Plan of Attack

1. Determine priority of program hard/soft requirements
   * Complete by Nov. 15
2. Determine and list basic/possible test scenarios before design and implementation
   * Complete by Nov. 16
3. Design UML class model
   * Complete by Nov. 18
4. Build/implement program, implantation priorities are determined from before
   * Complete 80 - 90% by Nov. 30
5. Create design document
   * Complete by Dec. 4

(since I’m working alone on this project, I did not list partner responsibilities)

Program Priority List:

* Basic program functionality: Command Interpreter (Player vs. Player), Text Display, Scoring System (depending on difficulty, might exclude castling, en passant, and pawn promotion)
* Graphic Display, Player vs. Computer (level 1), castling, en passant, and pawn promotion if not yet implemented
* Player vs. Computer (level 2-3)
* Player vs. Computer (level 4)
* Undo (unlimited) move function

Q1 Answer: To implement a book of standard openings, I think a database of the starting moves and the corresponding responses is needed. As with the explorer of chess.com, the database would need to store data of (potentially) all possible starting moves and responses to opponent’s move, along with the corresponding win/draw/loss percentages. My idea is to store string data of each move, such as “W1 P E4” where “W1” represents the first move from white-player, “P” is the name of the chess piece moved (Pawn is this case), and “E4” is the final position of the chess piece moved. We would also have data like “B1 P C5” and “W2 N F3” where “B1” represents the first move from black-player, “W2” represents the second move from white-player, “N” represents “Knight”, and so on. We would then need to associate each move with the corresponding win/draw/loss percentages. I assume chess.com collects data of each chess match on their platform to calculate these percentages, but we could also find books that suggest standard openings and responses based on previous moves. This is of course, much less scientific than percentages, but we could associate each move with a recommended next moves. In terms of implementation, we could associate each move with a priority data structure of next moves (where the priorities would be recommendations from the book).

Q2 Answer: To implement an undo last move feature, my idea is to have a tracker of each turn of the game. For each turn, the tracker would store data of which chess piece moved, the starting and end coordinates of that chess piece, and if any chess pieces were removed from that turn. We could implement the tracker as a stack of the data of each turn, so that when we need to undo a move, we pop off the data from the stack and “undo” the move from that turn (an “undo” function would be needed). My only concern is that this might not scale well, since as the game goes on, more turns and thus more non-constant data will be stored in the stack.

Q3 Answer: To implement four-handed chess, we would need to change the size of the board, the number of chess pieces, associate a different chess colour for each of the four players, and most importantly, a point system for capturing each chess piece would need to be implemented. This of course implies that each player would have a corresponding scoring system. If a player resigns, points may be added to a player depending on if there is only one player left. I assume that most of the chess piece functionality would remain the same, except for maybe checking multiple Kings with the same move.