Chess: Plan of Attack Documentation

Inqiyad Patwary, Mahzabin Rashid Fariha, Sadman Ahmed

Our plan of attack is to build the chessboard, players, and pieces. We ensure that the text-based board is displayed and the pieces are in their correct initial positions.

Inqiyad will build the players and the board, Sadman will work on creating the pieces and the moves each piece takes, and Fariha will work on implementing the observer design pattern and creating the text-based display, and testing its functionality. We will then merge all of our works to get a functioning chessboard where individual players can make moves and win, lose or draw through checkmate, stalemate, or resign.

After we confirm that our chess game is functioning, we will split the work again. Fariha will work to make the graphics display, Inqiyad will work on the difficulty levels for the computer, and Sadman will work on adding the additional chess functionalities such as castling, en passant, undo and pawn promotion.

We plan on completing our work by 11th December. This will allow us enough time to test, debug and make necessary changes to our chess game before the final deadline on 15th December.

Question 1.
Chess programs usually come with a book of standard opening move sequences, which list accepted opening moves and responses to opponents' moves, for the first dozen or so moves of the game. Although you are not required to support this, discuss how you would implement a book of standard openings if required.

We can use a map structure to store a series of opening moves that can be implemented. The opening moves will be immutable and the computer will iterate through it to decide the best move possible. We will also make a function that checks the respective response that the opponent can take from our map structure for the given move and then implement that. To check the next move, we will use the values of the moves stored in the map structure in our conditionality statements.

Question 2.
How would you implement a feature that would allow a player to undo their last move? What about an unlimited number of undos?

We will use a vector to store move history. This will keep track of whichever piece is moved where on the board. Each index will contain the previous coordinates (row,

column) of the piece and its name. If a player wants to undo their last move, the first piece from our vector will go to its previous coordinates again and the first move history entry is popped off our vector. This will work for an unlimited number of undos since we store move history from the beginning of our game.

Question 3.
Variations on chess abound. For example, four-handed chess is a variant that is played by four players (search for it!). Outline the changes that would be necessary to make your program into a four-handed chess game.

We will first need to change the display of our board to accommodate the addition of two additional players. We will need two new sides and colors and also allow them to make moves. We will allow players to choose whether they want to play as a team or as a free-for-all. The conditions which lead to check, checkmate, and the game-ending will change accordingly, depending on which format the players choose to play.