Dear students,

Following your first game assignment, you are now required to develop a extended version of same game. This assignment should incorporate the key object-oriented programming concepts, data structures, and algorithms covered in the course. You have the liberty to extend your game the way you want and demonstrate mastery of C++ features while delivering an engaging gameplay experience.

Sample Game Idea

1. Dungeon Crawler RPG
    - Classes: Player, Monster, Weapon, Potion
    - Data Structures: Inventory (stack/queue), enemy list (linked list)
    - Algorithms: Pathfinding (BFS/DFS), turn-based combat
    - Libraries: ncurses (text UI), Boost.Serialization (save/load)
2. Chess/Checkers
    - Classes: Board, Piece (with derived classes for each piece)
    - Algorithms: Move validation (recursion), minimax AI
    - OOP: Polymorphism for piece movements
    - Libraries: SFML (graphics), STL for board representation

2 Technical Requirements

2.1 Core OOP Concepts (Must Implement)

- Classes & Objects: All game entities must be proper classes
- Encapsulation: Proper use of private/public with getters/setters
- Constructors/Destructors: Including copy constructors where needed
- Operator Overloading: For game logic (e.g., player1 == player2)
- Inheritance & Polymorphism: Class hierarchies with virtual functions
- Abstract Classes: At least one pure virtual function interface

2.2 Data Structures & Algorithms (Choose at least 3)

- Dynamic Arrays/Linked Lists: For inventories, enemy lists, etc.
- Stack/Queue: For undo/redo, pathfinding, or turn management

- Recursion: For puzzles, maze generation, or backtracking
- Sorting Algorithms: For leaderboards or item organization
- Priority Queues: For event scheduling or damage systems

2.3 Advanced C++ Features (Choose at least 2)

- Templates: Generic containers or functions
- Exception Handling: Robust input validation and error recovery
- STL Containers: Proper use of vector, map, etc.
- Lambdas: For event callbacks or AI behavior
- Friend Classes: For controlled access between classes

3 Course Topic Applications

Topic	Game Implementation Example
Classes/Objects	All game entities (Player, Enemy, Item) as classes
Encapsulation	Private member variables with public getters/setters
Operator Overloading	operator+ for combining items, operator« for output
Inheritance	Enemy → FlyingEnemy, GroundEnemy hierarchy
Abstract Classes	Renderable interface with render() pure virtual function
Templates	Inventory<Item> container class
STL Containers	std::vector for bullets, std::map for game settings
Recursion	Maze generation using recursive division
Sorting	MergeSort for high score tables
Exception Handling	Invalid move exceptions in board games

4 Recommended Libraries

- Graphics: SFML, SDL2, raylib, OpenGL
- Text UI: ncurses (pdcurses for Windows)
- Serialization: Boost.Serialization, JSON for Modern C++
- Utility: STL, Boost (for smart pointers, etc.)

Submission Requirements

- Complete source code (properly structured .h/.cpp files)
- Makefile/CMakeLists.txt for building

- Documentation (README file)
- 5 page report explaining design decisions
- Demonstration (executable or video)

********** Best Of Luck**********