Avengers Battle Simulation

Objective

- Design a **Quantum Suit Management System** for the Avengers using Object-Oriented Programming (OOP) principles.
- Use **polymorphism**, **operator overloading**, and **function overloading** to manage real-time suit upgrades, attacks, and defense.
- Simulate a battle scenario between Avengers and their enemies.

Background Story

The Avengers are gearing up for another intergalactic battle. Tony Stark has developed Quantum Nanotech Suits (QNS) that dynamically adapt and upgrade based on battlefield conditions. These suits enable Avengers to maximize their chances of victory by:

- Absorbing energy from attacks to upgrade their power. ● Reinforcing armor dynamically to increase durability.
- Overclocking power systems for high-damage attacks.

However, there are limitations: excessive upgrades can cause overheating, which may render the suit unusable.

Problem Description

The Avengers have a set of Quantum Nanotech Suits (QNS), each with unique attributes like **power level**, **durability**, **energy storage**, and **heat level**. These suits dynamically upgrade during battle but are prone to overheating if overused. Avengers battle enemies using these suits, and your task is to manage the battle simulation.

You need to:

1. Create and manage a list of Quantum Suits.
2. Create Avengers and assign them suits.
3. Simulate battles and maintain a **battle log** of all events.
4. Handle all possible constraints and edge cases.

Class Definitions

1. **Class: QNS (Quantum Nanotech Suit)**

Represents a dynamic battle suit that upgrades in real time. By default we will consider the suit format as (P,D,E,H).

Attributes:

1. `powerLevel` *(private, int)*: The suit's attack strength. Default val:1000, represented as P
2. `durability` *(private, int)*: Armor strength against attacks. Default val: 500 , represented as D.
3. `energyStorage` *(private, int)*: Stores absorbed energy from enemy attacks. Default val:300, Represented as E
4. `heatLevel` *(private, int)*: Increases with overuse; if too high, the suit shuts down.

Default val:0, Represented as H.

Methods:

1. **Constructors**:

   ○ Parameterized Constructor: Initializes `powerLevel`, `durability`, `energyStorage`, and `heatLevel`.

   ○ Copy Constructor: Duplicates another QNS instance.
   ○ Default Constructor: Initializes all attributes with safe defaults .

2. **Operator Overloading**:

   ○ **Addition (+)**: used for suit upgradation and to power boost the suit.

   Formula: *(P1, D1, E1,H1) + (P2, D2, E2,H2) = (P1 + E2, D1 + D2, E1 + P2,H1)*

   ○ **Subtraction (−)**: Simulates damage from an attack, *reducing durability and increasing heat.*

Formula: *(P, D, E, H) - X = (P, D - X, E + X, H + X)*

Constraint: If `D ≤ 0`, the suit is destroyed.

- **Multiplication (`*`)**: Activates *Power Boost* mode, amplifying attack strength but raising heat.

  Formula: *(P, D, E, H) * X = (P+(P * X)/100), D, E + 5X, H + X)*

  Constraint: If `H > 500`, the suit overheats.

- **Division (`/`)**: Activates the *cooling system*, *reducing heat* by transferring energy to durability.

  Formula: *(P, D, E, H) / X = (P, D + X, E, H - X)*

  Constraint: Heat cannot drop below 0.If heat becomes < 0 set it to 0

  For simplicity assume the division is int division i.e. taking floor value.

3. **Function Overloading**:
   - `void boostPower(int factor)`: Boosts(add) power, but increases overheating risk i.e. increases heat This method returns void. [hint: can use * operator]
   - `void boostPower(QNS otherSuit)`: Transfers energy from another suit to boost durability. This method returns void. [hint: can use + operator]
4. **Comparison Overloading**:
   - bool Equality (`==`): Compares suits based on `powerLevel` and `durability`. If both are the same returns true, else return false.
   - bool Less Than (`<`): Compares suit effectiveness in battles. If sum of suit1 power and durability is lesser than other return true else returns false.
5. **Other functions**:
   - Getters and Setters for private attributes.

Validation Constraints:

- If `heatLevel > 500`, the suit shuts down.
- If `durability = 0`, the suit is destroyed.

- Total power (`powerLevel`) must not exceed 5000. If it exceeds 5000, then set powerLevel to 5000.
- If heatLevel gets lower than 0 then set heatLevel to 0.
- Suits are named as Qi for ith suit. Eg: 1st suit is named as Q1, 2nd suit is named as Q2. [Suit name can be anything, this is not an mandatory case]

---

2. Class: Avenger

Represents an Avenger equipped with a Quantum Suit.

Attributes:

1. `name` *(private, string)*: The Avenger's name (e.g., "Iron Man", "Thor").
2. `suit` *(private, QNS)*: The quantum suit they wear.By default suit will be assigned on First Come First Serve basis.
3. `attackStrength` *(private, int)*: Determines (battle efficiency) the amount of damage this avenger can make to an opponent.

Methods:

1. `Avenger(string avName, QNS avSuit, int strength):` For assigning the values to the attribute.
2. `void attack(Avenger& enemy)`: Damages the enemy's durability by a factor of <Avenger_attackStrength>.
3. `void upgradeSuit()`:If any extra suit is left out, transfer power to their suit in First come First serve order [can use + operator] from the extra suit and it can't be further used.
4. `void repair(int x)`: Restores some durability to their suit by *cooling down the suit* by factor x.
5. `printStatus()`: Displays the Avenger's suit details in the format "<Avenger_Name> <P> <D> <E> <H>". [Note that there are space in between]

Validation Constraints:

- An Avenger cannot fight if their suit is shut down.
- If an Avenger's suit is destroyed, they retreat(have to leave the battle) from the battle.
- If there are no suits left for Avenger X then, print "X is out of fight", and don't include X in the avenger list.
- If the avenger suit is overheated, He can cool down his suit and take part again in the battle.

---

3. Class: Battle

Simulates battles between Avengers and enemies.

Attributes:

- `heroes` *(private, vector)*: List of Avengers.
- `enemies:` *(private, vector)*: List of Enemies.
- `battleLog` *(private, vector)*: Records the battle events in a list.Only the following events have to be recorded.
    a. Consider the command “Attack A B”

       If A is able to attack (A.durability>0 and A.heatLevel<=500) and B is alive to take the attack (B.durability>0) then add “<A> attacks <B>” and after the attack if B’s durability becomes <= 0 add “<B> suit destroyed”,if B’s heatLevel becomes > 500 add “<B> suit overheated” to log.If A is not able to attack then add nothing to log. Incase if a suit is both overheated and Damaged then it will be considered as Damaged only and the avenger will be out of the battle.

    b. If Avenger is boosting his suit, add to the list “ <AvengerName> boosted”. After boosting if the heatLevel gets > 500 then add “<AvengerName> suit overheated”.
    c. If an Avenger Upgrades his suit, add to the list “<AvengerName> upgraded”.If no suit is left then add to the list “<AvengerName> upgrade Fail”.
    d. If an Avenger Repairs his suit, add to the list “<AvengerName> repaired”.

Methods:

2. `startBattle()`: Runs the battle simulation. It is the flag to start the battle.
3. `printBattleLog()`: Shows battle history. Print the battlelog element separated by lines (\n) .
4. `int Result`(): sums the Power and durability of all heroes currently in battle and similarly sums the power and durability of all the enemies currently in battle and if the sum of avenger is > sum of enemies then return 1, if the enemies are winning return -1, otherwise return 0 incase of a tie.

System Constraints

1. Each Quantum Suit must not exceed `powerLevel > 5000`.
2. Avengers cannot fight if their suit overheats.
3. If durability drops to 0, the suit is destroyed.
4. Suit assignment follows a first-come-first-serve basis.
    - **First suits are created then Avengers are assigned. Avenger cant be formed before creation of the suit.**
    - Extra suits remain unused if there are more suits than Avengers.

Input and Output Format

Input Format

1. Initialization:
    - First line: `k n m` , where k is the number of suits, `n` is the number of heroes, and `m` is the number of enemies. (heroes and enemies belong to the avenger class and k>0,n>0,m>0)
    - Next `k` lines: details of suit Qi in the format: `P D E H`.
    - Next `n + m` lines: details of avenger in format <name> <attackStrength>, assign suits to avengers in First Come First Serve Order. If no suit is left for an avenger X then print “X is out of fight” and this avenger cannot participate in battle.All the names of avengers are distinct.
2. Commands:
    - `BattleBegin`: Initializes the battle.
    - `Attack <Avenger1> <Avenger2>`: Avenger1 attacks Avenger2..
    - `Repair <AvengerName> x`: Repairs Avenger's suit by `x` durability.
    - `BoostPowerByFactor <AvengerName> y:` Boost the avenger suit by factor of y.
    - `BoostPower <AvengerName> <P> <D> <E> <H>:` Boost the avenger suit by transferring the power of the suit2(assume it as newly created suit) to Avenger’s suit .
    - `AvengerStatus <AvengerName>:` print the suit status of the avenger in the format.
    - `Upgrade <AvengerName>`: Upgrades Avenger's suit if extra suits exist.
    - `End`: Ends the simulation

- `PrintBattleLog` – Displays the battle log.
- `BattleStatus` – Prints `"heroes are winning"` if Battle.Result() ==1 ,print `"enemies are winning"`if Battle.Result() ==-1, print “tie” if Battle.Result()==0.

Sample Input: Input 1:

```
3 2 2
1000 500 300 0
1200 700 200 10
1500 800 400 20
IronMan 100
Thor 150
Loki 120
Ultron 180
BattleBegin
Attack IronMan Loki
Attack Thor Loki
Repair IronMan 50
BoostPowerByFactor Thor 2
Upgrade IronMan
AvengerStatus IronMan
AvengerStatus Thor
PrintBattleLog
BattleStatus
End
```

Expected Output 1:

```
Ultron is out of fight

IronMan 1000 550 300 0

Thor 1224 700 210 12

IronMan attacks Loki

Thor attacks Loki

IronMan repaired

Thor boosted

IronMan upgrade Fail

heroes are winning
```