Chess Notation Traversal

DS Project 1st Year 1st Semester

RISHI VARDHAN-D1
A SHASHIKANTH- D3
E VIVEK SAI SUBRAMANYAM-E6
KARTHIK CHOWDARY-F6
SHASHANK-J8
GRIET
IT-C(2022-26)

INDEX

Overview

This is an implementation of Chess moves based on notation in C. It uses Doubly Linked Lists and Arrays in order to validate, store and display the chess moves to the user.

The usage of DLL is to allow for free traversal in both the forward and backward direction while examining the list of existing moves. This allows for the analysis of chess moves that were done by two users.

Goals

1. To develop a Graphical User Interface for displaying chess board and its current position.
2. To build a DLL using structure data type.
3. To take input from the user to play a chess move, validate it, and enter it into DLL for traversal.
4. To allow for the traversal of the DLL and display the corresponding move to the user.
5. To handle errors and invalid inputs from the user.

Chess

Chess is a board game for two players. It is played on a square board, made up of 64 smaller squares, with eight squares in each row and column.

Each player starts with sixteen pieces: eight pawns, two knights, two bishops, two rooks, one queen and one king. The goal of the game is for each player to try and checkmate the king of the opponent. Checkmate is a threat ('check') to the opposing king which no move can stop. It ends the game.

During the game, the two opponents take turns to move one of their pieces to a different square of the board. One player ('White') has pieces of a light color; the other player ('Black') has pieces of a dark color. There are rules about how pieces move, and about taking the opponent's pieces off the board. The player with white pieces always makes the first move.

With the advent of computers, this game has gone digital from the 1970's. The first chess website, which allowed playing through a graphical interface, was Caissa.com (known at the time as Caissa's Web) which launched in 1995. Since then, a number of chess websites have been developed.

Chess Engines

The highly strategic nature of chess and the highly organized and codified nature of the game's rules make it very suitable to work out the best moves using computers. Humans are limited in their ability to grasp and work out all the moves that can be done in the future as the possible numbers of variations that a game can lead to in any position can sometimes run into billions. Trying to figure out the best possible move from all these possible variations is a task well suited to computers rather than humans. This was realized from the advent of modern computers itself, with Alan Turing himself, the father of modern computer science,creating the first computer chess playing algorithm in the 1950s.

In computer chess, a **c**hess engine is a computer program that analyzes chess or chess variant positions, and generates a move or list of moves that it regards as strongest.The rapid advancement of computing in the 1960s and 1970s was key in increasing chess engine strength, both drastic software and hardware innovations lead to stronger engines.

The software advancements include the most iconic game algorithm of all, the Minimax algorithm and its alpha-beta pruning optimization,that was and remains key to chess programming and optimization. This algorithm, initially proven in 1928 by John von Neumann, focuses on maximizing one player's score while minimizing the others. Major improvements to this algorithm would be developed specifically for chess programming, with the main goal of increasing search depth. These include:

- Move selection techniques
- Heuristic approaches
- Iterative deepening
- Opening/ending databases

To display the various moves played by a chess engine to the user in order to analyze the motives, an interface between human user and chess engine is required. Thus, this project is about building this interface, to receive data from the games played by engines and display it to the user.

Doubly Linked List

A doubly linked list is a data structure used in computer science to store and manage a collection of elements. It is an extension of the more basic singly linked list, where each node contains a data element and two pointers, one pointing to the next node in the list and another pointing to the previous node. This bidirectional linkage allows for more versatile operations compared to a singly linked list, as it enables traversal in both directions.

DLL in C

~~~~
struct Node {
    int data;
    struct Node* prev;
    struct Node* next;
};
```
~~~~

In a doubly linked list, each node contains three main components: the data element, a pointer to the previous node (often named 'prev'), and a pointer to the next node (usually named 'next'). The first node's 'prev' pointer and the last node's 'next' pointer typically point to NULL, indicating the boundaries of the list.

Advantages of Doubly Linked Lists

1. Bidirectional Traversal:Unlike singly linked lists, doubly linked lists allow traversing in both directions, which can be beneficial for certain applications.

2. Insertions and Deletions: Insertion and deletion operations are more efficient in a doubly linked list when the position of the element to be inserted/deleted is known, as you can directly update the pointers of the adjacent nodes.

3. Reverse Traversal: Doubly linked lists enable easy reverse traversal of elements, which can be useful in certain scenarios.

Applications of Doubly Linked Lists:

1. Text Editors: Doubly linked lists are used in text editors to manage and support undo and redo operations efficiently.

2. Browser History: Browser history can be implemented using a doubly linked list to track visited web pages and enable backward and forward navigation.

3. Music Players: Doubly linked lists can be used to create playlists in music players, allowing users to play songs in both directions.

4. Cache Management: In computer systems, doubly linked lists can be employed to manage cache entries and optimize data retrieval.

In conclusion, doubly linked lists offer bidirectional traversal, efficient insertion and deletion operations, and a range of applications in various fields. However, they require more memory due to the additional 'prev' pointers, and their implementation can be slightly more complex than singly linked lists. Understanding the intricacies of doubly linked lists is crucial for programmers aiming to design efficient data structures and algorithms.

Chess Game Documentation

Global Variables

a[8], b[8], c[8], d[8], e[8], f[8], g[8], h[8]:

Type: Character array of size 8

Purpose: Arrays representing the initial setup of pieces on each row of the chessboard. Each element in the array represents a square on the chessboard, and the characters represent the pieces in their initial positions (e.g., 'R' for rook, 'P' for pawn).

art[9], brt[9], crt[9], drt[9], ert[9], frt[9], grt[9], hrt[9]:

Purpose: Arrays to display the chess positions while forward and backward traversals. While simply traversing the moves these arrays will be used and the present position in these arrays can be wiped anytime in order to enter more moves. White the other set of arrays show the permanent present setup of the board and are non-updatable, only insertable.

count:

Type: Integer

Purpose: Keeps track of the number of moves made during the game.

ret[3]:

Type: Character array of size 3

Purpose: Used to store information about a captured piece during inserting a move into DLL. If a piece is captured, its position will be temporarily stored here (e.g., "e3" if a piece at that position is captured).

struct Chess

Type: Struct

Purpose: Represents each chess move. The struct contains the following fields:

popo[3]: Starting position of the piece (e.g., "Pe2").

post[3]: Destination position of the piece (e.g., "Pe4").

gone[3]: Represents a piece that was captured during the move (e.g., "N" for no piece captured, or the piece's position, like "e3" if a piece was captured at that position).

chess *start, *end:For implementing DLL

Functions

void chessstart():

Purpose: Displays the initial chess board setup with the pieces in their starting positions.

void chessmove():

Purpose: Allows the user to input a chess move and adds it to the DLL and the present move displaying arrays(a[8],b[8] etc.) using its helper functions.

const char* checkpos(char lol[3], int k, int m):

Purpose: Checks the validity of a chess move and updates the chessboard displaying arrays accordingly. It returns a string indicating whether the move was successful or if there was an issue with the move.

int letter(char b):

Purpose: Helper function to checkpos function to convert a character representing a column to its integer equivalent (e.g., 'a' to 0, 'b' to 1, etc.). This is done for convenience in the printing of chessboards, the rows are named a,b,c,d whereas it’s the columns that are named a,b,c,d in normal chess notation.

void movedis(chess* start):

Purpose: Displays the move history stored in the DLL. It prints the starting position, destination position, and whether any piece was captured during each move.

void movedis2(chess* start):

Purpose: The heart of the whole program. It takes the move history until the present move and lets the user go forwards and backwards by printing either the next move or the previous move.

main() Function:

The main() function is the entry point of the program. It contains an infinite loop that continuously takes user input and performs the following actions based on the input.

PROGRAM

```
#include<stdio.h>
#include<stdlib.h>
#include<string.h>
char a[8]="RNBKQBNR";
char b[8]="PPPPPPPP";
char c[8];
char d[8];
char e[8];
char f[8];
char g[8]="PPPPPPPP";
char h[8]="RNBKQBNR";
char art[9]="RNBKQBNR";
char brt[9]="PPPPPPPP";
char crt[9];
char drt[9];
char ert[9];
char frt[9];
char grt[9]="PPPPPPPP";
char hrt[9]="RNBKQBNR";
struct Chess{
   struct Chess *pre,*next;
   char popo[3];
   char post[3];
   char gone[3];
};
```

```
typedef struct Chess chess;
chess *start=NULL,*end=NULL;//HEAD  and tail POINTER
int count=0;
char ret[3];
void chessstart();
void chessmove();
const char* checkpos(char lol[3],int k,int m);
int letter(char b);
void movedis(chess *start);
void movedis2(chess *start);
void main(){
   int i;
   char prevpos[2],postpos[2];
   while(1){
   scanf("%d",&i);
   switch(i){
      case 1:chessstart();
      break;
      case 2:
      chessmove();
      break;
      case 3:movedis(start);
      break;
      case 4:movedis2(start);
      break;
      case 5:
      exit(0);
      default:printf("hmm");
      break;
   }
   }
}
void chessmove(){
   int i;
   chess *newnode,*temp=start;
   newnode=(chess*)malloc(sizeof(chess));
```

```
    char bef[3],posi[3];
    scanf("%s",bef);
printf("Enter data into node of DLL:");
if(checkpos(bef,0,0)=="NNN") strcpy(newnode->popo,bef);
else return;
newnode->next=newnode->pre=NULL; // optional
count++;
if(start==NULL){
    start=newnode;
    newnode->pre=NULL;
    end=newnode;
    newnode->next=NULL;
}
else{
    while(temp->next!=NULL){
        temp=temp->next;
    }
    newnode->pre=temp;
    temp->next=newnode;
    newnode->next=NULL;
    end=newnode;
}
scanf("%s",posi);
strcpy(newnode->post,posi);
if(checkpos(posi,1,0)=="NNN")strcpy(newnode->gone,"NNN");
else strcpy(newnode->gone,ret);
}
int letter(char b){
    if(b=='a') return 0;
    else if(b=='b')return 1;
    else if(b=='c')return 2;
    else if(b=='d')return 3;
    else if(b=='e')return 4;
    else if(b=='f')return 5;
    else if(b=='g')return 6;
    else if(b=='h')return 7;
```

```
  else return 8;
}
const char* checkpos(char lol[3],int k,int m){
  char n=lol[0];
  int l=lol[2]-'0';
  int v=letter(lol[1]);
  if(l==1){
    if((a[v]==n)&&(k==0)){
    a[v]='\0';
    return "NNN";}
    else if((art[v]==n)&&(m==0)){
      art[v]='\0';
      return "NNN";
    }
    else if(m==1){
      art[v]=n;
      return "NNN";
    }
    else if((k==1)){
      if(a[v]=='\0'){
      a[v]=n;
      return "NNN";}
      else{
        ret[0]=a[v];
        ret[1]=lol[1];
        ret[2]=lol[2];
        a[v]=n;
        return ret;
      }
    }
    else return "TTT";
  }
  else if(l==2){
    if((b[v]==n)&&(k==0)){
    b[v]='\0';
    return "NNN";}
    else if((brt[v]==n)&&(m==0)){
        brt[v]='\0';
        return "NNN";
    }
    else if(m==1){
        brt[v]=n;
        return "NNN";
    }
    else if((k==1)){
        if(b[v]=='\0'){
        b[v]=n;
        return "NNN";}
        else{
            ret[0]=b[v];
            ret[1]=lol[1];
            ret[2]=lol[2];
            b[v]=n;
            return ret;
        }
    }
    else return "TTT";
}
else if(l==3){
    if((c[v]==n)&&(k==0)){
    c[v]='\0';
    return "NNN";}
    else if((crt[v]==n)&&(m==0)){
        crt[v]='\0';
        return "NNN";
    }
    else if(m==1){
        crt[v]=n;
        return "NNN";
    }
    else if((k==1)){
        if(c[v]=='\0'){
```

```
        c[v]=n;
        return "NNN";}
        else{
            ret[0]=c[v];
            ret[1]=lol[1];
            ret[2]=lol[2];
            c[v]=n;
            return ret;
        }
    }
    else return "TTT";
}
else if(l==4){
    if((d[v]==n)&&(k==0)){
    d[v]='\0';
    return "NNN";}
    else if((drt[v]==n)&&(m==0)){
        drt[v]='\0';
        return "NNN";
    }
    else if(m==1){
        drt[v]=n;
        return "NNN";
    }
    else if((k==1)){
        if(d[v]=='\0'){
        d[v]=n;
        return "NNN";}
        else{
            ret[0]=d[v];
            ret[1]=lol[1];
            ret[2]=lol[2];
            d[v]=n;
            return ret;
        }
    }
```

```
   else return "TTT";
}
else if(l==5){
   if((e[v]==n)&&(k==0)){
   e[v]='\0';
   return "NNN";}
   else if((ert[v]==n)&&(m==0)){
      ert[v]='\0';
      return "NNN";
   }
   else if(m==1){
      ert[v]=n;
      return "NNN";
   }
   else if((k==1)){
      if(e[v]=='\0'){
      e[v]=n;
      return "NNN";}
      else{
         ret[0]=e[v];
         ret[1]=lol[1];
         ret[2]=lol[2];
         e[v]=n;
         return ret;
      }
   }
   else return "TTT";
}
else if(l==6){
   if((f[v]==n)&&(k==0)){
   f[v]='\0';
   return "NNN";}
   else if((frt[v]==n)&&(m==0)){
      frt[v]='\0';
      return "NNN";
   }
```

```
        else if(m==1){
            frt[v]=n;
            return "NNN";
        }
        else if((k==1)){
            if(f[v]=='\0'){
            f[v]=n;
            return "NNN";}
            else{
                ret[0]=f[v];
                ret[1]=lol[1];
                ret[2]=lol[2];
                f[v]=n;
                return ret;
            }
        }
        else return "TTT";
    }
    else if(l==7){
        if((g[v]==n)&&(k==0)){
        g[v]='\0';
        return "NNN";}
        else if((grt[v]==n)&&(m==0)){
            grt[v]='\0';
            return "NNN";
        }
        else if(m==1){
            grt[v]=n;
            printf("%c",grt[5]);
            return "NNN";
        }
        else if((k==1)){
            if(g[v]=='\0'){
            g[v]=n;
            return "NNN";}
            else{
```

```
            ret[0]=g[v];
            ret[1]=lol[1];
            ret[2]=lol[2];
            g[v]=n;
            return ret;
        }
    }
    else return "TTT";
}
else if(l==8){
    if((h[v]==n)&&(k==0)){
    h[v]='\0';
    return "NNN";}
    else if((hrt[v]==n)&&(m==0)){
        hrt[v]='\0';
        return "NNN";
    }
    else if(m==1){
        hrt[v]=n;
        return "NNN";
    }
    else if((k==1)){
        if(h[v]=='\0'){
        h[v]=n;
        return "NNN";}
        else{
            ret[0]=h[v];
            ret[1]=lol[1];
            ret[2]=lol[2];
            h[v]=n;
            return ret;
        }
    }
    else return "TTT";
}
else return "TTT";
```

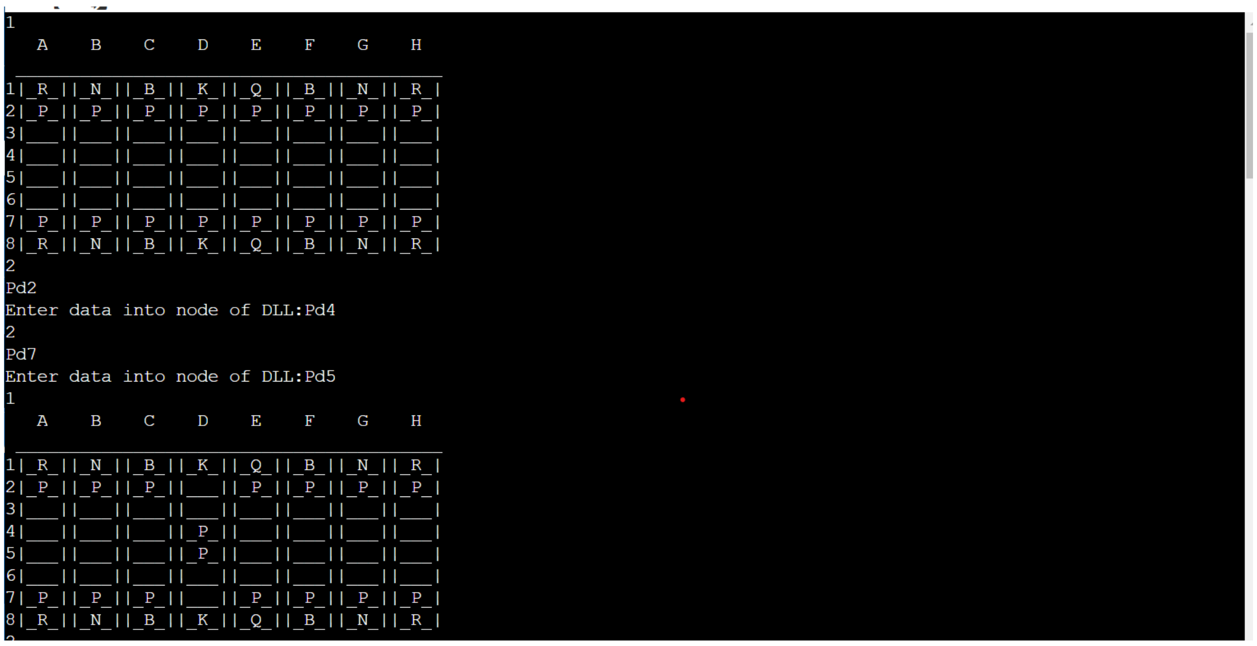
```
}
void movedis(chess* start){
    chess* temp = start;
    while (temp != NULL) {
        printf("pre: %c%c%c post:%c%c%c\n", temp->popo[0],temp->popo[1],temp->popo[2],
temp->post[0],temp->post[1],temp->post[2]);
        printf("gone:%s\n",temp->gone);
        temp = temp->next;}
}
void movedis2(chess* start){
int t=0;
chess* temp = start;
strcpy(art,"RNBKQBNR");
strcpy(brt,"PPPPPPPP");
strcpy(grt,"PPPPPPPP");
strcpy(hrt,"RNBKQBNR");
while(t!=1){

int i,m;
scanf("%d",&m);
char befo[3],afto[3],tempo[3],l[3];
if(m==0){
    befo[0]=temp->popo[0];
    befo[1]=temp->popo[1];
    befo[2]=temp->popo[2];
    afto[0]=temp->post[0];
    afto[1]=temp->post[1];
    afto[2]=temp->post[2];
    temp=temp->next;
}
if(m==1){
    temp=temp->pre;
    befo[0]=temp->post[0];
    befo[1]=temp->post[1];
    befo[2]=temp->post[2];
    afto[0]=temp->popo[0];
    afto[1]=temp->popo[1];
    afto[2]=temp->popo[2];
    tempo[0]=temp->gone[0];
    tempo[1]=temp->gone[1];
    tempo[2]=temp->gone[2];
    char blo[3];
    printf("%c%c",afto[2],afto[1]);
    if(tempo[0]!='N')
    strcpy(blo,checkpos(tempo,2,1));
}
if(m==2){
    t=1;
    break;
}
int aw;
if(checkpos(befo,2,0)=="NNN") aw=0;
else aw=1;
if(checkpos(afto,2,1)=="NNN") aw=0;
else aw=1;
    printf("   A    B    C    D    E    F    G    H\n");
    printf(" _________________________________________\n");
    printf("1");
    for(i=0;i<8;i++){
        if(art[i]!='\0')
        printf("|_%c_|",art[i]);
        else printf("|___|");
    }
    printf("\n");
    printf("2");
    for(i=0;i<8;i++){
        if(brt[i]!='\0')
        printf("|_%c_|",brt[i]);
        else printf("|___|");
    }
    printf("\n");
    printf("3");
```

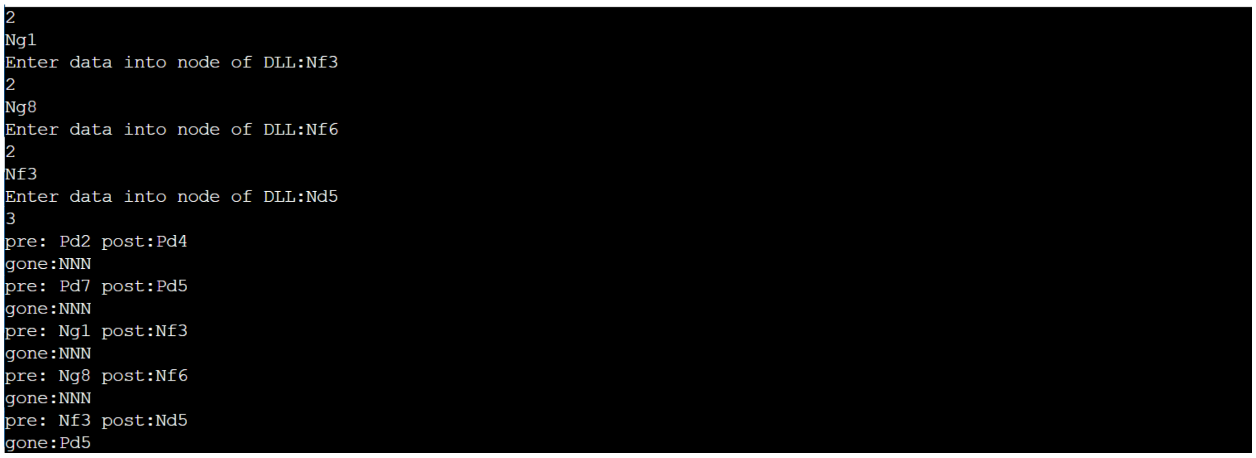
```
for(i=0;i<8;i++){
   if(crt[i]!='\0')
   printf("|_%c_|",crt[i]);
   else printf("|___|");
}
printf("\n");
printf("4");
for(i=0;i<8;i++){
   if(drt[i]!='\0')
   printf("|_%c_|",drt[i]);
   else printf("|___|");
}
printf("\n");
printf("5");
for(i=0;i<8;i++){
   if(ert[i]!='\0')
   printf("|_%c_|",ert[i]);
   else printf("|___|");
}
printf("\n");
printf("6");
for(i=0;i<8;i++){
   if(frt[i]!='\0')
   printf("|_%c_|",frt[i]);
   else printf("|___|");
}
printf("\n");
printf("7");
for(i=0;i<8;i++){
   if(grt[i]!='\0')
   printf("|_%c_|",grt[i]);
   else printf("|___|");
}
printf("\n");
printf("8");
for(i=0;i<8;i++){
        if(hrt[i]!='\0')
        printf("|_%c_|",hrt[i]);
        else printf("|__|");
    }
    printf("\n");
    //t=1;

}
}
void chessstart(){
    int i;
    printf("   A    B    C    D    E    F    G    H\n");
    printf(" ________________________________\n");
    printf("1");
    for(i=0;i<8;i++){
        if(a[i]!='\0')
        printf("|_%c_|",a[i]);
        else printf("|__|");
    }
    printf("\n");
    printf("2");
    for(i=0;i<8;i++){
        if(b[i]!='\0')
        printf("|_%c_|",b[i]);
        else printf("|__|");
    }
    printf("\n");
    printf("3");
    for(i=0;i<8;i++){
        if(c[i]!='\0')
        printf("|_%c_|",c[i]);
        else printf("|__|");
    }
    printf("\n");
    printf("4");
    for(i=0;i<8;i++){
        if(d[i]!='\0')
        printf("|_%c_|",d[i]);
        else printf("|___|");
    }
    printf("\n");
    printf("5");
    for(i=0;i<8;i++){
        if(e[i]!='\0')
        printf("|_%c_|",e[i]);
        else printf("|___|");
    }
    printf("\n");
    printf("6");
    for(i=0;i<8;i++){
        if(f[i]!='\0')
        printf("|_%c_|",f[i]);
        else printf("|___|");
    }
    printf("\n");
    printf("7");
    for(i=0;i<8;i++){
        if(g[i]!='\0')
        printf("|_%c_|",g[i]);
        else printf("|___|");
    }
    printf("\n");
    printf("8");
    for(i=0;i<8;i++){
        if(h[i]!='\0')
        printf("|_%c_|",h[i]);
        else printf("|___|");
    }
    printf("\n");
}
```

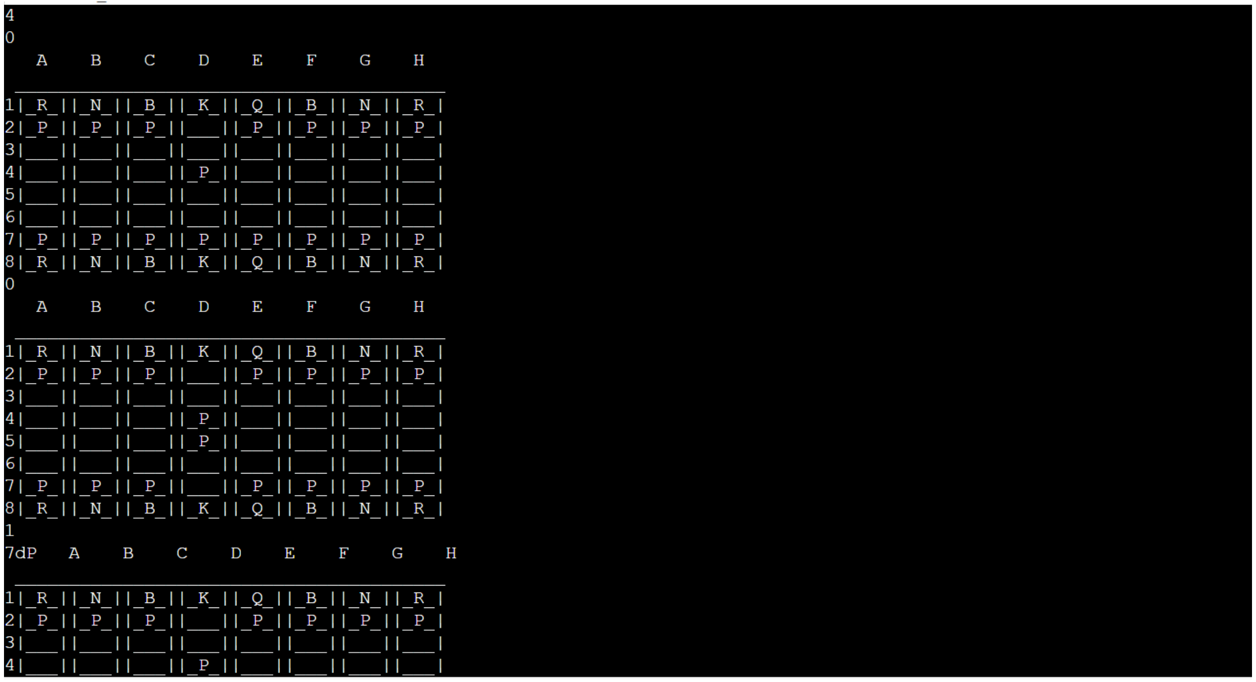
OUTPUT



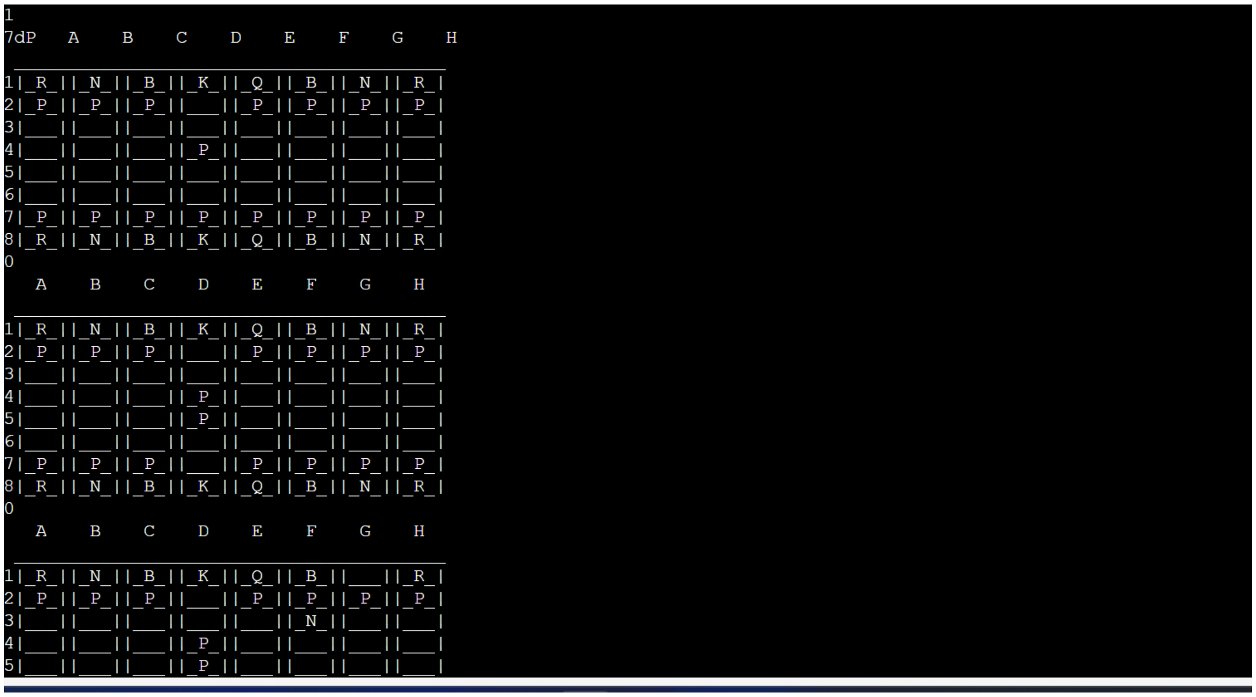
```
1
   A    B    C    D    E    F    G    H
 ________________________________________
1|_R_||_N_||_B_||_K_||_Q_||_B_||_N_||_R_|
2|_P_||_P_||_P_||_P_||_P_||_P_||_P_||_P_|
3|___||___||___||___||___||___||___||___|
4|___||___||___||___||___||___||___||___|
5|___||___||___||___||___||___||___||___|
6|___||___||___||___||___||___||___||___|
7|_P_||_P_||_P_||_P_||_P_||_P_||_P_||_P_|
8|_R_||_N_||_B_||_K_||_Q_||_B_||_N_||_R_|
2
Pd2
Enter data into node of DLL:Pd4
2
Pd7
Enter data into node of DLL:Pd5
1
   A    B    C    D    E    F    G    H
 ________________________________________
1|_R_||_N_||_B_||_K_||_Q_||_B_||_N_||_R_|
2|_P_||_P_||_P_||___||_P_||_P_||_P_||_P_|
3|___||___||___||___||___||___||___||___|
4|___||___||___||_P_||___||___||___||___|
5|___||___||___||_P_||___||___||___||___|
6|___||___||___||___||___||___||___||___|
7|_P_||_P_||_P_||___||_P_||_P_||_P_||_P_|
8|_R_||_N_||_B_||_K_||_Q_||_B_||_N_||_R_|
```



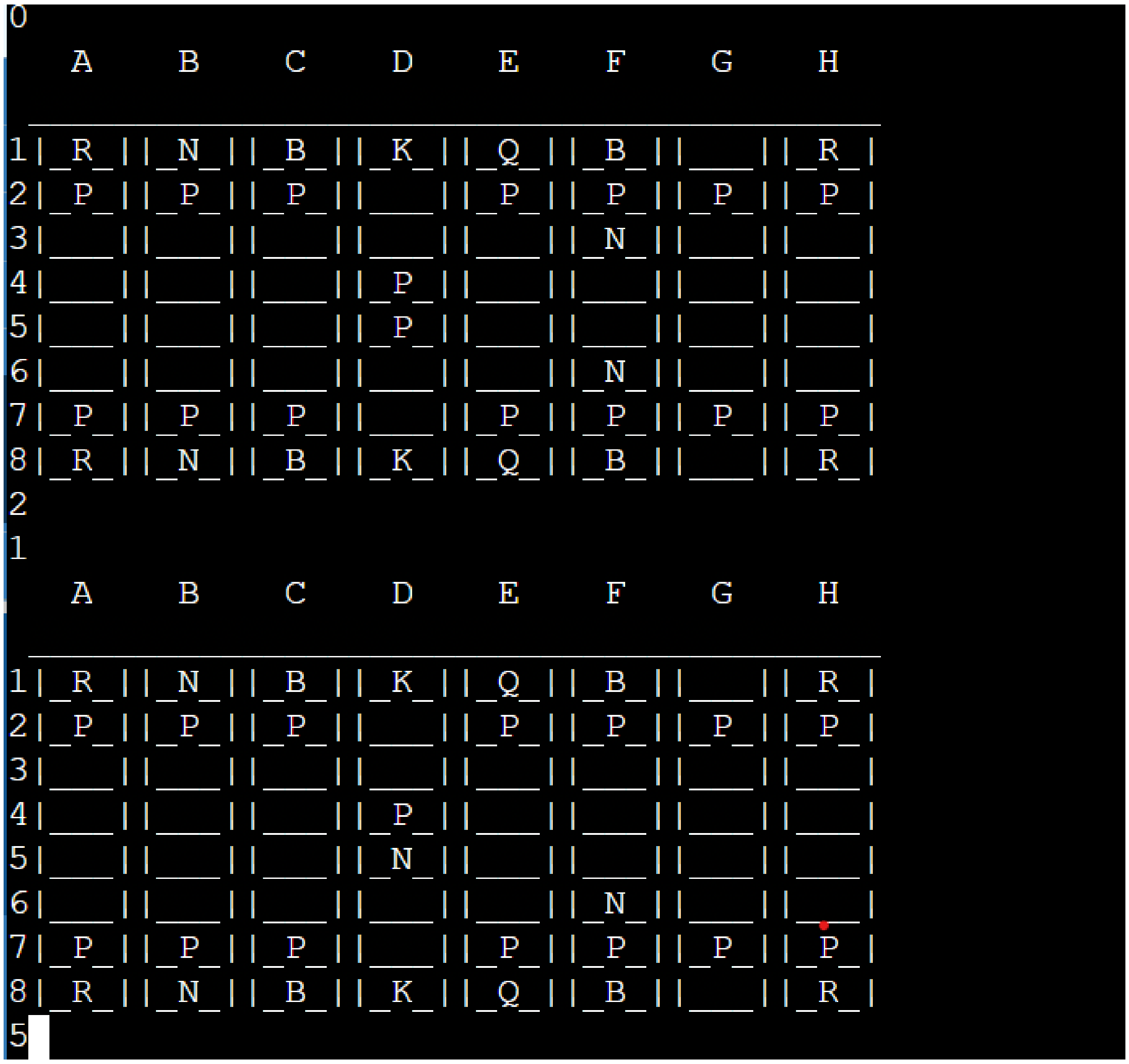
```
2
Ng1
Enter data into node of DLL:Nf3
2
Ng8
Enter data into node of DLL:Nf6
2
Nf3
Enter data into node of DLL:Nd5
3
pre: Pd2 post:Pd4
gone:NNN
pre: Pd7 post:Pd5
gone:NNN
pre: Ng1 post:Nf3
gone:NNN
pre: Ng8 post:Nf6
gone:NNN
pre: Nf3 post:Nd5
gone:Pd5
```



```
4
0
   A    B    C    D    E    F    G    H
 ________________________________________
1|_R_||_N_||_B_||_K_||_Q_||_B_||_N_||_R_|
2|_P_||_P_||_P_||___||_P_||_P_||_P_||_P_|
3|___||___||___||___||___||___||___||___|
4|___||___||___||_P_||___||___||___||___|
5|___||___||___||___||___||___||___||___|
6|___||___||___||___||___||___||___||___|
7|_P_||_P_||_P_||_P_||_P_||_P_||_P_||_P_|
8|_R_||_N_||_B_||_K_||_Q_||_B_||_N_||_R_|
0
   A    B    C    D    E    F    G    H
 ________________________________________
1|_R_||_N_||_B_||_K_||_Q_||_B_||_N_||_R_|
2|_P_||_P_||_P_||___||_P_||_P_||_P_||_P_|
3|___||___||___||___||___||___||___||___|
4|___||___||___||_P_||___||___||___||___|
5|___||___||___||_P_||___||___||___||___|
6|___||___||___||___||___||___||___||___|
7|_P_||_P_||_P_||___||_P_||_P_||_P_||_P_|
8|_R_||_N_||_B_||_K_||_Q_||_B_||_N_||_R_|
1
7dP   A    B    C    D    E    F    G    H
 ________________________________________
1|_R_||_N_||_B_||_K_||_Q_||_B_||_N_||_R_|
2|_P_||_P_||_P_||___||_P_||_P_||_P_||_P_|
3|___||___||___||___||___||___||___||___|
4|___||___||___||_P_||___||___||___||___|
```



```
1
7dP   A    B    C    D    E    F    G    H
 ________________________________________
1|_R_||_N_||_B_||_K_||_Q_||_B_||_N_||_R_|
2|_P_||_P_||_P_||___||_P_||_P_||_P_||_P_|
3|___||___||___||___||___||___||___||___|
4|___||___||___||_P_||___||___||___||___|
5|___||___||___||___||___||___||___||___|
6|___||___||___||___||___||___||___||___|
7|_P_||_P_||_P_||_P_||_P_||_P_||_P_||_P_|
8|_R_||_N_||_B_||_K_||_Q_||_B_||_N_||_R_|
0
   A    B    C    D    E    F    G    H
 ________________________________________
1|_R_||_N_||_B_||_K_||_Q_||_B_||_N_||_R_|
2|_P_||_P_||_P_||___||_P_||_P_||_P_||_P_|
3|___||___||___||___||___||___||___||___|
4|___||___||___||_P_||___||___||___||___|
5|___||___||___||_P_||___||___||___||___|
6|___||___||___||___||___||___||___||___|
7|_P_||_P_||_P_||___||_P_||_P_||_P_||_P_|
8|_R_||_N_||_B_||_K_||_Q_||_B_||_N_||_R_|
0
   A    B    C    D    E    F    G    H
 ________________________________________
1|_R_||_N_||_B_||_K_||_Q_||_B_||___||_R_|
2|_P_||_P_||_P_||___||_P_||_P_||_P_||_P_|
3|___||___||___||___||___||_N_||___||___|
4|___||___||___||_P_||___||___||___||___|
5|___||___||___||_P_||___||___||___||___|
```



```
0
   A    B    C    D    E    F    G    H

  ________________________________________
1|_R_||_N_||_B_||_K_||_Q_||_B_||___||_R_|
2|_P_||_P_||_P_||___||_P_||_P_||_P_||_P_|
3|___||___||___||___||___||_N_||___||___|
4|___||___||___||_P_||___||___||___||___|
5|___||___||___||_P_||___||___||___||___|
6|___||___||___||___||___||_N_||___||___|
7|_P_||_P_||_P_||___||_P_||_P_||_P_||_P_|
8|_R_||_N_||_B_||_K_||_Q_||_B_||___||_R_|
2
1
   A    B    C    D    E    F    G    H

  ________________________________________
1|_R_||_N_||_B_||_K_||_Q_||_B_||___||_R_|
2|_P_||_P_||_P_||___||_P_||_P_||_P_||_P_|
3|___||___||___||___||___||___||___||___|
4|___||___||___||_P_||___||___||___||___|
5|___||___||___||_N_||___||___||___||___|
6|___||___||___||___||___||_N_||___||___|
7|_P_||_P_||_P_||___||_P_||_P_||_P_||_P_|
8|_R_||_N_||_B_||_K_||_Q_||_B_||___||_R_|
5
```

Limitations

The current implementation of the chess game has some limitations:

- It lacks a full set of chess rules and does not enforce legal moves.
- It does not handle various special moves (e.g., castling, en passant).
- There is no checkmate or game end detection.
- The user input handling is basic and does not provide clear instructions or error messages.
- There is no distinction between White and Black Pieces.

- While traversal, we don't know the previous position of the moved piece.

Improvements

To make the chess game more complete and user-friendly, we can consider the following improvements:

- Implement proper chess move validation and enforce legal moves.
- Implement special moves like castling ,en passant and add checkmate detection and end-of-game handling.
- Improve user input handling with clear instructions and error messages.
- Implement a better graphical representation of the chessboard for better visualization.

CONCLUSION

This program successfully employs the chess playing setup in C and lets the user functionally play the game. It efficiently tries to insert moves into the Doubly Linked List and Arrays after checking their validity. Then it allows the user to freely traverse to the next or previous move that happened already in the game and displays it to him.

This program successfully digitalises chess. It acts as a mediator between ML programs in order to receive data from machine learning algorithms that compute the optimal move and display it to the human in order for him to analyze and learn from it.

Bibliography

- DATA STRUCTURES THEORY MATERIAL- GRIET Textbook
- www.google.com
- www.onlinegdb.com
- www.docs.google.com
- https://en.wikipedia.org/wiki/Chess_notation#:~:text=Chess%20notation%20systems%20are%20used,record%20of%20an%20ongoing%20game.
- https://en.wikipedia.org/wiki/Algebraic_notation_(chess)
- https://en.wikipedia.org/wiki/Chess_engine
- https://en.wikipedia.org/wiki/Online_chess#:~:text=Online%20chess%20is%20chess%20that,or%20similar%20chess%20rating%20system.
- https://en.wikipedia.org/wiki/History_of_chess_engines#:~:text=Finally%2C%20in%201957%20an%20IBM,real%20beginning%20of%20chess%20computing.
- https://www.youtube.com/watch?v=CFkhUajb8c8