Software Specification
V1.1

Developer:

Kevin Zhu
Yuan Wang
Chenyu Mou
Tiantian Lu
Pengfei Yan

Affiliation: **University of California, Irvine**

Table of Contents

Glossary

API - Application programming interface

Board - A place where the whole game happens and displays every move of every piece.

Pieces - Each pieces of the chess game

AI - The computer

Enum - a data type used in C

Struct - a data type that contains other data types in C

IO - contains the input and output.

1. Software Architecture Overview

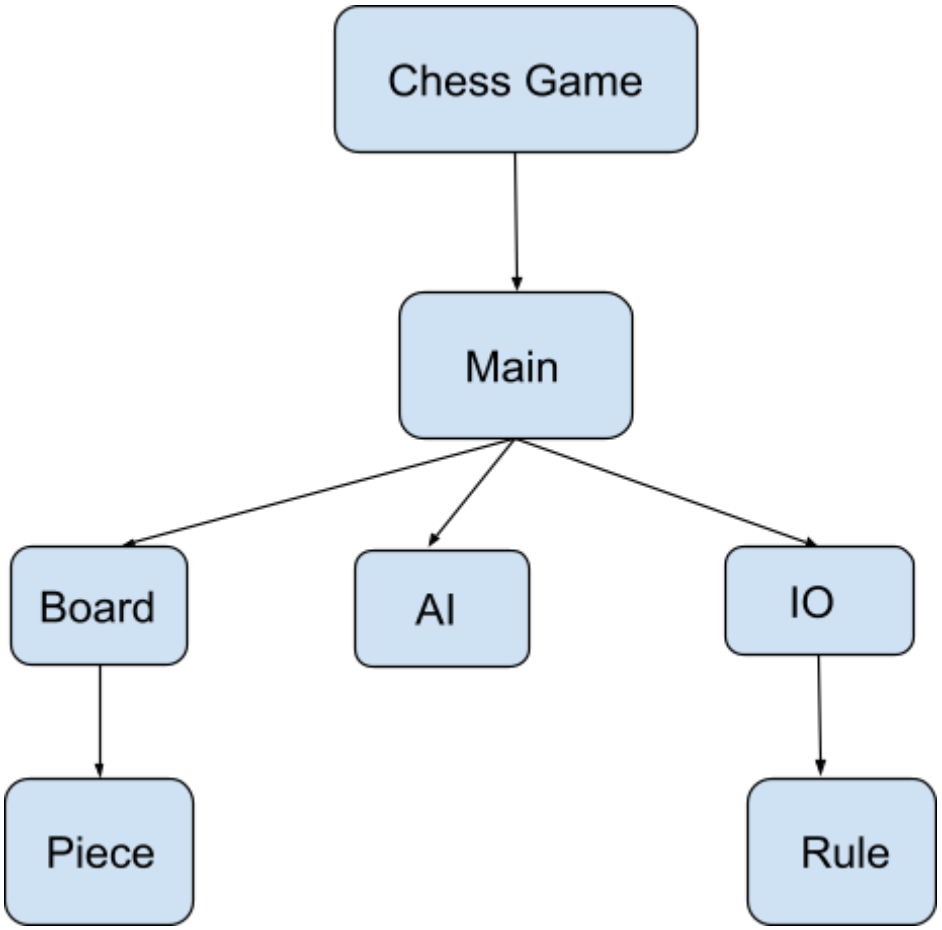
1.1 Main data types and structures

Data Type: We will use **class**, **enum**, as our main data types.
The structures we will use are:

- typedef struct piece(Contains chess types and chess color)
- typedef enum { PAWN,ROOK ,BISHOP ,KNIGHT ,QUEEN, KING }piece_type; (To make a define for a type of chess more convenient)
- typedef coords (To determine the coordinates of the chess, TBD)

1.2 Major software components

- **Diagram of the module hierarchy**



1.3 Module interfaces

- **API of major module functions**

Main.c: Representing the menu and setup of the game. Containing the main function.

Dependencies: IO module, Board module, AI module

possible header files: <stdio.h>, <stdlib.h>, <math.h>, <time.h>

Rules.c, Rules.h: After the user inputs their move in the terminal(IO module), the Rule module is used to check whether the user makes the legal move or not.

Board.c, Board.h: The Board module is responsible for the initialization of the new chessboard and updating the pieces on the chessboard after the user make move.

Void Blankboard(c_piece *board[8][8]) for initialization

Void Copyboard(c_piece *oldboard[8][8], c_piece *newboard[8][8]) for updating

Dependencies: Piece module

Piece.c, Piece.h: The Piece module is responsible for setting up the chess pieces on the board

AI.c, AI.h: When it is the user's turn to make move, the AI module will analyze all potential possible movements and calculated the best one for the user.

(Require <math.h> to calculate the necessary steps)

IO.c, IO.h: The IO module takes the user's input string and transforms it into variables for the function to implement. It also records every single legal move made by the user.

(require <stdio.h> for printf and scanf)

int changeMove(char *move, o_coords *from, p_coord *to)

Arguments: move -user string input

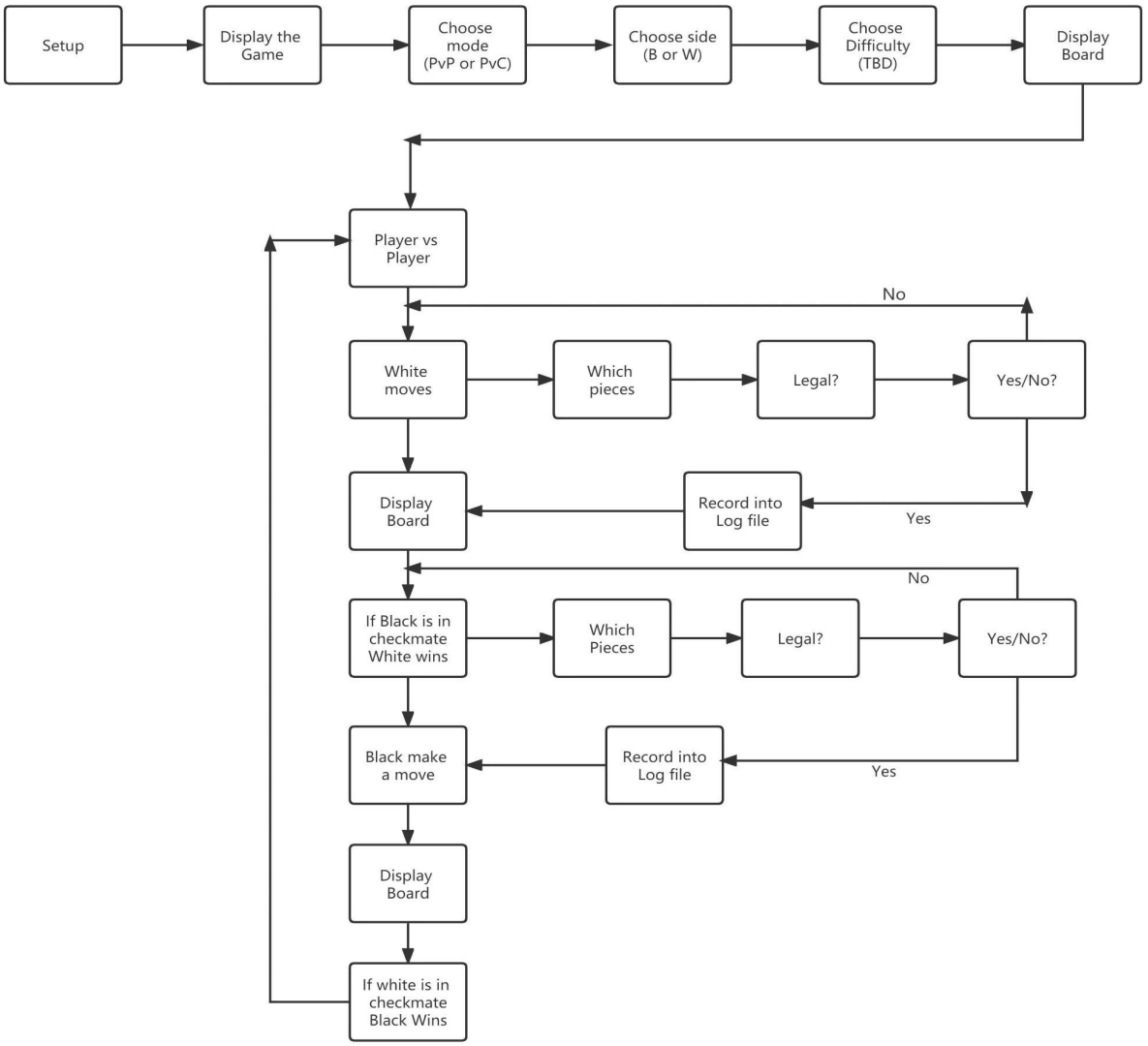
from - the piece location user want to manipulate

to - the destination of the moving piece

Result: return 0

Dependencies: Rule module to examine whether user input is legal or not.

1.4 Overall program control flow



2. Installation

2.1 System Requirement

- Linux based on OS
- Minimum of 2GB RAM
- Monitor, Keyboard, and Mouse
- Equipped GCC and GNU make

2.2 Setup and Configuration

1. First, open a terminal window or Putty window.
2. Logging into a Linux account given by the developers.
3. Use the "cd" command to go to the folder where the game at.
4. Unpack the source code from the archive by “ tar -xvzf chess.tar.gz” (For now, our file name is chess)
5. Compile the source code of the program by “ make clean” and then “make”.
6. Run the compiled program by the command ” ./chess”.

2.3 Building, compilation, installation

1. Build: Type the command “ **Make all**”.
2. To launch the program: Firstly go to the directory which includes the chess game and then type the command” **./Chess**”.

3. Documentation of packages, modules, interfaces

3.1 Detailed description of data structures

Pieces:
This struct uses two enumerated types to define a chess type. Piece_type is used to define a specific piece on the board, whether it is a king, queen, or other pieces. Color used to define these pieces is in which color (black and white).

```
typedef struct{
enum piece_type type;
enum c_color color;
}c_piece;
```

chess type:

```
typedef enum{
PAWN,
ROOK,
KNIGHT,
QUEEN,
KING,
BISHOP} piece_type;
```

color:

```
typedef enum{
BLACK = 1,
WHITE = 0,
}c_color;
```

Position:
Positions contain the specific coordinates for a piece on the board. Each piece has a y position and a x position.

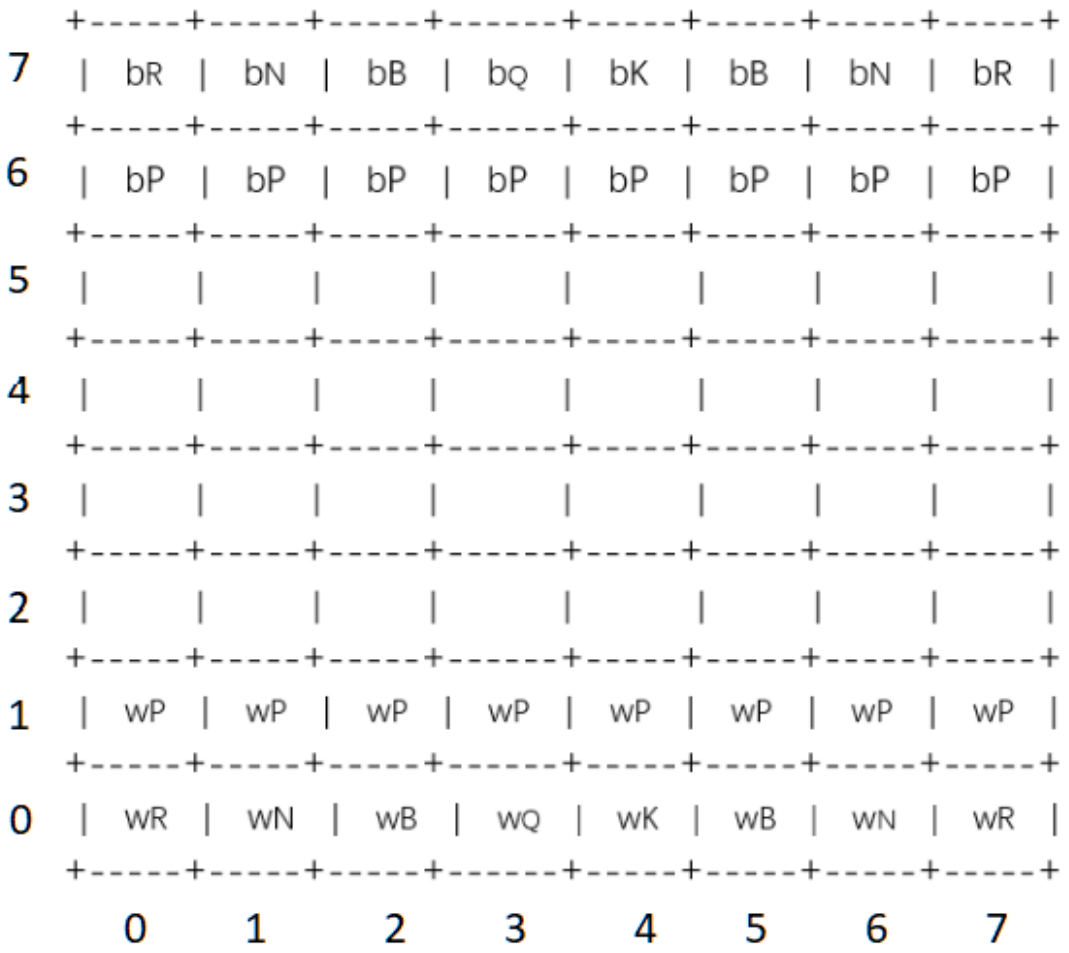
```
type struct{
int pos_y;
int pos_x;} c_positions;
```

3.2 Detailed description of functions and parameters

void StartMenu();
Print out the initial menu when the program begins. Then get the input from the user for the game mode.

void GameMenu();
Print the instruction and get the color from the user.

void printBoard(c_piece* gameboard[8][8]);
Print the current board with the position of pieces.



piece_type GetType(c_piece *p);
When we want to get the type of a piece, we can use this function to return the type of a piece and use it directly.

c_color GetColor(c_piece *p);
When we want to get the color of a piece, we can use this function to return the color of a piece and use it directly.

void Blankboard(c_piece *board[8][8]);
This function is used to initialize a blank board with no peices

void CreateBoard(c_piece *board[8][8], c_piece* pieces[12]);
This function is used to generate a board with pieces when a game starts. It initializes each piece.

void addPiece(c_piece *board[8][8], piece_type type, c_color color, c_coords position);
This function is used mainly in the promotion. It adds a new piece to specific position on the board.

void Copyboard(c_piece *oldboard[8][8],c_piece *newboard[8][8]);
Copy the old board to a new board, ssed when we have to upgrade our board.

void move(c_piece *board[8][8],int x1, int y1, int x2, int y2);
This function is used to move a piece in the board. It asks the board and the coordinates made of 4 integers.

void Undo(c_piece *board[8][8],int x1, int y1, int x2, int y2);
When the player want to undo his or her last step, he or she need to use this function to move a pieces back.

int CheckMove(c_piece *board[8][8], int x1, int y1, int x2, int y2);
This function is used to check whether a movement of a piece is legal. It will return 1 if it is legal and 0 for illegal movements.

3.3 Detailed description of input and output formats

In the program, the user should use the terminal to input any command.

When choosing the mode, user should input **1** for a game with 2 local players and **2** for a game between 1 player and AI, and **3** to exit the program.

When choosing the color, user should input **1** for white that goes first and **2** for black that goes second.

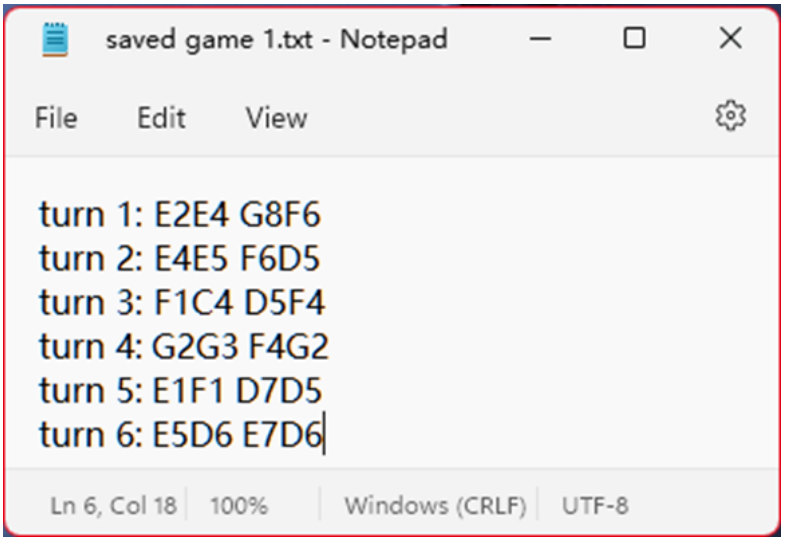
For each turn, player should input 4 numbers. The first numbers are the position of the piece the player chooses to move. The second two are the final position of the piece. For example, **0102** means you want to move the piece in A2 1 steps forward to A3.

After one game, the user can input **Save** to save a txt log file that contains every movement in that game. It will give you something like:

turn 1: E2E4 G8F6

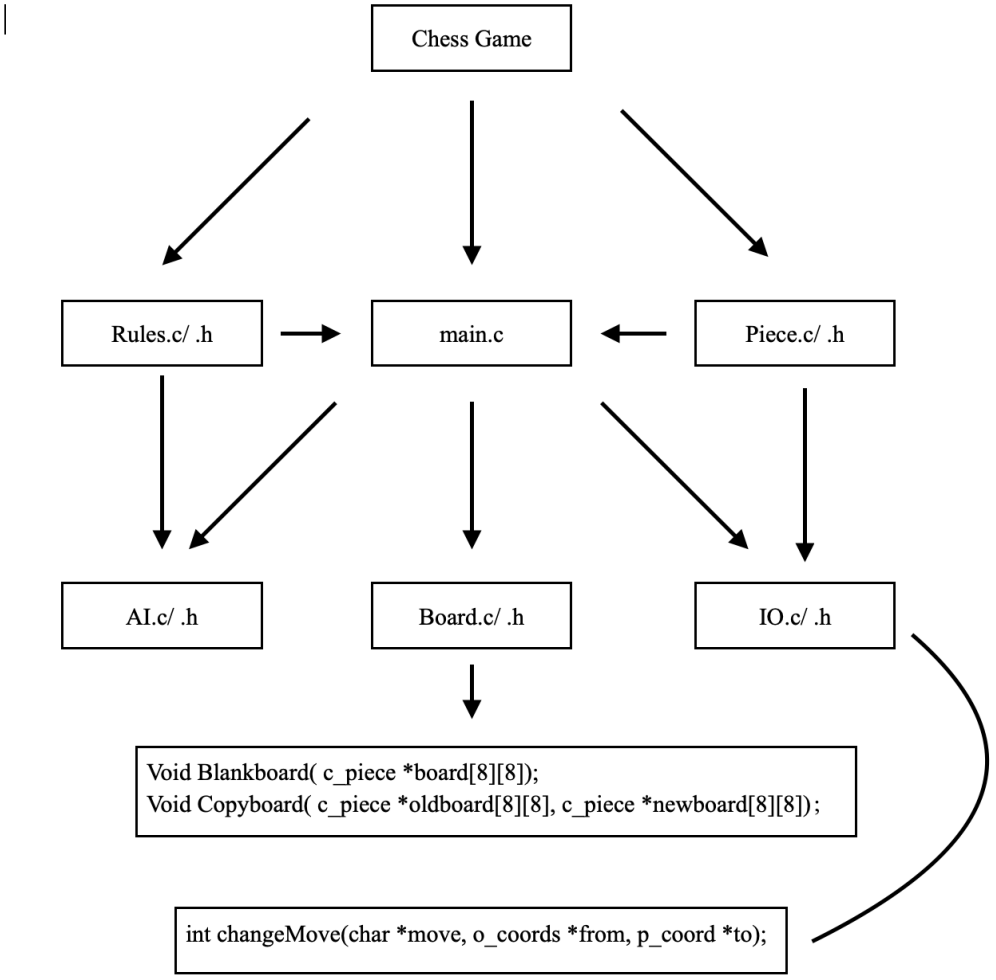
turn 2: E4E5 F6D5

In each turn, the first 4-character string is the movement of white and the second one is that of the black.



4. Development plan and timeline

4.1 Partitioning of Tasks



4.2 Team Member Responsibilities

Kevin Zhu:	AI.h/AI.c, Board.c/Board.h
Yuan Wang:	AI.h/AI.c, Board.c/Board.h
Chengyu Mou:	Rules.h/Rules.c
Tiantian Lu:	IO.c/IO.h
Pengfei Yan:	Piece.c/Piece.h

5. Back Matters

5.1 Copyright

5.2 Error Messages

- "The color you entered does not exist. Please try again."
  This message is displayed if the player enters a color that is neither black nor white.

- "The opponent you entered does not exist. Please try again."
  This message is displayed if the player enters a string that is neither player nor computer.

- "You have not made any move yet. Please make a move."
  This message is displayed if the player attempts to undo a previous move when the player has not made the first move.

- "You are attempting an illegal move. Please try again."
  This message is displayed if the player
  1. enters a displacement that is against the rule of the displacement of the selected piece.
  2. enters an initial/ final position that is not on the board.
  3. enters a final position that has one of the player's pieces.

- "You do not have a piece on the position you entered. Please try again."
  This message is displayed if the player enters an initial position that does not have any piece of the player's.

- “Time out. You lose the game.”
  This message is displayed if the player does not enter any displacement within 60 seconds of the player’s turn.

5.3 Index