CS246-FINAL PROJECT

Plan of Attack

July 15, 2016

Prepared by:
Amogh Nayak
Mohammad Hyder
Saad Husain

Timeline

Task	**Estimated Complete Date**	**Ownership**	**Status**
Completing PoA	**15-Jul**	**Everyone**	Done
Meeting Team and PoA	12-Jul	Everyone	Done
Design First draft of UML	12-Jul	Everyone	Done
Completing Draft PoA	14-Jul	Everyone	Done
Completing Final PoA	15-Jul	Everyone	Done
Creating Piece Class	**15-Jul**	**Everyone**	**In progress**
King and Queen	15-Jul	Amogh	In progress
Knight and Bishop	15-Jul	Mo	In progress
Rook and Pawn	15-Jul	Saad	In progress
Creating Board class	**16-Jul**	**Everyone**	**Not Started**
ischeck()	16-Jul	Amogh	Not Started
ischeckMate()	16-Jul	Mo	Not Started
ischeckStaleMate()	16-Jul	Saad	Not Started
Design Game Logic	**17-Jul**	**Mo**	**In progress**
main.cc	16-Jul	Mo	In progress
Resign, Move, Setup, and other game options	17-Jul	Mo	Not Started
Calls to appropriate methods	17-Jul	Mo	Not Started
Implementing View Class	**18-Jul**	**Amogh**	Not Started
display board after initial configuration	18-Jul	Amogh	Not Started
Creating Board/View Basic Text	**18-Jul**	**Saad**	**Not Started**
Creating player and different levels	**19-Jul**	**Everyone**	Not Started
Human	19-Jul	Amogh	Not Started
Level 1	19-Jul	Mo	Not Started
Level 2	19-Jul	Saad	Not Started
Add Graphical Display	**20-Jul**	**Amogh**	**Not Started**
Design Document and Final UML	**21-Jul**	**Everyone**	**In progress**
Makeup day (Testing and Fixing Bugs)	**21-Jul**	**Everyone**	**Not Started**
All bonus features listed in the assignment	**21-Jul**	**Everyone**	**Not Started**

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.