# User Manual

Minor Carlson

Version 1.0

Team 21:
Anthony Flores
Chenyu Wang
Davis Nguyen
Nathan Bae
Timothy Chan

# Table of Contents



 

# Glossary

## Chess Pieces:

**Pawn:** The Pawn can only move forward. It can only attack diagonally. On its first move it can either move forward once or twice. Once it reaches the completely other side of the board it is able to transform into either a Bishop, Knight, Queen, or Rook.

**Rook:** The Rook can move vertically or horizontally for as many spaces it wants given there isn't any piece in the way. It can attack any piece horizontal or vertical.

**Knight:** The Knight can move in L shape so either two moves vertically and one move horizontally or two moves horizontally and one move vertically. It is able to jump over pieces. It can attack any piece it lands on.

**Bishop:** The Bishop can move diagonally as many spaces as it wants. It can not jump over pieces and it attacks diagonally as well.

**Queen:** The Queen can move diagonally, vertically, or horizontally as many spaces as it wants. It can not jump over pieces and it attacks diagonally, vertically, or horizontally.

**King:** The King can move diagonally, vertically, or horizontally but for only one space. It can not jump over pieces and it attacks diagonally, vertically, or horizontally.

## Chess Moves and Outcomes:

**Castling:** Is a special move where the Ring and Rook simultaneously move towards each other. The king moves two squares towards the rook and the rook moves right next to the king on the opposite side. Castling can only be done when the King and Rook have yet to move and there are no pieces between them.

**Enpassant:** this special move involves the pawns. This move consists of a pawn being able to capture a pawn right after the pawn has taken two moves to avoid capture. The capture must be done right after the pawn's two space evasion. The capturing pawn can diagonally attack the opposing pawn at the square right behind the opposing pawn.

**Check:** When the king is being threatened it is called check. When an opposing piece has the opportunity to capture the King in the next turn if no preventative action is taken this is called

“Check”. Check forces the threatened player to take action to prevent his king from being captured in the next turn.

**Checkmate:** When in Check the defending player has no protective action available to save their King.

**Draws:**

**Stalemate:** This occurs when a player has no possible moves and is not in Check.
Impossibility of Checkmate: This occurs when there aren't enough pieces in each of the players to produce a Checkmate.
**75-move-rule:** If no pawn moves or capture occurs within 75 moves, it is an automatic draw.
**Draw agreement:** Players are allowed to agree on a draw at any point in the game.

# Computer Chess

## 1.1 Usage scenario

```
Welcome to Minor Carlson Chess!
Please select a gamemode (1 for Player vs Player, 2 for Player vs AI): 1
Please select side for Player 1(White or Black): White
White side has been selected for Player 1.
The game will now initialize.

  +----+----+----+----+----+----+----+----+
8 | bR | bN | bB | bQ | bK | bB | bN | bR |
  +----+----+----+----+----+----+----+----+
7 | bP | bP | bP | bP | bP | bP | bP | bP |
  +----+----+----+----+----+----+----+----+
6 |    |    |    |    |    |    |    |    |
  +----+----+----+----+----+----+----+----+
5 |    |    |    |    |    |    |    |    |
  +----+----+----+----+----+----+----+----+
4 |    |    |    |    |    |    |    |    |
  +----+----+----+----+----+----+----+----+
3 |    |    |    |    |    |    |    |    |
  +----+----+----+----+----+----+----+----+
2 | wP | wP | wP | wP | wP | wP | wP | wP |
  +----+----+----+----+----+----+----+----+
1 | wR | wN | wB | wQ | wK | wB | wN | wR |
  +----+----+----+----+----+----+----+----+
    a    b    c    d    e    f    g    h

Player 1, input a move: E2E4

  +----+----+----+----+----+----+----+----+
8 | bR | bN | bB | bQ | bK | bB | bN | bR |
  +----+----+----+----+----+----+----+----+
7 | bP | bP | bP | bP | bP | bP | bP | bP |
  +----+----+----+----+----+----+----+----+
6 |    |    |    |    |    |    |    |    |
  +----+----+----+----+----+----+----+----+
5 |    |    |    |    |    |    |    |    |
  +----+----+----+----+----+----+----+----+
4 |    |    |    |    | wP |    |    |    |
  +----+----+----+----+----+----+----+----+
3 |    |    |    |    |    |    |    |    |
  +----+----+----+----+----+----+----+----+
2 | wP | wP | wP | wP |    | wP | wP | wP |
  +----+----+----+----+----+----+----+----+
1 | wR | wN | wB | wQ | wK | wB | wN | wR |
  +----+----+----+----+----+----+----+----+
    a    b    c    d    e    f    g    h

Player 1's pawn has moved from e2 to e4.
It is now Player 2's turn.
...
(Game will continue until one of players make illegal move or checkmate)
Player 1 (White) wins!
Press 1 to go back to menu or 2 to rematch: 2
Game will restart...
```

For our program, the normal usage scenario involves the user loading our program up which leads to the main menu with options for the user to select. After a selection is made then the user will be given a choice to choose which color the player would like to be. Then a board with pieces will be displayed and white goes first. After each move is made a log is generated and an updated board will be created. Moves are exchanged until a player has a checkmate then the game ends. The player gets to choose to play again or return to the menu to select another mode or exit the program.

## 1.2 Goals

Our main goal for this project is to create a fully functional chess program that includes the required basic functions and integrating the advanced options that are desirable if time permits. This will incorporate a multitude of software engineering concepts. These include documenting the program itself, designing the program, writing the code, and debugging the code. Additionally, this project will provide a way to work as a team with other group members. Students will need to successfully cooperate with one another in order to complete this project.

## 1.3 Features

The most basic features that will be included in this program include:

- Follows the official rules of chess
- User interface where the use can see the board and type in where to move
- Allows user to decide which side to play (white or black) and to see the board
- Supports human player vs player or human player vs computer
- Provides a log file of all actions after the game is finished

The advanced feature that will be included in this program include:

- Choosing difficulty of AI
- Computer vs. Computer gamemode
- Timer on the board to see remaining time left for turn

# Installation

## 2.1 System requirements

A functional computer (Either Windows or Mac) that is capable of running Linux. The machine can run this program in both a 32-bit and 64-bit operating system. This program

requires little processing power to function. Additionally, an insignificant amount of hard drive space is required.

## 2.2 Setup and configuration

For the setup of the chess program, simply download the provided file and choose the location of where you want the program to be placed. Then simply run the executable file in order to open up the program.

## 2.3 Uninstalling

To uninstall the program, please enter the following command into the terminal “rm filename”.

# Chess Program Functions and Features

## 3.1 function main()

This function:
1)sets up the board;
2)initializes the game;
3)creates a log file for the game;
4)lets the human player choose the sides;
5)runs the loop until the game ends if one player wins or an illegal move occurs.

## 3.2 function move()

input: location_src, location_dest example: ‘A1, A5’
Output: none
This function uses the input information to move a certain piece to a certain location, if there’s capture involved, the capture will be automatically done and recorded in the log file

## 3.3 function computer_move()

Input:none
Output:  an instance of struct move
Invoked in main() when it's the computer's turn to move.

 

## 3.4 function isLegal()

User input: none
Output: print out the error if the move is illegal
This function takes the input data of the function move and evaluates whether it's a legal move. It'll print out information if it's not legal and end the game using endgame() function

## 3.5 function draw()

Input: struct board
Output: print out the board in the user interface
This function uses the board information to print out the board in the user interface

## 3.6 function endgame()

Input: none
Output: announcement about the result of the game
This function will be invoked in main() directly or in isLegal() if an illegal move occurs.

## 3.7 function findAllMoves()

Input: the board and the current player
Output: a list of possible moves

## 3.8 function inCheck()

Input: board with current status
Return value: True if the board is in check, false otherwise
Output: Prints out warning to computer or to player if previous move resulted in a check
This function takes the input data of the previous move and evaluates whether that move resulted in a check. It will print a warning on the screen indicating that a check has happened and you must resolve the check

## 3.9 function recordMove()

Input: the last move that has been made and the player who made it
Output:none
This function records the move into the log file

# Back matter

## • Copyright



## • Error messages

Illegal Moves

- “There is no piece at selected location”
- “That piece cannot move to selected position”
- “That piece is not yours”

## • Index



## • References

http://www.wachusettchess.org/ChessGlossary.pdf
https://cdn.shptrn.com/media/mfg/1725/media_document/8976/Imp_ChessR.pdf?1401227342