Plan of Attack

To start the project, we decided to start with a skeleton structure of the project which includes all necessary classes with its variables and methods declared (not implemented yet). The skeleton should involve properly setting up the board, the chess pieces and the players. We should have a basic program running at first, with the human Vs human mode. Once the program is running, we can add the AI component and extra features. We should also have the text display class implemented early on for testing purposes. To avoid causing confusion between teamembers, we decided to split the project into three components as they are fairly independent.

David will be responsible for the first part. The first component involves setting up the Grid class, the Chess class and its seven subclasses. The Grid class has a field which contains a vector of vector of pointers to the Chess class (2D array) and acts as the Subject for text display and graphics display. Since the Grid class oversees the entire board, it is responsible for moving a piece, setting a piece, detecting check/checkmate, castling, and *en peasant*, etc. The Chess class is an abstract class with seven subclasses indicating each piece (Bishop, Queen, King, Knight, Rook, Pawn, and the empty piece). It stores the piece's position, type, and what colour it belongs to. Each chess piece is responsible for determining if a move is valid based on its piece type.

The second part consists of the implementations of text display and graphics display. These two classes will act as observers to the Grid class (i.e. every time the Grid is updated, they will get notified and update its own fields). This part will be completed by Sarah.

Joyce will be implementing the third part of the project, which contains the two Player classes. Player will contain information regarding its level, its colour and keeps track of its scores. In addition, The Player class will have a move method which in turn calls the Grid's move method.

We plan to have these three components working by Friday, July 26th. By then we should have a working program with the basic features. Then Sarah would work on implementing the functions for the AI component (level 1 – 3), where the other two teamembers will test the program and ensure it runs smoothly without crashing. We hope to fully implement and test the program by Sunday, July 28th. Then if time permits, we would add the bonus features, preferably the undo feature and the command which prints a list of past moves.

Question: 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.

A: In order to implement a book of standard opening move sequences, we could use a map that maps the current state of the Grid to a suggested move. Thus for the first dozen moves, when a computer plays, it would take in the current state of the Grid and tries to find a corresponding key in the map. If it does, it would make a move according to the suggested move, otherwise, it would move on to determining its own moves.

Question: How would you implement a feature that would allow a player to undo his/her last move? What about an unlimited number of undos?

A: In order to allow for a player to undo his/her last move, we need to have two more fields in the Grid class. Each of these fields will contain a structure called BlackLastMove or WhiteLastMove respectively for Black and White players. The structure contains the information regarding a player's last move, including the old position, the new position, the piece eaten (if there is a piece eaten) and the type of the piece moved. Each time an undo is called, we call the move function with the information stored in the structure field, but with the new and old positions swapped. Also, every time a move is played, we update the field respectively depending on the player. To allow for an unlimited number of undos, we would instead use a stack to store player's last moves. Each time an undo is called, two moves are popped off the stack (your move and your opponent's move). Otherwise, the strategy used for one undo can be used repeatedly.

Question: 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.

A: First, we would need to add two more players and two set of chess pieces, so that in the initialization stage when the game starts, four players of different colours with their pieces will be set up. Also, the Grid has to expand from a 8x8 2D array to a 14x14 one, where the 3x3 squares at the four corners are considered to be out of bounds (i.e. invalid move spots). Also, when checking if there's a winner, we need to make sure that only one player is remaining and eliminates players as the game progresses. At the start of the game, players should be able to choose if they want to play in teams. If the game is played by two teams, we have to have an extra field in the Player class which indicates the colour of its alliance. Also, we need to have more restrictions on the move function as now a play can only attack pieces that have the colour of its enemies.