MTH 4300: Algorithms, Computers and Programming II

Spring 2026

Section: SMWA

Problem Set 4

I know that it's very easy to just use AI to solve these problems, but please resist! I expect you all to give these problems an honest attempt. Spend about 30 minutes trying each problem yourself first. If you're still not getting to a good place with your solution, then prompt AI to guide you to a solution rather than expect it to solve the entire problem for you.

Problem 1: Hero's Inventory

You are building an inventory system for a role-playing game. Your task is to implement a `GameItem` class that represents items a hero can carry, equip, and upgrade.

Your solution must be split across three files:

- `game_item.h` — the class declaration (header file)
- `game_item.cpp` — the class method definitions (source file)
- `main.cpp` — the `main()` function

To compile a multi-file program, run:

```
g++ -std=c++17 -o out main.cpp game_item.cpp
```

Make sure to use **include guards** in your header file to prevent double inclusion:

```
#ifndef GAME_ITEM_H
#define GAME_ITEM_H

// class declaration here...

#endif
```

Both `game_item.cpp` and `main.cpp` should `#include "game_item.h"`.

Part 1: Class Declaration and Constructor (`game_item.h` and `game_item.cpp`)

In `game_item.h`, declare a class called `GameItem` with the following **private** member variables:

- `name` — a `std::string` representing the item's name
- `rarity` — an `int` from 1 to 5 representing how rare the item is
- `attack_power` — an `int` (must be ≥ 0) representing the item's damage
- `is_equipped` — a `bool` indicating whether the item is currently equipped

Declare the constructor and all method signatures in the header file. Then, in `game_item.cpp`, **define** the constructor. It takes three parameters: `name`, `rarity`, and `attack_power`. The constructor should:

- Set `is_equipped` to `false`
- **Clamp** `rarity` to the range 1–5 (if below 1, set to 1; if above 5, set to 5)
- **Clamp** `attack_power` to 0 if negative

Part 2: Getters and Setter (`game_item.cpp`)

In `game_item.cpp`, implement the following **const** getter methods:

- `std::string get_name() const` — returns the item's name
- `int get_rarity() const` — returns the rarity value
- `int get_attack_power() const` — returns the attack power
- `bool is_item_equipped() const` — returns whether the item is equipped

Implement a setter with validation:

- `void set_rarity(int r)` — sets `rarity` to `r` only if `r` is between 1 and 5 (inclusive). Otherwise, print:

  ```
  Error: rarity must be between 1 and 5.
  ```

Part 3: Equip, Unequip, and Upgrade (`game_item.cpp`)

Implement the following methods in `game_item.cpp`:

- `void equip()` — sets `is_equipped` to `true`
- `void unequip()` — sets `is_equipped` to `false`
- `void upgrade()` — increases `attack_power` by `rarity * 2`

Part 4: Print Info (`game_item.cpp`)

Implement a method `void print_info() const` in `game_item.cpp` that prints the item's information in this exact format:

```
--- Item Info ---
Name: Sword of Flames
Rarity: 4 (Epic)
Attack Power: 30
Equipped: Yes
```

The rarity label is determined by the rarity value:

Rarity	Label
1	Common
2	Uncommon
3	Rare
4	Epic
5	Legendary

The "Equipped" field should print `Yes` if the item is equipped, `No` otherwise.

Part 5: Main Function (`main.cpp`)

In `main.cpp`, include `"game_item.h"` and write a `main()` function that does the following, in order:

1. Create three `GameItem` objects:
   - `"Sword of Flames"` with rarity 4 and attack power 30
   - `"Wooden Shield"` with rarity 1 and attack power 5
   - `"Cursed Amulet"` with rarity 7 and attack power -10 (both values should be clamped)
2. Print info for all three items.
3. Equip the sword and upgrade it twice. Print its info again.
4. Attempt to set the shield's rarity to 0 (should print error). Then set it to 3. Print the shield's info.
5. Unequip the sword. Print its info one more time.

Expected Output

Your program should produce **exactly** this output:

```
--- Item Info ---
Name: Sword of Flames
Rarity: 4 (Epic)
Attack Power: 30
Equipped: No

--- Item Info ---
Name: Wooden Shield
Rarity: 1 (Common)
Attack Power: 5
Equipped: No

--- Item Info ---
Name: Cursed Amulet
Rarity: 5 (Legendary)
Attack Power: 0
Equipped: No

--- Item Info ---
Name: Sword of Flames
Rarity: 4 (Epic)
Attack Power: 46
Equipped: Yes

Error: rarity must be between 1 and 5.
--- Item Info ---
Name: Wooden Shield
Rarity: 3 (Rare)
Attack Power: 5
Equipped: No

--- Item Info ---
Name: Sword of Flames
Rarity: 4 (Epic)
```

```
Attack Power: 46
Equipped: No
```

Submit all three files: `game_item.h`, `game_item.cpp`, and `main.cpp`.