Course Project: Chess

1 Game Description

Chess is a two-player board game played on a chessboard, a checkered game board (usually black and white) with 64 squares arranged in an eight-by-eight grid. Each player begins the game with 16 pieces: one king, one queen, two rooks, two knights, two bishops, and eight pawns. Each of the six piece types moves differently. The most powerful piece is the queen and the least powerful piece is the pawn. The objective is to checkmate the opponent's king by placing it under an inescapable threat of capture. Read the detailed explanation of the pieces' movements and other rules that you are required to implement (See the Required Features section) carefully on the Wikipedia page here.

2 Required Features

Two players

Human vs. Human.

User Interface

- You should represent black and white squares using different ASCII characters, e.g. "." for black squares and "-" for white squares.
- White pieces are represented by "p","r","n","b","q","k" for pawns, rooks, knights, bishops, queen and king respectively. Black pieces are represented by the same letters but capitalized.
- You should display next to the game board the pieces that are currently taken out from both players.
- You should display the characters from 'A' to 'H' to denote the columns above and below the board, along with the numbers from 1 to 8 to denote the rows to the right and left of the board.

Rules

- Your program should support promotion and stalemate.
- The game is won by checkmate.
- The game may end in a draw by stalemate.

Movement

As shown in the Wikipedia page, each square is indexed by a character and a number (e.g. D1,C3,...etc.). The user specifies his next move by entering the index of the piece he wishes to move, followed by the index of the square he wishes to move it to.(e.g. A3B4 will move the piece at A3 to the square B4). The only exception is in the case of promotion, where the user must specify an additional character indicating the desired piece. (e.g. H7H8B will promote the pawn to a bishop).

Undo and Redo

The game could be undone till the first move.

Save and Load

- At any time, the game could be saved to a file.
- A saved game could be loaded and continued.
- No need to save the undo data.

3 Game Flow

Loop till the end of the game:

1. Read a move from Player 1.
2. Check if it's a valid move.
3. If not valid, ask for another move.
4. Do the move.
5. If the move takes out one of the pieces of the other player, add this piece to the list of taken out pieces displayed.
6. Check if the move results in a **Check/Checkmate/Stalemate/Promotion** and display a **message** accordingly.
7. Repeat the same logic for **Player 2**.

4 Implementation Notes

- You are required to implement the game using the C programming language.
- [**Top Priority**] Your program **MUST NOT crash** under any circumstances, even against malicious users!
- All users' inputs should be validated. Any missed validation will reduce your marks.
- Take care of the following:
  - Naming conventions.
  - Abstraction and code reusability.
  - Code organization and style.
- You are required to structure your code in multiple directories, multiple **.h** and **.c** files.
- You are required to build your project using a makefile.
- Your are required to use git for version control and github for team collaboration (using branches and pull requests).
- You are welcome to add extra features to your game, good software engineering practices, and optimization in time and space. Good extra work will be graded as Bonus.

5 Deliverables

- Complete project and source files.
- A well structured **Report** containing the following:
  - Description of your application, and the features it provides.
  - Overview of your design.
  - Any design/implementation assumptions made, and details you find necessary to be clarified.
  - Description of all used data structures.
  - Description of the important functions/modules in your implementation.
  - **User Manual** that describes how to deal with your application.
  - Sample runs.

  - **References:** Any copied material from anywhere should be referenced. Any discovered unreferenced copied material will result in severe reduction in your grade.
  - Any extra work that you have done (features, optimizations, software engineering practices).

6 Notes

- You are required to work in groups of **two**.
- Take your time to design the project and discuss your design / ask questions about it before the implementation.
- Start in the project early and come forward with your questions.
- All actual programming should be an independent effort. If any kind of cheating is discovered, penalties will apply to all participating students.
- Late submissions are not accepted.