Tom Higgins – 19343176
Focus Board Game Project Description

Link to repository: https://github.com/tomh00/Focus.git

1. Initialisation of game
Step 1 involved the initialisation of the key components of the board game. This section required understanding of the programming tools structures and enums in order to complete. We had to use struct notation "." and "→" when using pointers to structs in order to complete initialisation of the players and the board.

2. Implementing the main functionality of the game
The game involves moving pieces around on the board and stacking pieces on top of one another. For this reason the concept of stacks were used in the project. Implementing this logic meant a need for good understanding of linked lists and stacks. Each turn, we needed to decide which player would go. In order to achieve this I used a boolean array with 2 elements representing each player. The contents would say true if the player was last to move and false if the player was not last to move and this was adjusted with each turn.

Each square of the board (each a struct) had a pointer to the top piece of that square. So when moving pieces we navigated to the bottom of the stack at the given square and set the next pointer of that piece to the top piece pointer of the other square. We also had to make sure to adjust the number of pieces on each square just represented by an integer.

3. Stack size maintenance
The rules of the game say that stacks cannot be greater than 5 in size. Part 3 of the project required us to implement control of this. Each player struct had a pieces kept integer and pieces captured for removing excess pieces from the bottom of the stack when it became larger than 5. We needed to use the free function to free memory for this part as each piece struct was memory created using malloc.

4. Winning condition
The winning condition is that when a player cannot take a turn as they have no pieces on top of any stacks and also no pieces to move. I used two functions to check these two conditions and an if statement which would determine if either the player has a top piece on the board or has a kept piece they can place. If either of these were true the program would proceed to the turn functionality but if neither were true the game was over and the turn loop was broken from.

I then used the playerStatus function and nextTurn function which were used throughout the game to print the winning player's details.