AYBÜ - CENG 209 - Fall 2024 - Homework
Due December 17th 2024, 23:59
Late submission: -10% per day

Homework Assignment (25%)

Objective:

Develop a command-line-based adventure game in C that simulates a dungeon exploration scenario.Implement command parsing, input handling, file I/O, and basic game logic, while learning to create a user-friendly text-based interface in a shell environment.

Game Description:

The game begins with the player starting in a dungeon room. Players navigate through various rooms, collect items, and encounter creatures using shell commands. Commands should be typed at a custom shell prompt and include actions like `move`, `look`, `inventory`, and `attack`.

Assignment Requirements:

1. **Basic Game Structure:**
    - **Player Stats**: The player has health, strength, and inventory capacity, all starting with default values.
    - **Rooms and Navigation**: Each room can have connections to other rooms (up, down, left, right), and each room is uniquely described with text.
    - **In-Game Commands**:
        - `move <direction>`: Move to a different room if possible.
        - `look`: Display the description of the current room and items or creatures in it.
        - `inventory`: List items the player has collected.
        - `pickup <item>`: Add an item to the inventory if present in the room.
        - `attack`: Fight a creature if present in the room.
    - **Menu Commands:**
        - `list` : Lists the saved games
        - `save <filepath>` : saves the current state in to the given file
        - `load <filepath>` : loads the state from the given file
        - `exit`
2. **Detailed Functional Requirements:**
    - Implement game commands as functions within the shell.
    - Use parsing to interpret commands.
    - Implement health and battle mechanics: a simple turn-based system where the player can attack a creature until one side is defeated.

- Inventory management: Limit the inventory size and allow the player to manage it.
- Implement file-based data storage for room descriptions, item attributes, and creature stats.
- Use Manage inventory and room descriptions with dynamically allocated memory. For example, each item and room description can be stored as dynamically allocated strings that students must free when no longer in use.
- Define struct types for game entities like Player, Room, and Creature, each with properties (e.g., health, items, room connections). Encourage typedefs to improve code readability, especially with function pointers for game actions.
- Implement robust error handling, especially for file I/O, memory allocation, and game commands (e.g., handling invalid commands, room navigation boundaries, and inventory capacity).

3. Supplementary documents
   - Create a Google doc report https://docs.google.com/ explaining how your game works, and providing an overview of your application design and implementation. Turn sharing on in your Google doc in "**anyone with link**" mode.
   - Record a demonstration video explaining your application. Your video should NOT exceed 10 minutes. The video quality should be good enough to see what you are showing on your computer screen. Your voice should be clear and audible during your demonstration. Upload your video to YouTube as an **unlisted** video.
   - Create a **public** GitHub repository on https://github.com/ and upload your project files to your GitHub repository including your
     - Source code files (.c and .h)
     - A `README.md` file explaining gameplay, code structure, and game logic.
     - A `Makefile` for compilation.
4. **Submission Instructions:**
   - Fill out the following form https://forms.gle/2MrJvcm97a1CN15C7 and provide the following:
     - Your public GitHub repository URL.
     - Your unlisted YouTube video URL.
     - Your Google doc report URL.
   - Submission is due **December 17, 2024, 23:59.**
   - Late submission: -10% per day
5. **Academic Integrity:** While creativity is encouraged, ensure the code and gameplay logic are your own.

Grading Criteria (Total: 100 points):

- **Game:**
  - **Core Functionality (15 points)**

        - Game command handling and parsing
        - Room navigation and descriptions
        - Inventory and item management
        - Combat mechanics
    - **Game Logic and Design (15 points)**
        - Thoughtful game balance and mechanics
        - Clear and engaging descriptions
        - Effective memory and file handling
    - **Creativity and Additional Features (15 points)**
        - Random encounters, traps, or special items
        - Save/load functionality or other unique ideas
- **Documentation (10 points)**
    - Detailed `README.md` for gameplay instructions and code explanation (5 points)
    - `Makefile` for easy compilation (5 points)
- **Source code quality (15 points)**
- **Audio/Video/Demo quality (15 points)**
- **Report quality (15 points)**