Assessment 1 - Implement a Zork Clone.

- The game should play in a normal windows terminal, using normal, line based, terminal I/O.
- The game should feature at least 10 ‘unique’ commands.
- The player needs some kind of internal ‘location’.
- The player needs some kind of internal ‘state’. (i.e inventory, or something more advanced).
- The game should contain some kind of ‘minigame’. Everything from ‘guess-the-word’ to ‘tic-tac-toe’. Does not have to be complicated, but it should put you outside of the ‘normal game-flow’.
- The game should have a command ‘LOG’. This command should lists a log of all commands that have been entered, during the current play session.
- The game should have a command ‘COMMANDS’. This command should list all possible commands that the user can type, along with their description.
- The game should have a command ‘HELP’. This command should give helpful information to the player.
- The game should have a command ‘QUIT’. This command should quit the game.
- The game should have a playtime of ~5 minutes.
- The game should not crash, ever, given normal ASCII input. Invalid commands should give a reasonable ‘error’ message.
- Input should be somewhat case-insensitive. That means that help & HELP should both work. Ideally you should also make hElp work too.
- Since the lowest minimum requirements are low, you must scope additions based on your ability. Ideally build a MVP that satisfies the requirements, before building more involved things.
- No global variables, with the possible exception of the internal variables that underlies the ‘LOG’ command. These must be interacted with via helper functions, not directly.
- Ideally, build a ***fun game & have some fun too.***
- No classes created by you, only structs with pure data.
- You are free & recommended to talk and discuss solutions with classmates, as long as you can explain every line of code written.
- Avoid searching the internet for solutions.

Assessment 1 - Zork Clone - Extra

- Make commands completely verb based.
- Make some commands have parameters. ‘Pickup ***Item***’.
- Implement an ‘UNDO’ command, which rewinds the game to the state of the previous command. This can be done in several ways. Using the log, using a stack, using the ‘command pattern’ (you may use the internet for this one). All of these are very useful concepts to learn at some point.
- Implement a ‘DROP’ command. This creates the need for a somewhat more involved ‘world state’.
- Make a ‘test suite’ for the game.
- Make an AI that can play the game.