Daniel O'Connell- 19448622

Evaluation Criteria

- The code is well commented and appropriately divided into modules (10%)
    - I tried to comment regularly and comprehensively while dividing the project into manageable comprehensive modules; input/output, initialisation and turn taking
- The code committed often in the repository. (10%)
    - I committed every time I advanced the project.
- The data structures adopted to represent game entities (players, board, squares) are appropriate, i.e. represent the required information (10%)
    - I implemented the players, board and squares according to the specifications of the assignment

- 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 initialised the players asking for names and assigning attributes, then set each square on the board either valid or invalid, and finally set the correct squares red and green
    - The board and information about the current player and the winning player are printed correctly. (10%)
        - I used an unsigned int to keep track of which player's turn it was, and from there the remainder of count/2 showed the correct current player and at the end the correct winning player also.
    - 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 implemented a function for moving stacks and setting the square moved from as empty, as well as a function to choose the stack which implemented restrictions eg can only move a stack of the same colour
    - 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%)
        - In the choose stack function I checked to make sure the absolute value of the changes in number of squares moved across and up/down was equal to the size of the stack.
    - 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 had a check in the choose stack function to check if the square to be moved to was valid, and if not then get correct input
    - A piece/stack can be placed correctly on an empty square (2%)

-i made a place piece function which checks if the square is empty and valid, sets the size of the stack to one and sets its colour to that of the player

- A piece/stack can be placed correctly on another stack (8%)
    - In the move stack function, the top piece is set to the moved stack's top piece, and the pointer of the moved stacks last piece is set to the other stacks top piece. The top of the other stack is then set to the new top, the moved stack's old square is set empty and the excess elements are freed, with captured pieces added to the appropriate counts
- Pieces are removed correctly from the bottom of the stack to keep its size equal to 5. (5%)
    - see above
- A player can accumulate his/her pieces correctly and place reserved pieces on the board if desired (10%)
    - When a player captures pieces the excess elements are freed, with captured pieces added to the appropriate counts. I added a place piece functionality so if they have the option they can place a piece on the board
- The stack is implemented using a linked list (10%).
    - I used a linked list for the stack with the first element pointing to the top piece of the stack

Link:
https://github.com/daniel-oconnell/boardgame_focus/settings/access