Team 9
Jeffrey Ching, Jayniel Gagui, Fabian Hernandez, Vanessa Lauridsen, Varvara Vorobieva, Tangqin Zhu

Chess GPT
Software Architecture Document

Version Alpha
04/24/2023

Table of Contents

Appendix A: Glossary

Array - An array is a series of memory locations – or 'boxes' – each of which holds a single item of data, but with each box sharing the same name.

Assert - Function can be used to add diagnostics into the C program.

Assert.h - Header file that has to be included to use assert function.

Char - Char is a computer science term referring to a single display unit of information equivalent to one alphabetic symbol, digit, or letter.

Continue - Statement forces the next iteration of the loop to take place, skipping any code in between.

Data Type -A data type is the classification for the type of data that a variable or function will accept.

Do-while loop -A statement that repeats at least once, and keeps repeating, until the test expression is no longer true.

Enum - A user-defined data type in C. It is used to assign names to integral constants, the names make a program easy to read and maintain.

Fflush- Function flushes the output buffer of a stream.

For loop - A statement that repeats a block of code a set amount of times.

Free - Function deallocates the memory previously allocated by a call to calloc, malloc, or realloc.

Function - A function is simply a "chunk" of code that you can use over and over again, rather than writing it out multiple times.

Function Prototype - A function prototype is a definition that is used to perform type checking on function calls when the EGL system code does not have access to the function itself.

if/else - A statement that executes two different codes depending on whether the test expression is true or false.

#include - The #include directive tells the C preprocessor to include the contents of the file specified in the input stream to the compiler and then continue with the rest of the original file.

Int - Integers are numbers without a fractional component, and don't support decimal points.

Malloc - Function that takes the size of the memory block as an argument, and allocates the requested memory and returns a pointer to it.

Pointer - A pointer is a variable that stores the memory address of another variable as its value.

Printf - The printf() function sends a formatted string to the standard output (the display).

Return - Statement that ends the execution of a function, and returns control to the calling function.

Return Type - In computer programming, the return type (or result type) defines and constrains the data type of the value returned from a subroutine or method.

Scanf - The scanf() function is a commonly used input function in the C programming language. It allows you to read input from the user or from a file and store that input in variables of different data types.

Signed - A data type that is guaranteed to be signed.

Stdio.h - stdio. h is a header file which has the necessary information to include the input/output related functions in our program. Example printf, scanf etc.

Stdlib.h -Header file that defines 4 variable types, several macros, and various functions for performing general functions.

Strcmp -Function compares two strings character by character. If the strings are equal, the function returns 0.

String - An array of characters (char data type).

Struct - Structures (also called structs) are a way to group several related variables into one place. Each variable in the structure is known as a member of the structure.

Typedef - The typedef is a keyword used in C programming to provide some meaningful names to the already existing variable in the C program.

Unsigned -A data type that is one of the type modifiers which are used for altering the data storage of a data type.

Void - When used as a function return type, the void keyword specifies that the function doesn't return a value.

1 Software Architecture Overview

1.1 Main Data Types and Structures

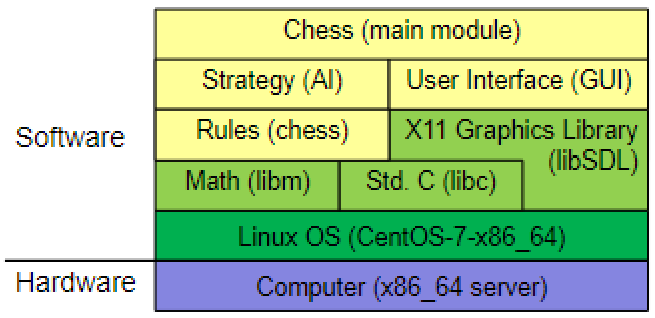
Main Data Types used:

- Int : Rank
- Char : File
- Void : certain functions return type
- Enum : Player Color, Piece Type
- Struct: represents a Piece on the board that contains the Player Color and Piece Type

1.2 Major Software Components

Important software components:

1. Board Layout - board and pieces represented by an 8 x 8 2D array of structs (squares on board)
2. AI - allowing the user to play against a computer; calculates legal and best move to make
3. User Interface - making sure the user is able to set up and play the game without any confusion
4. Special Move Set - implementing en passant, castling, capturing and pawn promotion
5. Logging - track moves and provide debugging functionality
6. Diagram of module hierarchy



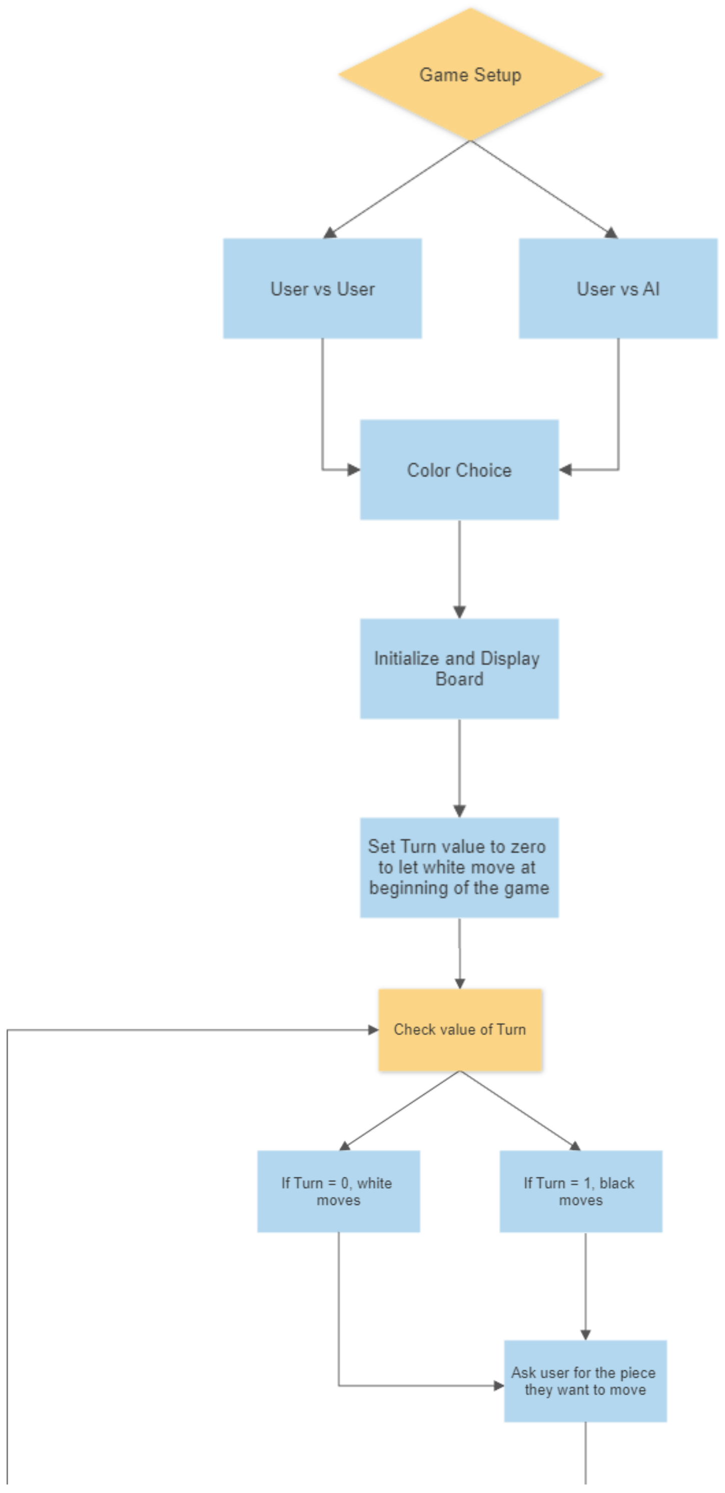
1.3 Module Interfaces

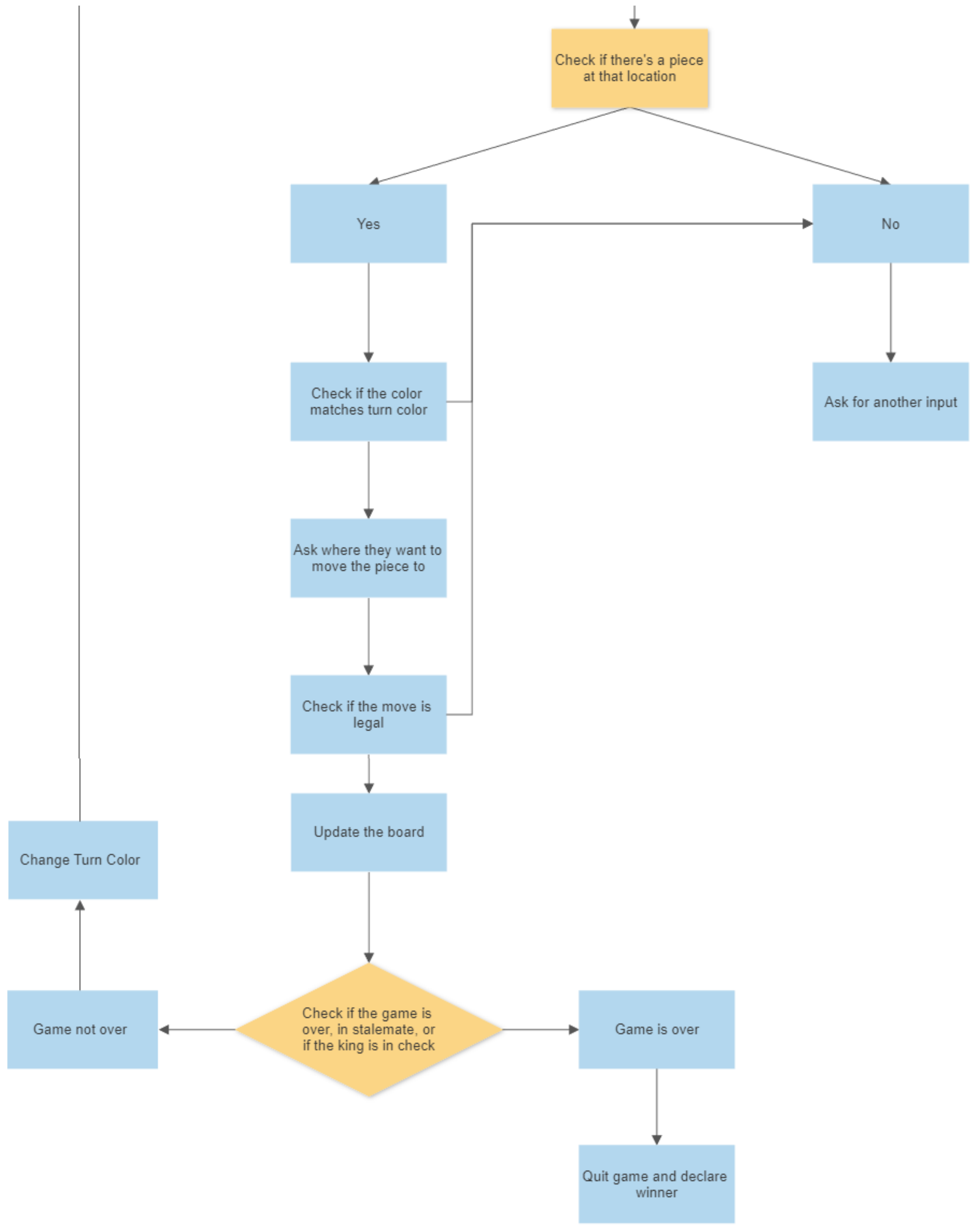
1. Initializing the board
2. Update the board once a move is made
3. Move Set - make sure the move made by the user is legal
4. Game Logic - After a player moves a piece, a function will be called to see if the king is in check, the game is in stalemate, or if the game is over by seeing if the king can't move without putting itself in check. If the stalemate or game end is triggered then the game will be over and the user(s) will be asked if they want to do a new game or exit.

5. Error Handling - display messages to the user if an improper move was made

1.4 Overall Program Control Flow

1. Allow the user to set up the game to their liking
    a. User vs User or User vs AI
    b. Color choice
2. Initialize and display the board and pieces to the user
3. Initialize Turn Color to zero for white. This will give white the first move of the game
4. Input from white
    a. Take the input from the user and validate if the move is legal
    b. Ask the user for a new input if the move was illegal
5. Change Turn Color to one for black
6. Update the board and pieces when a move is made
7. Check if the king is in check, the game is in stalemate, or if game is over
8. Input from black
    a. Take the input from the user and validate if the move is legal
    b. Ask the user for a new input if the move was illegal
9. Change Turn Color to zero for white
10. Update the board and pieces when a move is made
11. Check if the king is in check, the game is in stalemate, or if game is over.
12. Loop 4-11
13. When the game is over, clear the board and ask if the user wants to play again or exit the game


Game Setup
User vs User
User vs AI
Color Choice
Initialize and Display Board
Set Turn value to zero to let white move at beginning of the game
Check value of Turn
If Turn = 0, white moves
If Turn = 1, black moves
Ask user for the piece they want to move


Check if there's a piece at that location
Yes
No
Check if the color matches turn color
Ask for another input
Ask where they want to move the piece to
Check if the move is legal
Update the board
Change Turn Color
Game not over
Check if the game is over, in stalemate, or if the king is in check
Game is over
Quit game and declare winner

2 Installation

2.1 System Requirements, Compatibility

Hardware Requirements:

- Computer (x86_64 server)

Software Requirements:

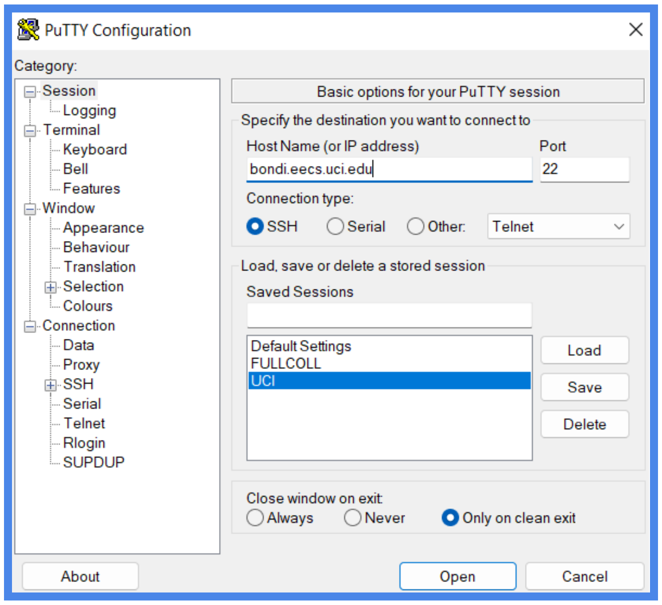
- Linux OS (CentOS-7-x86_64)

2.2 Setup and Configuration

Accessing The Linux Server

To access ChessGPT you must first install a SSH client such as PuTTy for Windows computers in order to access the shared Linux server.

After installation, open PuTTy and under host name you must either use *bondi.eecs.uci.edu* or *crystalcove.eecs.uci.edu* as shown below:


PuTTY Configuration

Category:

Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address): bondi.eecs.uci.edu

Port: 22

Connection type: [✓ SSH] Serial Other: Telnet

Load, save or delete a stored session

Saved Sessions

Default Settings
FULLCOLL
UCI

Close window on exit: Always Never [✓ Only on clean exit]

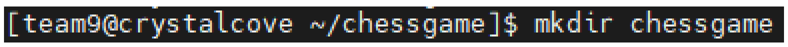
Note: Make sure SSH is selected under connection type and port is set to 22.

2.3 Building, Compilation, Installation

Navigating the Linux Server

To install our program, you should first make a directory in order to organize the installation of the game. You can name the directory to your liking, but in this case we'll name the directory "chessgame". To do this type the command as shown below:

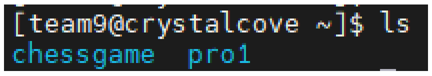
```
mkdir chessgame
```



```
[team9@crystalcove ~/chessgame]$ mkdir chessgame
```

To confirm that the directory was made, type the command as shown below:

ls

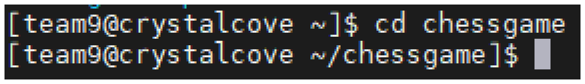


```
[team9@crystalcove ~]$ ls
chessgame  pro1
```

We should now see that the directory chessgame was created.

Next, we want to move into the chessgame directory, type the command as shown below:

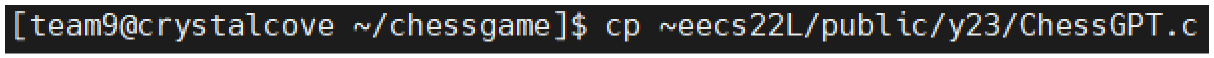
cd chessgame



```
[team9@crystalcove ~]$ cd chessgame
[team9@crystalcove ~/chessgame]$
```

Now that we're in the correct directory, we can now copy the game into the directory by typing the command as shown below:

cp ~eecs22L/public/y23/ChessGPT.c

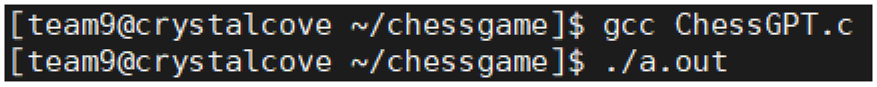


```
[team9@crystalcove ~/chessgame]$ cp ~eecs22L/public/y23/ChessGPT.c
```

The final step is to compile and run the program. To do this type the following two commands and the game will execute.

gcc ChessGPT.c

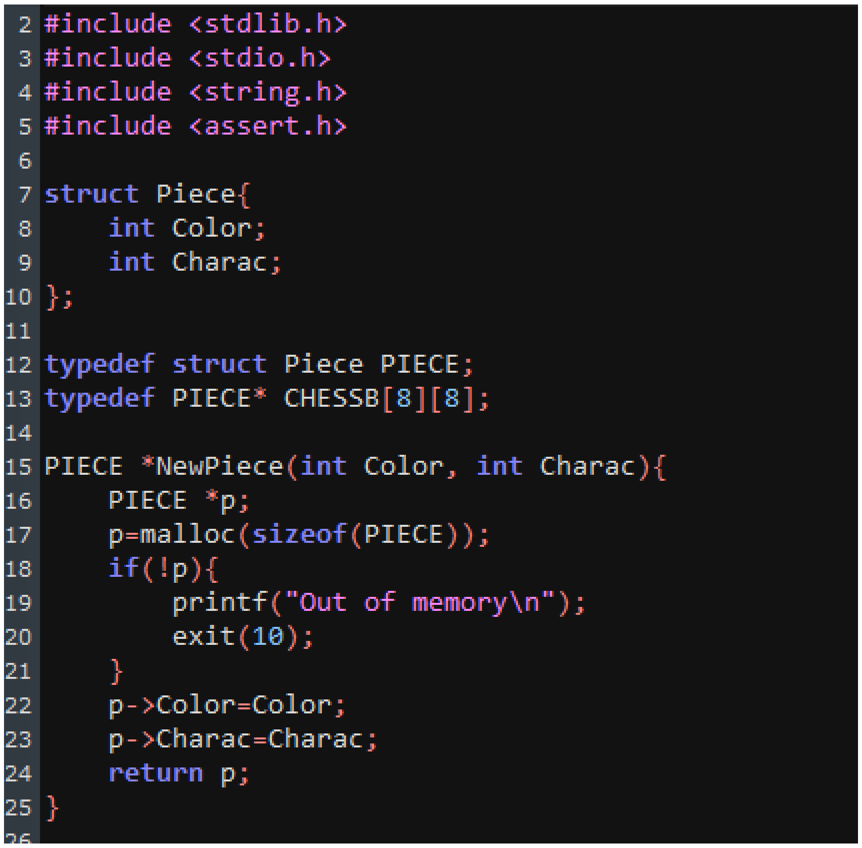
./a.out



```
[team9@crystalcove ~/chessgame]$ gcc ChessGPT.c
[team9@crystalcove ~/chessgame]$ ./a.out
```

3 Document of Packages, Modules, and Interfaces

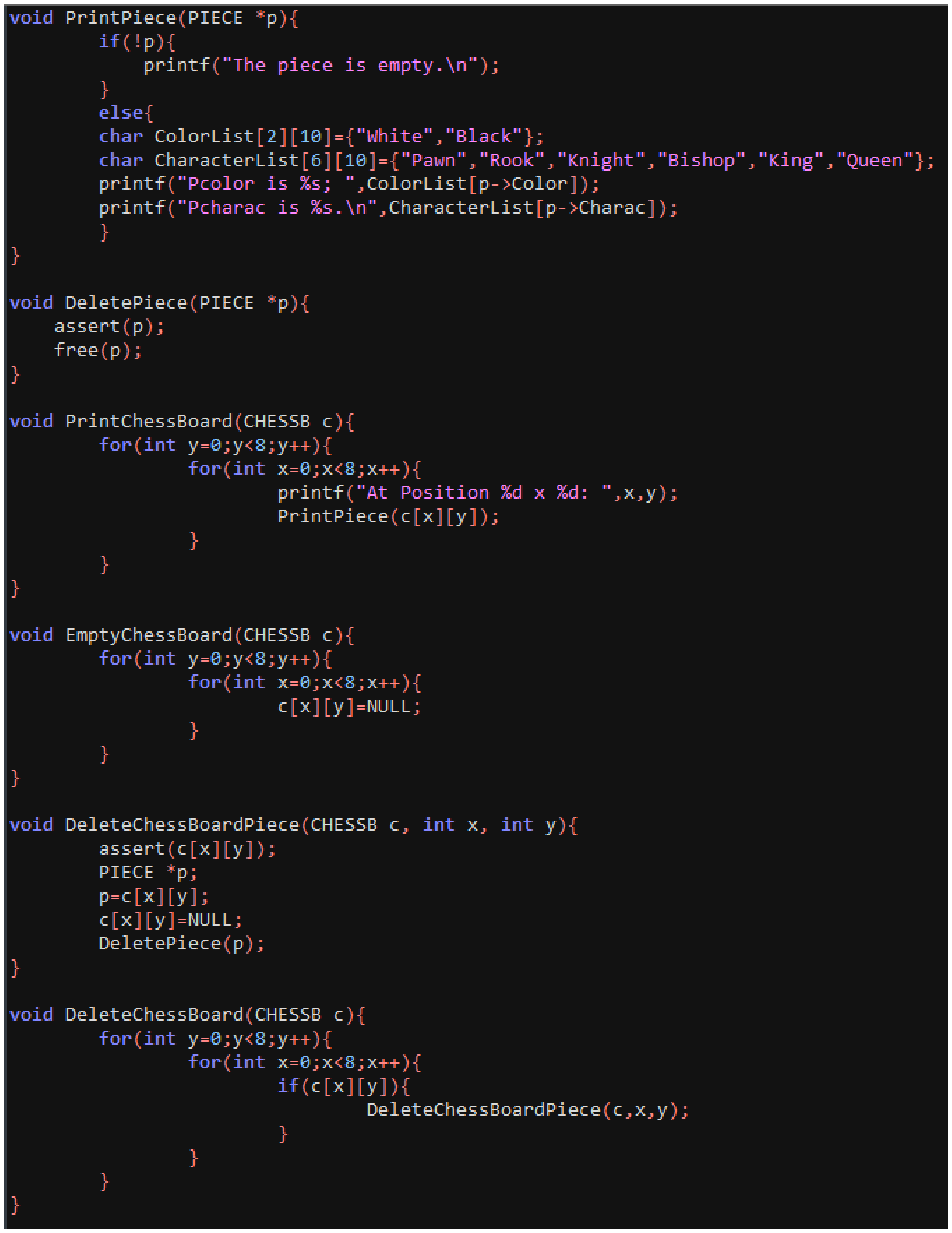
3.1 Detailed Description of Data Structures



```c
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <assert.h>

struct Piece{
    int Color;
    int Charac;
};

typedef struct Piece PIECE;
typedef PIECE* CHESSB[8][8];

PIECE *NewPiece(int Color, int Charac){
    PIECE *p;
    p=malloc(sizeof(PIECE));
    if(!p){
        printf("Out of memory\n");
        exit(10);
    }
    p->Color=Color;
    p->Charac=Charac;
    return p;
}
```

To create our chess game, we need different data structures to show various components of the game. Here we have a snippet of our code representing the chessboard and the chess pieces. To create our board we have an 8x8 array. For the chess pieces, we have a struct to define the chess pieces by their color and character. This information is then passed to the ‘NewPiece’ function that returns a pointer to the ‘PIECE’ object. If we get no errors, memory will be allocated using malloc to create a ‘PIECE’ object.

3.2 Detailed Description of Functions and Parameters

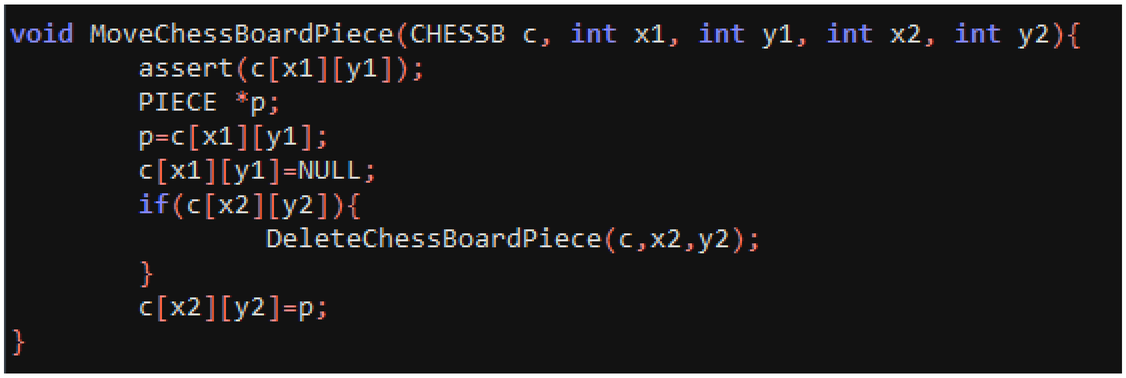


```
void PrintPiece(PIECE *p){
        if(!p){
            printf("The piece is empty.\n");
        }
        else{
        char ColorList[2][10]={"White","Black"};
        char CharacterList[6][10]={"Pawn","Rook","Knight","Bishop","King","Queen"};
        printf("Pcolor is %s; ",ColorList[p->Color]);
        printf("Pcharac is %s.\n",CharacterList[p->Charac]);
        }
}

void DeletePiece(PIECE *p){
    assert(p);
    free(p);
}

void PrintChessBoard(CHESSB c){
        for(int y=0;y<8;y++){
                for(int x=0;x<8;x++){
                        printf("At Position %d x %d: ",x,y);
                        PrintPiece(c[x][y]);
                }
        }
}

void EmptyChessBoard(CHESSB c){
        for(int y=0;y<8;y++){
                for(int x=0;x<8;x++){
                        c[x][y]=NULL;
                }
        }
}

void DeleteChessBoardPiece(CHESSB c, int x, int y){
        assert(c[x][y]);
        PIECE *p;
        p=c[x][y];
        c[x][y]=NULL;
        DeletePiece(p);
}

void DeleteChessBoard(CHESSB c){
        for(int y=0;y<8;y++){
                for(int x=0;x<8;x++){
                        if(c[x][y]){
                                DeleteChessBoardPiece(c,x,y);
                        }
                }
        }
}
```

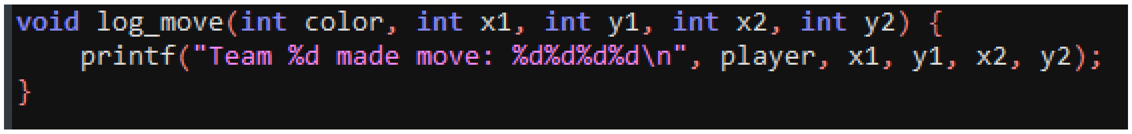
In this snippet of code we have various functions like printing the board and pieces, clearing the board, and deleting pieces. For the function 'PrintPiece' we take a pointer to 'PIECE' to get information from the two arrays 'CharacterList' and 'ColorList' in order to get the piece's color and character. This information is fed into the corresponding string values to print the pieces. The 'PrintChessBoard' function gets its inputs from 'CHESSB' and prints the chess pieces on the chessboard by calling the 'PrintPiece' function. To clear the board we have the function

‘EmptyChessBoard’ which sets all the elements to NULL. To delete a piece at a specific position we have the ‘DeleteChessBoardPiece’ function which gets its x and y coordinates as inputs and looks at that position of the board. It uses assert to see if the position is occupied and if it is the function ‘DeletePiece’ is called.

3.3 Detailed description of input and output formats



```
void MoveChessBoardPiece(CHESSB c, int x1, int y1, int x2, int y2){
        assert(c[x1][y1]);
        PIECE *p;
        p=c[x1][y1];
        c[x1][y1]=NULL;
        if(c[x2][y2]){
                DeleteChessBoardPiece(c,x2,y2);
        }
        c[x2][y2]=p;
}
```

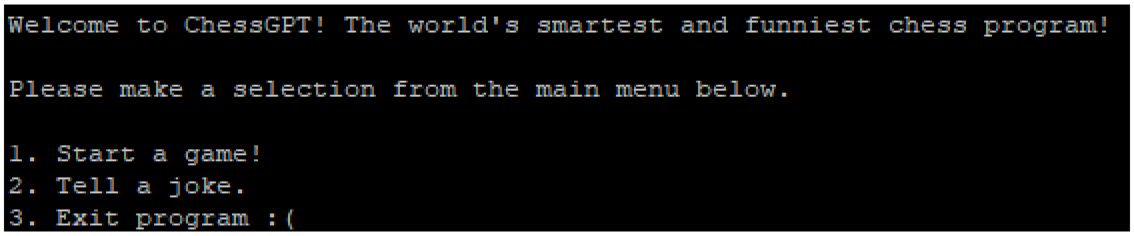


```
void log_move(int color, int x1, int y1, int x2, int y2) {
    printf("Team %d made move: %d%d%d%d\n", player, x1, y1, x2, y2);
}
```

To move a piece on the board, we take the input of the piece’s current position for x1 and y1 and then take x2 and y2 for the new position of the piece. Assert is used to see if the space in the new position is being occupied and if it is, then that piece would be captured and deleted with the function ‘DeleteChessBoardPiece’. For logging moves, we have a function called ‘log_move’ which can be used to log each move from both users. It takes the team color and the current position and new position of the piece to track what the user is doing.

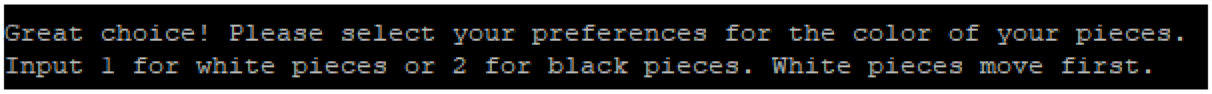
Input and Output order(Game):

The user will be loaded into the Main Menu where they will be asked to play the game, tell a joke, or quit the game.



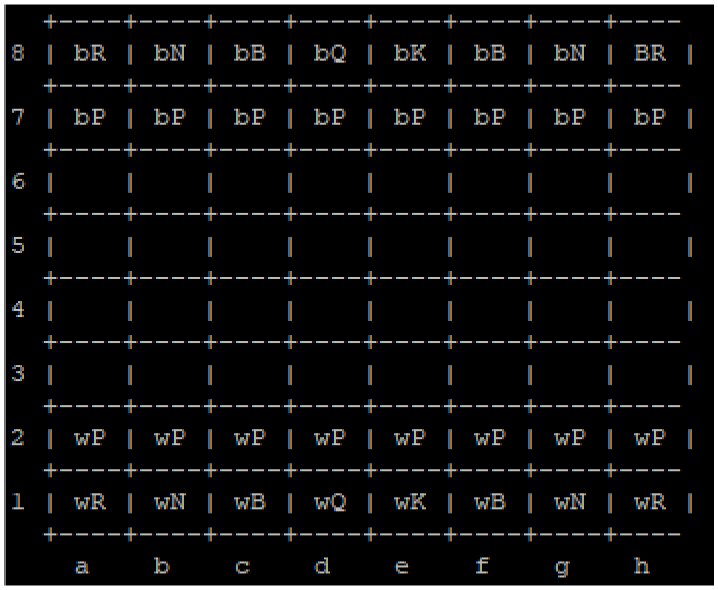
```
Welcome to ChessGPT! The world's smartest and funniest chess program!

Please make a selection from the main menu below.

1. Start a game!
2. Tell a joke.
3. Exit program :(
```

If the user types 1, a new menu will appear asking if the user wants to be white or black.



```
Great choice! Please select your preferences for the color of your pieces.
Input 1 for white pieces or 2 for black pieces. White pieces move first.
```

Once the user inputs the color preference the board should load in with pieces.



```
  +----+----+----+----+----+----+----+----
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
```

The user will be asked to input the rank and file. The other player will be asked the same, and the A.I. will move automatically. This will happen until the user or the AI's king get checkmated. Then the program will return the user to the main menu to prompt them to play again.

4 Development Plan and Timeline

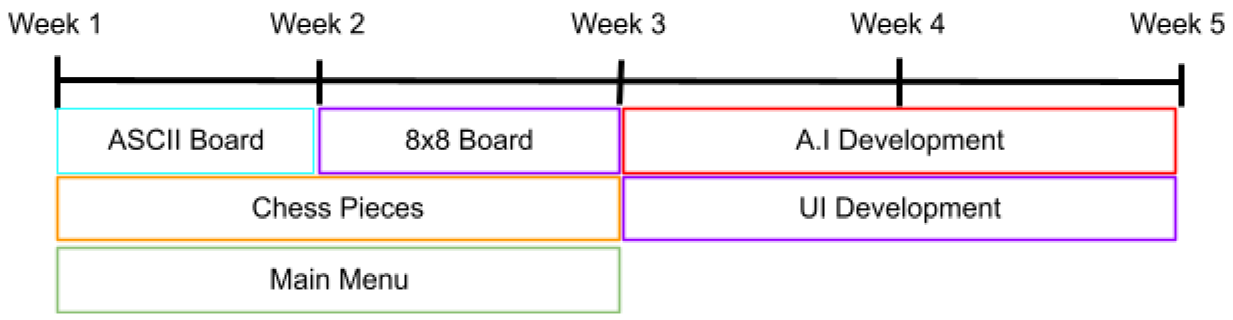
4.1 Partitioning of tasks

This is the order the team has agreed on, which we believe was important to start the code

1. Main Menu
    a. Menu asking the user to play the game
    b. Implement quit option
    c. Tell a funny joke
2. Board
    a. 8x8 main board to store piece type and color
    b. Other board that stores ASCII values to make it easier to print
    c. Possible add a colored board
3. Movement
    a. Implement different types of movements

    b. Create illegal movements.
4. Chess Pieces
    a. Connect each movement type to their designated piece
    b. Certain pieces can make specific movements or rules
5. A.I. Development
    a. AI will be able to make smart moves
    b. AI will be able to versus another AI

Estimated Timeline



4.2 Team Member Responsibilities

Team members, according to their strengths and weaknesses were placed in their corresponding sub teams. Tangqin and Fabian make up the Board team and will work together to create the ASCII and board tables. ASCII Table members will work with UI and main menu to display the pieces on the board. While the Table member will work alongside the Chess Pieces and Movement teams (Vanessa and Jay) to ensure the board is adequate for the task. Varvara will work on generating functions that receive user input and the main menu and UI. Jeffery will integrate all the modules to make sure everything works correctly and test the application.

Back Matter

Copyright

Alpha Version April 2023

Program and Manual by Jeffrey Ching, Jayniel Gagui, Fabian Hernandez, Vanessa Lauridsen, Varvara Vorobieva, and Tangqin Zhu.

For any questions contact help@chessgpt.com.

Published by Chess GPT Inc.

Index

References

Jarvis, Matt, and Dion Dassanayake. “How to Play Chess for Beginners: Rules, Moves and Setup.” *Dicebreaker*, Dicebreaker, 13 Mar. 2023, https://www.dicebreaker.com/games/chess/how-to/how-to-play-chess.

Serper, Gregory, and Nathaniel Green. “How to Play Chess: Learn the Rules & 7 Steps to Get You Started.” *Chess.com*, Chess.com, 28 Nov. 2022, https://www.chess.com/learn-how-to-play-chess.