Questions

**A:** To implement this you would have to create a class that specifically handles this requirement. To perform the actual moves the class will require a pointer to the piece that is necessary for the move. The information for the moves will have to specify what piece is involved and the required destination for that particular piece. The list of moves would be read from *cin* or from a file at the start of the program and then it will class our board’s methods to make the appropriate move. Every time an opponent makes a move the program assesses if the sequence should continue and then update the board accordingly.

**A:** A stack data structure would be a perfect fit for this problem. Each time a player’s piece is moved a copy of that piece can be added to the stack, including its previous position on the board. To implement the undo feature the player may call the *undo* method and essentially calling it will pop the last moved piece off the stack causing the board to go back to a previous state.

**A:** Implementing a four-handed chess game will require two more players and an increase in board size. There will be a total of six more columns, three on each side of the board. The increase in two more players also adds more pieces to the game, a total of sixty-four pieces, and adds two more colours into the program. Since the rules of this game is no different than the traditional one, the game logic will not change. However, assessing if a player’s move is valid needs to accommodate the other three players because their pieces are targets also. Furthermore, if the players are partners ensuring the capture of only enemy pieces is crucial, to indicate which players are partners a field *partner* is necessary in the *Player* class.