Software Engineering Project 1

Focus Board Game

https://github.com/ash-xyz/Focus
Ashraf Ali - 19315281

Disclaimer

This is a **Unix** based board game which means I've only configured it to run on Mac and Linux, the lecturer said this was alright.
Also make sure your terminal is in Fullscreen mode!
The README includes installation instructions for Mac and Linux.

Evaluation Criteria

1. **The code is well commented and appropriately divided into modules (10%)**
    - Added detailed comments and separated modules to keep files consistent with their purpose.
2. **The code committed often in the repository. Something I really don't want to see is the code committed at the last minute in the repository (10%)**
    - Code was committed after relevant change, 80+ commits.
3. **The data structures adopted to represent game entities (players, board, squares) are appropriate, i.e. represent the required information (10%)**
    - Added struct Player to hold player info
    - Added struct square to hold pieces(using a linked list) and their heights.
    - Added struct Game to hold the board(composed of squares) and players
    - Added struct gameState to hold relevant information about the current state of the game such as the current player turn.
4. **The game logic is correct (70%). This means that the following aspects are implemented correctly**
    - **The game board and the players are initialized correctly. (10%)**
        i. Used init_board() function provided to us to initialise the board with red and green pieces.
        ii. Used init_player() function to take in player info, including prompting users for their names using the promptName() function
    - **The board and information about the current player and the winning player are printed correctly. (10%)**
        i. The board is printed using an ncurses window through the function drawBoard() and drawStack()
        ii. The winning player displays all relevant info about the winning player through the function drawWinner()
    - **The game allows a player to move his own pieces or only a stack that has a top piece of the same colour of the player. (5%)**

    i. checkOwnedPlayer() function checks if the selected piece is owned by the player.
- **The game allows to move a piece/stack of a number of positions (up, down, left or right) corresponding to the size of the stack. (8%)**
    i. validMove() function checks if the move a player wants to do is valid and within bound.
    ii. movePieces() is then called if validMove() is true, it
- **The game does not allow a stack or piece to be moved outside the size of the board and in the angle squares ({(0,0), (0,1), (1,0), (0,5), (0,6), (1,6), (5,0), (6,0), (6,1), (5,6), (6,5), (6,6)}) that cannot contain any piece or stack. (2%)**
    i. The game functions on the keyboard's arrow keys, four functions called checkValidUp,Left,Right,Down() makes sure you can't go to invalid squares.
- **A piece/stack can be placed correctly on an empty square (2%)**
    i. This is accounted for by the movePieces() function
- **A piece/stack can be placed correctly on another stack (8%)**
    i. This is accounted for by the movePieces() function
- **Pieces are removed correctly from the bottom of the stack to keep its size equal to 5. (5%)**
    i. trimStack() ensures that a stack remains at a maximum of 5 pieces
- **A player can accumulate his/her pieces correctly and place reserved pieces on the board if desired (10%)**
    i. trimStack() retains a player's own pieces and places them in a 'graveyard' which counts how many pieces a player can place.
    ii. resurrectPieces() is triggered upon pressing 'g'
        1. It first checks if a player has any pieces to resurrect
        2. Generates and places a piece, of the players colour, on top of the selected stack
        3. Triggers trimStack() if the newly created stack exceeds 5 pieces
- **The stack is implemented using a linked list (10%).**
    i. The stack operates by using the piece_node struct which links piece_node to piece_node using piece_node *next, effectively acting as a linked list.