KONGU ENGINEERING COLLEGE
(Autonomous)
Perundurai, Erode – 638 060
DEPARTMENT OF COMPUTER APPLICATIONS

CHESS GAME USING GRAPH DATA STRUCTURE

AN APPLICATION PROJECT REPORT

for

ADVANCED DATA STRUCTURES AND ALGORITHMS LABORATORY (22MCL11)

Submitted by

S. DEEPANCHAKRAVARTHI (22MCR014)
K. KAVINKUMAR (22MCR044)
N. KAVIN RAJ (22MCR045)

KONGU ENGINEERING COLLEGE

(Autonomous)

Perundurai, Erode – 638 060

DEPARTMENT OF COMPUTER APPLICATIONS

BONAFIDE CERTIFICATE

Name : **S. DEEPANCHAKRAVARTHI (22MCR014)**

K. KAVINKUMAR (22MCR044)

N. KAVIN RAJ (22MCR045)

Course Code : **22MCL11**

Course Name : **ADVANCED DATA STRUCTURES AND ALGORITHMS LABORATORY**

Semester : **I**

Certified that this is a bonafide record of work for application project done by the above students for **22MCL11 – ADVANCED DATA STRUCTURES AND ALGORITHMS LABORATORY** during the academic year **2022-2023.**

Submitted for the Viva Voce Examination held on _____

Lab-in-Charge **Head of the Department**

INDEX

ABSTRACT

The aim of this Chess Game application is to implement a completely functional chess program. This application is a text-based game. It utilizes standard chess notation for movements. The Chess game follows the all rules of chess, and all the pieces only move according to the player's movement.

The ancient game of chess is played by two players, where two identical sets of pieces batttle each other. Each set consists of eight pawns, two rooks, two knights, two bishops, one queen, and one king. The initial configuration of the chess board is composed of an 8x8 grid of 64 equal squares alternately light or white squares and dark or black squares. The chessboard is placed between the players in such a way that the near comer square to the right of the player is white. The position of each piece is represented by a letter (a-k) and a number (1-8).

The Chess Game application code started by initializing the chess board. By making 8 lists of 8 elements inside a larger list, the application was effectively able to simulate an 8x8 chess board and Unicode characters are used for displaying of the actual chess pieces.

INTRODUCTION

Playing games is fun and exciting. It gives us relief from stress and unwinds from our stressful work. Many of us spend our free time or others that use most of their time in playing and exploring new games. Today, with the rapid development of technology we have, games that are rising together with it. Nowadays with technology users have many games that are developed for computers specifically for windows. With the high technology equipped with these computer games become robust and attract many people to buy or have this gadget for them to experience what is inside it which makes it a trend for the new generation of gadget.

Chess game is one of the oldest and most popular board games. It is played by two opponents on a checkered board with specially designed pieces of contrasting colors, commonly white and black. The objective of the game is to capture the opponent's king.

Chess is played on a board of 64 squares arranged in eight vertical rows called files and eight horizontal rows called ranks. These squares alternate between two colors: one light, such as white, beige, or yellow; and the other dark, such as black or green. The board is set between the two opponents so that each player has a light-colored square at the right-hand corner.

The aim of the game is to capture the opponent’s king. In this chess game, white piece moves first, after which the players alternate turns in accordance with fixed rules, each player attempting to force the opponent’s principal piece, the King, into checkmate—a position where it is unable to avoid capture.

MODULE DESCRIPTION

A module description offers detailed information that is accessible in a range of ways about the module and the components it supports. The project contains the following modules.

- Game instruction
- Movements
- Score board

GAME INSTRUCTION

In this module, it shows the game instructions. It contains the instruction about how to play the game and basic rules of the chess game. In this instruction module, each piece in the chess game is described and movement of the piece is also included.

MOVEMENTS

In this module, movement of each piece is made with keyboard by typing the position of the piece and the position where the piece needs to be placed. The position of the chess piece is denoted by algebraic notation. Algebraic notation (or AN) is the standard method for recording and describing the moves in a game of chess. Also called standard notation, it is based on coordinate notation, a system of coordinates to uniquely identify each square on the chessboard.

SCORE BOARD

In this module, the score of each move of the chess piece by the player is displayed. A pawn is worth one point, a knight or bishop is worth three points, a rook is worth five points and a queen is worth nine points. The king is the only piece that does not have a point value.

FLOW CHART

A flowchart is a picture of the separate steps of a process in sequential order. It is a generic tool that can be adapted for a wide variety of purposes, and can be used to describe various processes, such as a manufacturing process, an administrative or service process, or a project plan.

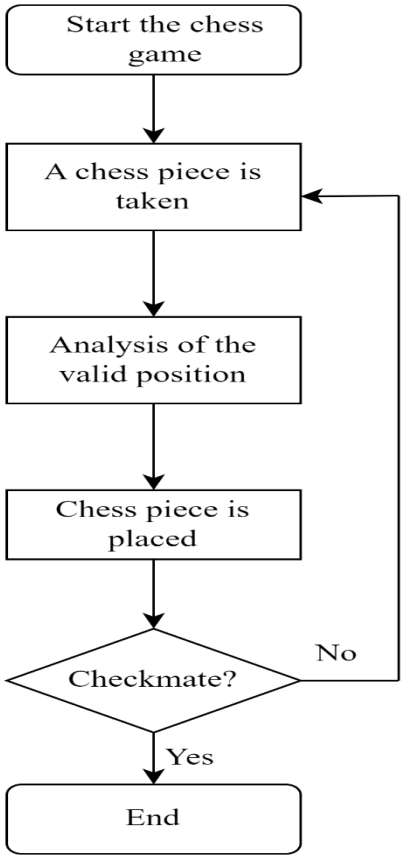

Figure 4.1 Flow chart

In figure 4.1 it represents the flow chart for the chess game. In this flow chart when the game starts, player analysis valid position for placing the chess piece. If the player's king is captured by opponent then the player meets checkmate and game ends. Else, game continues.

SOURCE CODE

```
#include <stdio.h>
#include <string.h>
int board[128] = {              // 0x88 board + positional scores
22, 20, 21, 23, 19, 21, 20, 22,    0,   0,   5,   5,   0,   0,   5,  0,
18, 18, 18, 18, 18, 18, 18, 18,    5,   5,   0,   0,   0,   0,   5,  5,
  0,   0,   0,   0,   0,   0,   0,   0,    5, 10, 15, 20, 20, 15, 10,  5,
  0,   0,   0,   0,   0,   0,   0,   0,    5, 10, 20, 30, 30, 20, 10,  5,
  0,   0,   0,   0,   0,   0,   0,   0,    5, 10, 20, 30, 30, 20, 10,  5,
  0,   0,   0,   0,   0,   0,   0,   0,    5, 10, 15, 20, 20, 15, 10,  5,
  9,   9,   9,   9,   9,   9,   9,   9,    5,   5,   0,   0,   0,   0,   5,  5,
14, 12, 13, 15, 11, 13, 12, 14,    0,   0,   5,   5,   0,   0,   5,  0
};
char *notation[] = {          // convert square id to board notation

   "a8", "b8", "c8", "d8", "e8", "f8", "g8", "h8",    "i8","j8","k8","l8","m8","n8","o8", "p8",
   "a7", "b7", "c7", "d7", "e7", "f7", "g7", "h7",    "i7","j7","k7","l7","m7","n7","o7", "p7",
   "a6", "b6", "c6", "d6", "e6", "f6", "g6", "h6",    "i6","j6","k6","l6","m6","n6","o6", "p6",
   "a5", "b5", "c5", "d5", "e5", "f5", "g5", "h5",    "i5","j5","k5","l5","m5","n5","o5", "p5",
   "a4", "b4", "c4", "d4", "e4", "f4", "g4", "h4",    "i4","j4","k4","l4","m4","n4","o4", "p4",
   "a3", "b3", "c3", "d3", "e3", "f3", "g3", "h3",    "i3","j3","k3","l3","m3","n3","o3", "p3",
   "a2", "b2", "c2", "d2", "e2", "f2", "g2", "h2",    "i2","j2","k2","l2","m2","n2","o2", "p2",
   "a1", "b1", "c1", "d1", "e1", "f1", "g1", "h1",    "i1","j1","k1","l1","m1","n1","o1", "p1",
};
char *pieces[] = {

                        ".", "-", "\u265F", "\u265A", "\u265E", "\u265D", "\u265C", "\u265B",
                        "-", "\u2659", "-", "\u2654", "\u2658", "\u2657", "\u2656", "\u2655",
};
```

```
enum { WHITE = 8, BLACK = 16};    // side to move
static int move_offsets[] = {
   15,  16,  17,   0,
  -15, -16, -17,   0,
    1,  16,  -1, -16,   0,
    1,  16,  -1, -16,  15, -15, 17, -17,  0,
   14, -14,  18, -18,  31, -31, 33, -33,  0,
    3,  -1,  12,  21,  16,   7, 12
};
int piece_weights[] = { 0, 0, -100, 0, -300, -350, -500, -900, 0, 100, 0, 0, 300, 350, 500, 900
};
int best_src, best_dst;    // to store the best move found in search
void PrintBoard()
{
   for(int sq = 0; sq < 128; sq++)
   {
      if(!(sq % 16)) printf(" %d  ", 8 - (sq / 16));
      //printf(" %c", ((sq & 8) && (sq += 7)) ? '\n' : pieces[board[sq] & 15]);     // ASCII
pieces
      printf(" %s", ((sq & 8) && (sq += 7)) ? "\n" : pieces[board[sq] & 15]);     // unicode
pieces
   }
   printf("\n     a b c d e f g h\n\nYour move: \n");
}
int SearchPosition(int side, int depth, int alpha, int beta)
{
   if(!depth)
   {
      int mat_score = 0, pos_score = 0, pce, eval = 0;

      for(int sq = 0; sq < 128; sq++)
```

```
        {
            if(!(sq & 0x88))
            {
                if(pce = board[sq])
                {
                    mat_score += piece_weights[pce & 15]; // material score
                    (pce & 8) ? (pos_score += board[sq + 8]) : (pos_score -= board[sq + 8]); //
positional score
                }
            }
        }

        eval = mat_score + pos_score;

        return (side == 8) ? eval : -eval;   // here returns current position's score
    }
    int old_alpha = alpha;
    int temp_src;
    int temp_dst;
    int score = -10000;
    int piece, type, directions, dst_square, captured_square, captured_piece, step_vector;
    for(int src_square = 0; src_square < 128; src_square++)
    {
        if(!(src_square & 0x88))
        {
            piece = board[src_square];

            if(piece & side)
            {
                type = piece & 7;
                directions = move_offsets[type + 30];
```

```
while(step_vector = move_offsets[++directions])
{
   dst_square = src_square;
   do
   {
      dst_square += step_vector;
      captured_square = dst_square;
      if(dst_square & 0x88) break;
      captured_piece = board[captured_square];
      if(captured_piece & side) break;
      if(type < 3 && !(step_vector & 7) != !captured_piece) break;
      if((captured_piece & 7) == 3) return 10000;    // on king capture
      board[captured_square] = 0;
      board[src_square] = 0;
      board[dst_square] = piece;

      // pawn promotion
      if(type < 3)
      {
         if(dst_square + step_vector + 1 & 0x80)
            board[dst_square]|=7;
      }
      score = -SearchPosition(24 - side, depth - 1, -beta, -alpha);
      board[dst_square] = 0;
      board[src_square] = piece;
      board[captured_square] = captured_piece;
      best_src = src_square;
      best_dst = dst_square;
      if(score > alpha)
      {
         if(score >= beta)
```

```
                    return beta;
                    alpha = score;
                    temp_src = src_square;
                    temp_dst = dst_square;
                }
                captured_piece += type < 5;
                if(type < 3 & 6*side + (dst_square & 0x70) == 0x80)captured_piece--;
            }
            while(!captured_piece);
        }
    }
  }
  if(alpha != old_alpha)
  {
    best_src = temp_src;
    best_dst = temp_dst;
  }
  return alpha;
}
int main()
{
  char user_move[5];
  int depth = 3;
  int side = WHITE;
  printf("-------------------INSTRUCTION FOR THE GAME-------------\n");
  printf("1.Rooks move in a continuous line forwards, backwards and side-to-side.\n2.Bishops move in continuous diagonal lines in any direction.\n3.The queen moves in continuous diagonal and straight lines. Forward, backward and side-to-side.\n4.The king can move in any direction, one square at a time.\n5.Pawns only move forward.\n6 Knights are the only pieces that jump off the board by 3 steps.\n7.You should make moves with
```

```
keyboard by typing the position of the piece.\n");
   printf("-------------------------------------------------------------------------------------------\n");
   PrintBoard();
   while(1)  // play vs computer
   {
      memset(&user_move[0], 0, sizeof(user_move));
      if(!fgets(user_move, 5, stdin)) continue;
      if(user_move[0] == '\n') continue;
      int user_src, user_dst;
      for(int sq = 0; sq < 128; sq++)
      {
         if(!(sq & 0x88))
         {
            if(!strncmp(user_move, notation[sq], 2))
               user_src = sq;

            if(!strncmp(user_move + 2, notation[sq], 2))
               user_dst = sq;
         }
      }
      board[user_dst] = board[user_src];
      board[user_src] = 0;
      if(((board[user_dst] == 9) && (user_dst >= 0 && user_dst <= 7)) ||
        ((board[user_dst] == 18) && (user_dst >= 112 && user_dst <= 119)))
         board[user_dst] |= 7;
      PrintBoard();
      side = 24 - side;   // change side
      int score = SearchPosition(side, depth, -10000, 10000);
      board[best_dst] = board[best_src];
      board[best_src] = 0;
      if(((board[best_dst] == 9) && (best_dst >= 0 && best_dst <= 7)) ||
```

```
        ((board[best_dst] == 18) && (best_dst >= 112 && best_dst <= 119)))
         board[best_dst] |= 7;
      side = 24 - side;    // change side
      PrintBoard();
      printf("\nscore: '%d'\n", score);
      if(score == 10000 || score == -10000) {printf("Checkmate!\n"); break;}
      printf("best move: '%s%s'\n", notation[best_src], notation[best_dst]);
   }
   return 0;
}
```

SCREENSHOTS

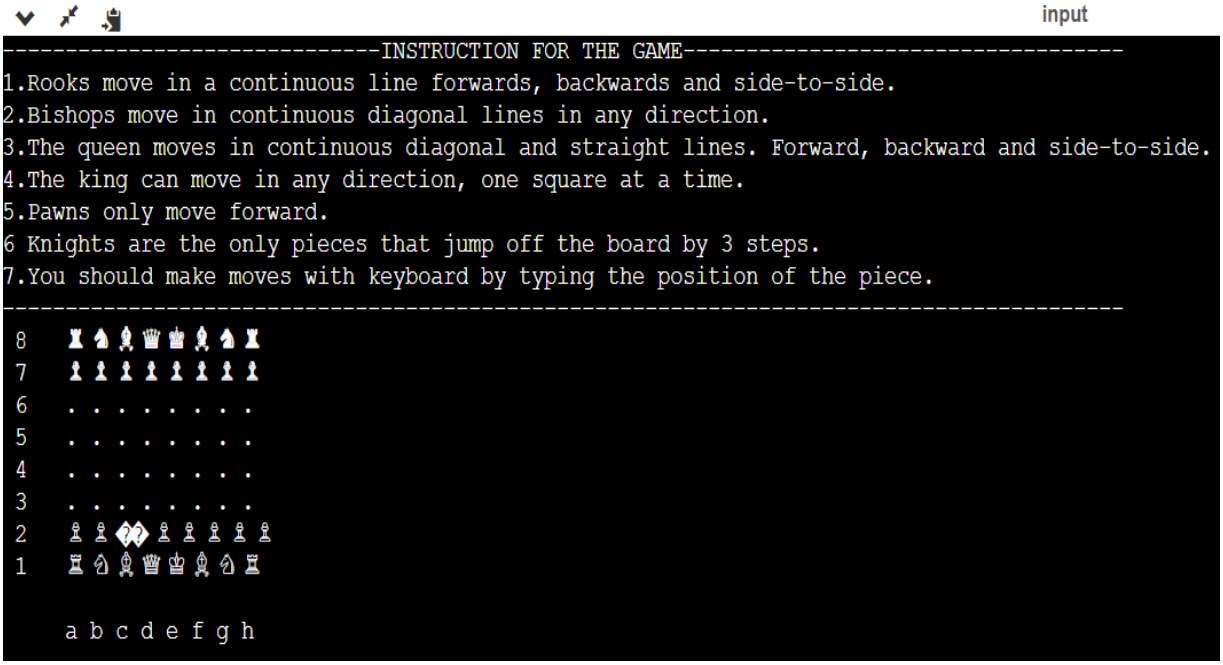

Figure 6.1 Instruction

In figure 6.1 the page displays the instructions of the chess game. The rules followed in the chess game and the movement of each chess piece is also added to the instruction.

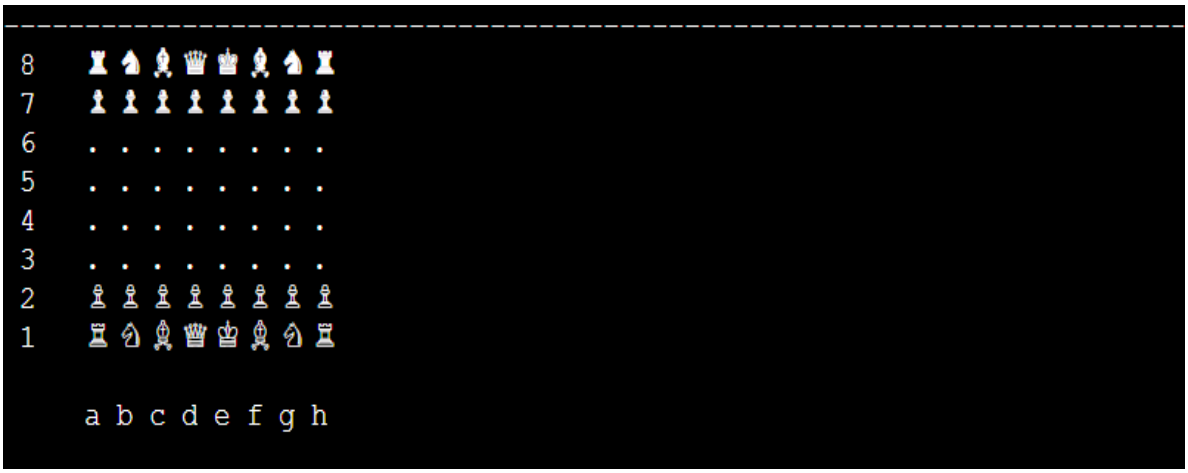

Figure 6.2 Chess Board

In figure 6.2 the page contains the Chess board with algebraic notations and board contains 16 chess pieces of two sets. The algebraic notation contains alphabets from a to h and numbers from 1 to 8.

Figure 6.3 Player move

In figure 6.3 the player's move is displayed. In this page, player moved a pawn which is in the position of c2 and placed it in the position c3.

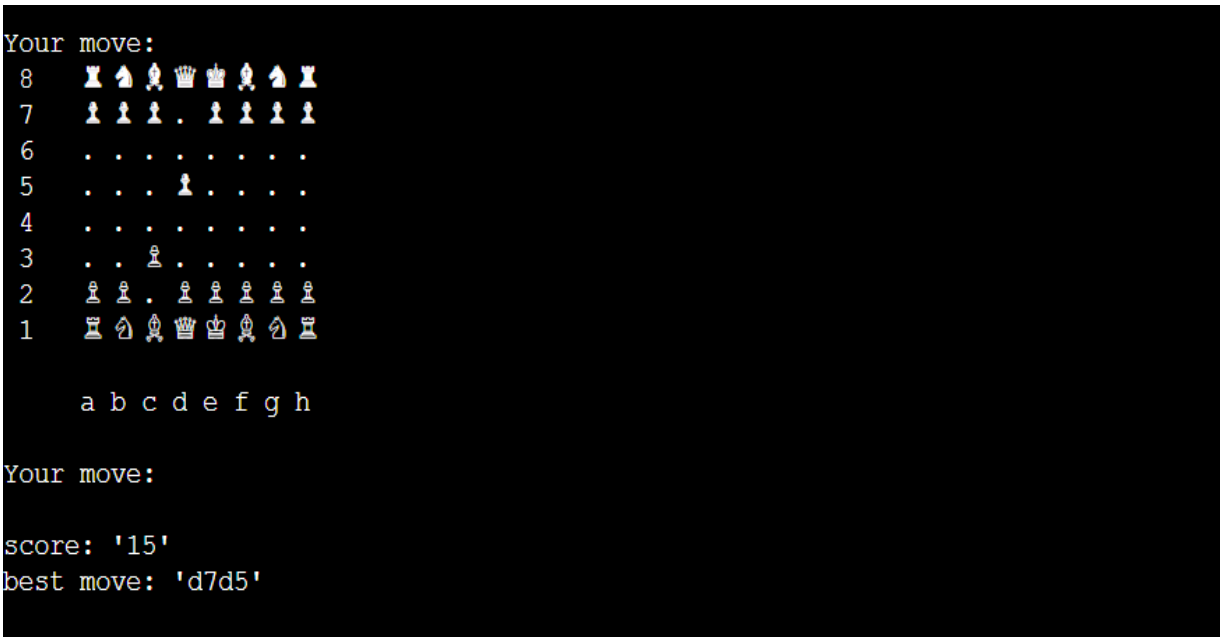

Figure 6.4 System move

In Figure 6.4 the system's move is displayed. In this page, a pawn in the position of d7 is placed in the position d5.

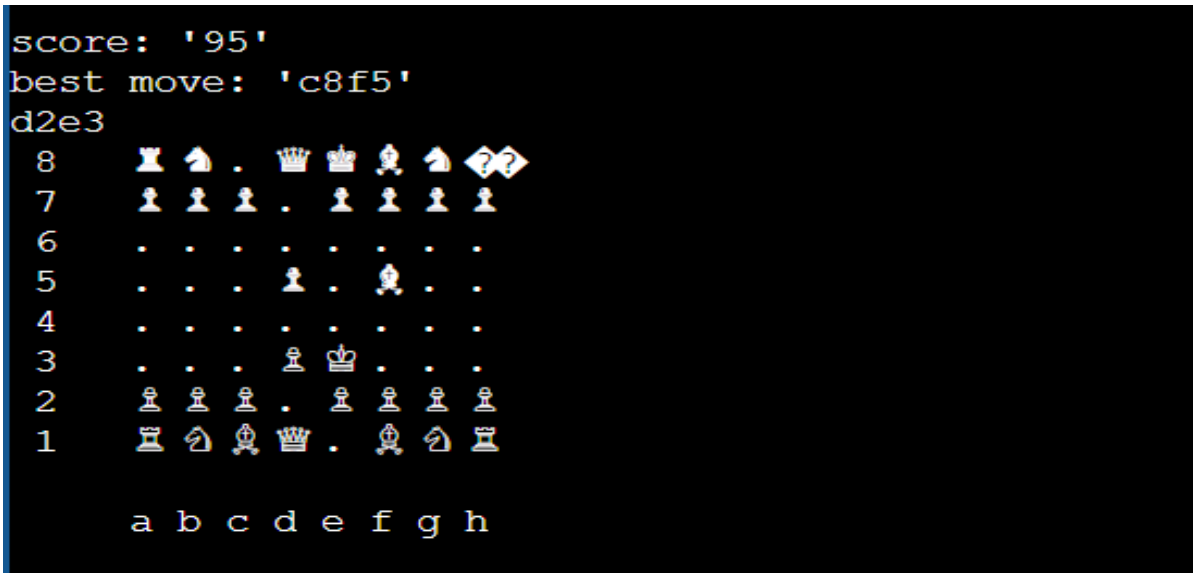

Figure 6.5 Score

In figure 6.5 the Score for each move of a chess piece is displayed. The score gets changed for each moves according to the chess piece and its move.

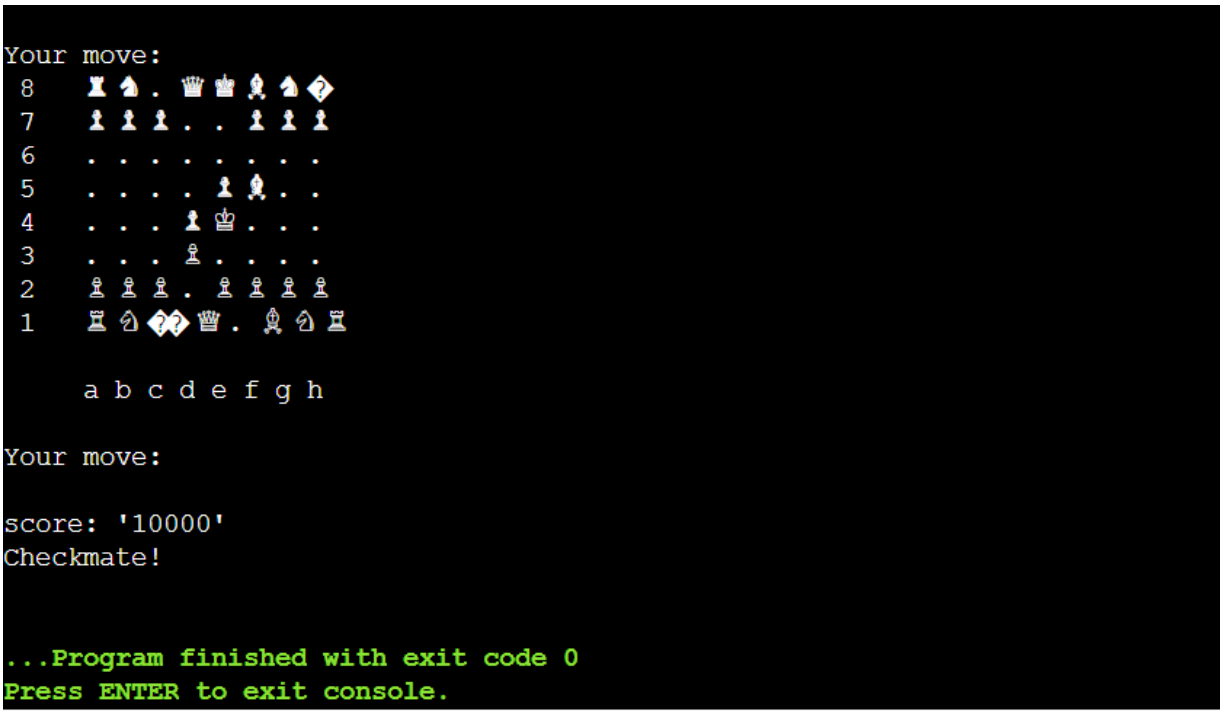

Figure 6.6 Checkmate

In figure 6.6 the page displays Checkmate where the player's king is captured by the opponent and there are no more moves for the player and the game gets end and will be displayed as checkmate.

CONCLUSION

A game of chess ends when a player puts the opposing player's king in a position that cannot avoid capture (checkmate). A game can also be won or lose through concession. A chess match can also end in a draw. This can happen through stalemate, mutual consent, checkmate being impossible to achieve, and in other ways.

The Chess Game is a simple console application with very simple graphics. In this project, the player can play the popular chess game just like played it elsewhere. Player can be able to play using the keyboards for algebraic notation movements of the pieces in the chess board. Whenever the move is done by the player the score is displayed.

REFERENCES

[1]. R.S.Salaria, “Data structures & Algorithms Using C”, 5th Edition, Khanna Book Publishing Co.Pvt. Ltd., SRS Enterprises, New Delhi, 2022.

[2]. ReemaThareja, “Data Structures using C”, 2nd Edition, Oxford University Press, NewDelhi, 2018.

[3]. Jean Paul Tremblay and Paul G. Sorensen, “An Introduction to Data Structures with Applications”, 2nd Edition, Tata McGraw Hill, New Delhi, 2017.

[4]. https://www.edureka.co/blog/c-data-structures/

[5]. https://www.javatpoint.com/data-structure-tutorial/

[6]. https://www.programiz.com/dsa/

[7]. https://www.mygreatlearning.com/blog/data-structures-using-c/