure).

**Example Output:**

**Welcome to the Battle Simulator!**

**Team A and Team B are about to fight...**

**Round 1:**

**Team A Warrior (Soldier) at (2, 3) attacks Team B Warrior (Queen) at (5, 6) for 25 damage.**

**Team B Warrior (Queen) is still alive with 125 health.**

**Team A Warrior (Queen) at (1, 4) attacks Team B Warrior (Soldier) at (3, 7) for 15 damage.**

**Team B Warrior (Soldier) is still alive with 85 health.**

**Round 2:**

**Team A Warrior (Soldier) at (2, 3) moves to (3, 3).**

**Team B Warrior (Queen) at (5, 6) moves to (6, 7).**

**Round 3:**

**Team A Warrior (Soldier) at (3, 3) attacks Team B Warrior (Soldier) at (3, 7) for 15 damage.**

**Team B Warrior (Soldier) is still alive with 70 health.**

**...**

**Team A wins the battle!**