Focus/Domination board game project description

Initialization:

The game starts by initializing the board and players. The board is set to its starting position and all valid squares are set to red, green or empty using different functions. This code was provided at the start so too much documentation is unnecessary.

Players store their data in structs, which are initialized in the playerInit() function. The function asks for both players names and asks player 1 to choose a colour. Player 2 is given the other colour. When the colour is taken, strcspn() is used to remove the newline so it can be compared to the strings "red" or "blue" for validity.

Board and Score Printing:

The board and score printing function is stored in GameIO.c. The board coordinates are printed so they correspond to each square. The printBoard() function checks every square on the board. Invalid squares are blank. Empty squares are printed with ++ so they can be distinguished from invalid squares. Squares with pieces show the colour of the top piece (G or R), as well as the number of pieces on that square.

After the board is printed, player information is printed. The name of each player is printed, with their colour, captured and reserved counts below their names.

Taking Turns:

A loop runs until a winner is found. To check for this, at the start of each turn the checkboard() function checks every square to see if a player can make a move. If they can't, they are checked for reserve pieces. If they have none, the game ends and the other player wins.

When moving a stack or placing a reserve piece, the getCoordinates() function is called. getCoordinates() verifies that the coordinates input is valid, and that the coordinates chosen are valid squares on the board. The function is general use, so some further verification is required when moving a stack. The stack is checked to ensure that it contains a piece that the player owns.

After choosing a stack, the player must input one direction from u (up), d (down), r (right), l (left) in order to move. The directions are verified to ensure that the player does not go off the board.

Moving/Removing Pieces:

addToStack() is used to move a piece. It places the pieces in the moved stack on top of the pieces in the target stack, and nullifies the pointer to the other stack. The piece counters are updated appropriately.

addReserve() is used to add reserve pieces. It works similarly to addToStack(), but instead of taking a stack from another square, it allocates memory for a new piece and adds it to the top of the stack.

popOne() removes one piece from the bottom of the stack. It frees the piece that is removed. Every turn, when a move is made, the target stack is checked to see if it has exceeded 5 pieces. If it has, the function is called in a loop, which iterates once for each extra piece.

End of the Game:

The game ends when one player cannot move a piece. The score is printed again, as well as a message saying which player won.

After the game has ended, all dynamically allocated memory is freed.