ASCII CHESS  
CLASS 12TH CBSE PROJECT

DHRUV DEVASTHALE, CLASS: XII-A

CERTIFICATE

This is to certify that this is the original work of Dhruv Devasthale of class X11-A of programming the software ASCII CHESS in the partial fulfilment of computer science practical examination conducted by CBSE under the guidance of my computer science teacher Sujata Bhardwaj during the year 2018-19 by Central Board of Secondary Education, New Delhi.

ACKNOWLEDGEMENT

I Dhruv Devasthale of class XII-A express my sincere gratitude towards our principal Mr. A.K. Sharma for giving me this golden opportunity to program the software ASCII CHESS. I would also like to thank my computer science teacher Sujata Bhardwaj for helping and guiding me on this project.

***INDEX***

|  |  |  |
| --- | --- | --- |
| **S.NO** | **TOPIC** | **PAGE NO.** |
| 1 | Aim | 4 |
| 2 | Introduction | 4 |
| 3 | Minimum System Requirements & My System Specifications | 5 |
| 4 | Header, Binary and Text Files Used | 6 |
| 4 | Classes Used | 7 |
| 5 | Logical Design | 8 |
| 6 | CODE | 9-48 |
| 8 | Output | 49-61 |
| 9 | Scope for Improvement and References | 16 |

AIM

To create a simple chess game with ASCII characters in C++.

INTRODUCTION

* The chess pieces are represented as follows:

-R: Rook

-K: King

-B: Bishop

-Queen

-Pawan

-Norse/Knight

* White chess pieces (player 1) are represented by capital letters while Black chess pieces (player 2) are represented by lowercase letters.
* The symbols are stored into 8x8 two-dimensional character array. (boardSymbols(char))
* To control a chess piece key presses are taken as inputs (w, a, s, d, esc, space).

MINIMUM SYSTEM REQUIREMENTS

* RAM: 4MB
* HARD DRIVE SPACE: 25MB
* OS: Microsoft DOS, Microsoft Windows 3.1 or later, PC DOS
* PROCESSOR: INTEL 386
* SOFTWARE: TurboC++

MY SYTEM SPECIFACTIONS

* RAM: 8GB
* HARD DRIVE SPACE: 1TB
* OS: Microsoft Windows 10(64 bit)
* PROCESSOR: Intel Core i3-6100U
* SOFTWARE: TurboC++
* Text Editor: Visual Studio Code Insiders

HEADER FILES USED

* fstream.h

(fstream: open (), fstream: write (),

fstream::read ())

* stdio.h

(rename (), remove ())

* conio.h

(getch (), clrscr ())

* dos.h

(delay ())

* math.h

(abs ())

BINARY FILES USED

* leaderboard.dat
* f2.dat

TEXT FILES USED

* mainMenu.txt
* Data.txt

CLASSES

**1.player**

* + Public data members:

char name[50]

int gamesWon

int gamesLost

char gameName[10]

char piece

int games

* Public member functions:

int move()

void displayGame(moveClass m, int boardi, int boardj)

**2.moveClass**

* Public data members:

int si, sj, di, dj

* Constructor:

si = 7;

sj = 7;

di = 0;

dj = 0;

CODE

#include <iostream.h>

#include <string.h>

#include <fstream.h>

#include <stdio.h>

#include <conio.h>

#include <dos.h>

#include <math.h>

char boardSymbols[8][8]; //8x8 2d array containing piece symbols

void displayBoard(int di, int dj);

char error[10];

class moveClass //this is for creating a moveClass object to use the 4 parameters of the move

{

public:

int si, sj, di, dj;

//source and destination coordinates of the moveClass object ( moveClass controller)

moveClass()

{

si = 7;

sj = 7;

di = 0;

dj = 0;

}

};

class player //

{

public:

char name[50];

int gamesWon;

int gamesLost;

char gameName[10];

char piece; //selected piece on the chess boards

int games;

int move();

void displayGame(moveClass m, int boardi, int boardj)

{

clrscr();

displayBoard(boardi, boardj);

cout << name << " W,A,S,D to move the controller , space to select a piece ,esc to forfeit " << endl;

cout << "controller : (" << m.si << "," << m.sj << ")" << endl;

cout << "Piece:" << piece << endl;

cout << "Destination : (" << m.di << "," << m.dj << ")" << endl;

cout << error << endl;

}

player()

{

gamesWon = 0;

gamesLost = 0;

games = 0;

piece = ' ';

}

};

int king(moveClass m) //checks if king can move within the given moveClass parameters

{

//KING

cout << abs(m.dj - m.sj) << " " << abs(m.si - m.di);

if (abs(m.dj - m.sj) <= 1 && abs(m.si - m.di) <= 1)

{

return 1;

}

else

{

return 0;

}

}

int pawn(moveClass m) //checks if pawn can move within the given moveClass parameters

{

if (m.si == 1 || m.si == 6)

{

if (abs(m.di - m.si) <= 2 && abs(m.dj - m.sj) == 0)

{

return 1;

}

else

{

return 0;

}

}

else if (abs(m.di - m.si) == 1 && abs(m.dj - m.sj) == 0)

{

return 1;

}

else if (abs(m.di - m.si) == abs(m.dj - m.sj) && boardSymbols[m.di][m.dj] != ' ')

{

return 0;

}

else

{

return 0;

}

}

int queen(moveClass m) //checks if queen can move within the given moveClass parameters

{

if (abs(m.di - m.si) == abs(m.dj - m.sj)) //diagonal -> change in x coordinate equals change in y coordinate

{

return 1;

}

else if (m.si == m.di || m.dj == m.sj)

{

return 1;

}

else

{

return 0;

}

}

int bishop(moveClass m) //checks if bishop can move within the given moveClass parameters

{

if (abs(m.di - m.si) == abs(m.dj - m.sj)) //diagonal -> change in x coordinate equals change in y coordinate

{

return 1;

}

else

{

return 0;

}

}

int rook(moveClass m) //checks if rook can move within the given moveClass parameters

{

if (m.si == m.di || m.dj == m.sj)

{

return 1;

}

else

{

return 0;

}

}

int knight(moveClass m) //checks if knight can move within the given moveClass parameters

{

//Case1(a guy moves two steps forward or backward and moves a step to the right or left)

if (abs(m.si - m.di) == 1 && abs(m.sj - m.dj) == 2)

{

return 1;

}

//Case2(a guys moves two steps to the right or to the left and moves a step backward or forward)

else if (abs(m.si - m.di) == 2 && abs(m.sj - m.dj) == 1)

{

return 1;

}

else

{

return 0;

}

}

int checkMove(moveClass m, char piece) //checks if piece can move within the given moveClass parameters

{

if (piece == 'k' || piece == 'K')

{

if (king(m))

{

return 1;

}

else

{

return 0;

}

}

else if (piece == 'p' || piece == 'P')

{

if (pawn(m))

{

return 1;

}

else

{

return 0;

}

}

else if (piece == 'q' || piece == 'Q')

{

if (queen(m))

{

return 1;

}

else

{

return 0;

}

}

else if (piece == 'r' || piece == 'R')

{

if (rook(m))

{

return 1;

}

else

{

return 0;

}

}

else if (piece == 'b' || piece == 'B')

{

if (bishop(m))

{

return 1;

}

else

{

return 0;

}

}

else if (piece == 'n' || piece == 'N')

{

if (knight(m))

{

return 1;

}

else

{

return 0;

}

}

else

{

return 0;

}

}

int player::move() //returns -1 for quit , 0 for move success ,1 for wrong move

{

char ch;

moveClass controller;

piece = ' ';

controller.sj = 0;

controller.si = 0;

while (ch != 27) //loop to move the source controller

{

ch = getch();

//moving

if (ch == 97) //ascii for a to move left

{

if (controller.sj != 0)

{

controller.sj = controller.sj - 1;

}

}

else if (ch == 119) //ascii for w to move up

{

if (controller.si != 0)

{

controller.si = controller.si - 1;

}

}

else if (ch == 100) //ascii for d to move right

{

if (controller.sj != 7)

{

controller.sj = controller.sj + 1;

}

}

else if (ch == 115) //ascii for s to move down

{

if (controller.si != 7)

{

controller.si = controller.si + 1;

}

}

else if (ch == 32) //ascii for space -> to fix the piece

{

piece = boardSymbols[controller.si][controller.sj];

controller.dj = controller.sj;

controller.di = controller.si;

displayGame(controller, controller.di, controller.dj);

while (ch != 27) //loop to move the destination controller

{

ch = getch();

//moving

if (ch == 97) //ascii for a to move left

{

if (controller.dj >= 0)

{

controller.dj = controller.dj - 1;

}

}

else if (ch == 119) //ascii for w to move up

{

if (controller.di >= 0)

{

controller.di = controller.di - 1;

}

}

else if (ch == 100) //ascii for d to move right

{

if (controller.dj <= 7)

{

controller.dj = controller.dj + 1;

}

}

else if (ch == 115) //ascii for s to move down

{

if (controller.di <= 7)

{

controller.di = controller.di + 1;

}

}

else if (ch == 32)

{

if (!checkMove(controller, piece))

{

strcpy(error, "wrong move ..");

displayGame(controller, controller.di, controller.dj);

return 1;

}

boardSymbols[controller.si][controller.sj] = ' ';

boardSymbols[controller.di][controller.dj] = piece;

displayGame(controller, controller.di, controller.dj);

return 0;

//piece gets inserted here on hitting enter

}

displayGame(controller, controller.di, controller.dj);

}

}

else if (ch == 27)

{

return -1;

}

displayGame(controller, controller.si, controller.sj);

}

return 1;

}

int checkMatePlayer1() //if player1's king is not present on the board , it returns 1

{

for (int i = 0; i < 8; i++)

{

for (int j = 0; j < 8; j++)

{

if (boardSymbols[i][j] == 'K')

{

return 0;

}

}

}

return 1;

}

int checkMatePlayer2() //same as checkMatePlayer1()

{

for (int i = 0; i < 8; i++)

{

for (int j = 0; j < 8; j++)

{

if (boardSymbols[i][j] == 'k')

{

return 0;

}

}

}

return 1;

}

void loadDefault() //extract the default data from the file into 2d array

{

fstream f;

f.open("Data.txt", ios::in);

char x[100];

int i = 0;

int j = 0;

for (i = 0; i < 8; i++)

{

f.getline(x, 100);

for (j = 0; j < 8; j++)

{

if (x[j] == 'X')

{

boardSymbols[i][j] = ' ';

}

else

{

boardSymbols[i][j] = x[j];

}

}

}

f.close();

}

int player1Move(player &p1, player &p2)

{

cout << p1.name << " ur move (u can only move upper pieces) :";

strcpy(error, " ");

int move = p1.move();

while (move != 0)

{

if (move == -1)

{

clrscr();

cout << p2.name << " wins!!" << '\a' << endl;

delay(2000);

++p2.gamesWon;

++p1.gamesLost;

return 1;

}

else if (checkMatePlayer2()) //if player2's king is not present then checkmate by player 1 -> player 1 wins

{

clrscr();

cout << "checkmate" << p1.name << "wins" << endl;

delay(2000);

p1.gamesWon++;

p2.gamesLost++;

return 1;

}

move = p1.move();

}

return 0;

}

int player2Move(player &p1, player &p2)

{

cout << p2.name << " ur move (u can only move lowercase pieces) :";

strcpy(error, " ");

int move = p2.move();

while (move != 0)

{

if (move == -1)

{

clrscr();

cout << p1.name << "wins !!" << '\a' << endl;

delay(2000);

++p1.gamesWon;

++p2.gamesLost;

return 1;

}

else if (checkMatePlayer1()) //if player1's king is not present then checkmate by player 2 -> player 2 wins

{

clrscr();

cout << "checkmate!!" << p2.name << " wins " << endl;

delay(2000);

p2.gamesWon++;

p1.gamesLost++;

return 1;

}

move = p2.move();

}

return 0;

}

void play(player &player1, player &player2) //evaluates player1Move and player2Move

{

while (1) //infinite loop till checkmate or if a player forfeits

{

clrscr();

displayBoard(-1, -1);

if (player1Move(player1, player2))

{

break;

}

clrscr();

displayBoard(-1, -1);

if (player2Move(player1, player2))

{

break;

}

}

}

void leaderboard\_write(player p) //writes and also modifies player details(if the player is already on the leaderboard)

{

fstream f;

player temp;

fstream f2;

int found = 0;

f.open("leaderboard.dat", ios::binary | ios::app | ios::in);

f2.open("f2.dat", ios::binary | ios::app);

while (f.read((char \*)&temp, sizeof(player)))

{

if (strcmp(temp.name, p.name) == 0)//writes updated details to f2 if player is found

{

p.gamesLost = p.gamesLost + temp.gamesLost;

p.gamesWon = p.gamesWon + temp.gamesWon;

p.games++;

f2.write((char \*)&p, sizeof(player));

found = 1;

}

else

{

f2.write((char \*)&temp, sizeof(player));//writes existing details of other players to f2

}

}

if (found == 0)

{

f2.write((char \*)&p, sizeof(player));//writes new player details if not found earlier to f2

}

f.close();

f2.close();

remove("leaderboard.dat");

rename("f2.dat", "leaderboard.dat");

}

void leaderboard\_search() //searches the players name in the leaderboard

{

fstream f;

player p;

f.open("leaderboard.dat", ios::binary | ios::app | ios::in);

char name[50];

cout << "enter name to search :";

gets(name);

while (f.read((char \*)&p, sizeof(p)))

{

if (strcmp(name, p.name) == 0)

{

cout << "Name" << '\t' << "Games" << '\t' << "GameName" << '\t' << '\t' << "Won" << '\t' << "Lost" << endl;

cout << p.name << '\t' << p.games << '\t' << p.gameName << '\t' << '\t' << p.gamesWon << '\t' << p.gamesLost << endl;

break;

}

}

f.close();

}

int leaderboard\_count() //returns number of players

{

fstream f;

player p;

f.open("leaderboard.dat", ios::binary | ios::in);

int count = 0;

while (f.read((char \*)&p, sizeof(p)))

{

++count;

}

f.close();

return count;

}

void leaderboard\_read() //reads and prints from leaderboard.dat

{

fstream f;

f.open("leaderboard.dat", ios::binary | ios::out | ios::app | ios::in);

player p;

cout << "Name" << '\t' << "Games" << '\t' << "GameName" << '\t' << "Won" << '\t' << "Lost" << endl;

f.read((char \*)&p, sizeof(p));

while (!f.eof())

{

cout << p.name << '\t' << p.games << '\t' << p.gameName << '\t' << '\t' << p.gamesWon << '\t' << p.gamesLost << endl;

f.read((char \*)&p, sizeof(p));

}

f.close();

}

int demo(char name[50]) //starts a demo for the chess

{

loadDefault();

clrscr();

char c;

while (1)

{

while (c != 'd')

{

clrscr();

cout << name << " Welcome to ASCII CHESS! THIS IS A Tutorial(Press esc to skip tutorial)" << endl

<< endl;

cout << "Step 1 : press d to move the controller to the right " << endl;

displayBoard(1, 3);

c = getch();

if (c == 27)

{

return 0;

}

else if (c == 'd')

{

clrscr();

displayBoard(1, 4);

break;

}

}

delay(1000);

while (c != 'w')

{

clrscr();

cout << name << " Welcome to ASCII CHESS! THIS IS A Tutorial(Press esc to skip tutorial)" << endl

<< endl;

cout << "Step 2: press w to move the controller up " << endl;

displayBoard(1, 3);

c = getch();

if (c == 27)

{

return 0;

}

else if (c == 'w')

{

clrscr();

displayBoard(0, 3);

break;

}

}

delay(1000);

clrscr();

while (c != 'a')

{

clrscr();

cout << name << " Welcome to ASCII CHESS! THIS IS A Tutorial(Press esc to skip tutorial)" << endl

<< endl;

cout << "Step 3 : press a to move the controller left " << endl;

displayBoard(1, 3);

c = getch();

if (c == 27)

{

return 0;

}

else if (c == 'a')

{

clrscr();

displayBoard(1, 2);

}

}

delay(1000);

clrscr();

while (c != 's')

{

clrscr();

cout << name << " Welcome to ASCII CHESS! THIS IS A Tutorial(Press esc to skip tutorial)" << endl

<< endl;

cout << "Step 4: press s to move the controller down " << endl;

displayBoard(1, 3);

c = getch();

if (c == 27)

{

return 0;

}

else if (c == 's')

{

clrscr();

displayBoard(2, 3);

}

}

delay(1000);

while (c != 32)

{

clrscr();

cout << name << " Welcome to ASCII CHESS! THIS IS A Tutorial(Press esc to skip tutorial)" << endl

<< endl;

displayBoard(1, 3);

cout << " Step 5 : press space to select the highlighted piece" << endl;

c = getch();

if (c == 27)

{

return 0;

}

}

clrscr();

cout << name << " Welcome to ASCII CHESS! THIS IS A Tutorial(Press esc to skip tutorial)" << endl

<< endl;

displayBoard(1, 3);

cout << endl

<< " Piece selected : " << boardSymbols[1][3] << endl;

delay(1000);

int i = 1;

int j = 3;

char ch;

clrscr();

cout << name << " Welcome to ASCII CHESS! THIS IS A Tutorial(Press esc to skip tutorial)" << endl

<< endl;

cout << " Step 6 : Navigate to 2,3 and press space to insert pawn to 2,3" << endl;

displayBoard(i, j);

cout << "Controller : (" << i << "," << j << ")" << endl;

cout << "piece selected : P" << endl;

ch = getch();

while (1)

{

if (ch == 97) //ascii for a to move left

{

if (j != 0)

{

j = j - 1;

}

}

else if (ch == 119) //ascii for w to move up

{

if (i != 0)

{

i = i - 1;

}

}

else if (ch == 100) //ascii for d to move right

{

if (j != 7)

{

j = j + 1;

}

}

else if (ch == 115) //ascii for s to move down

{

if (i != 7)

{

i = i + 1;

}

}

else if (ch == 27)

{

return 0;

}

clrscr();

cout << name << " Welcome to ASCII CHESS! THIS IS A Tutorial(Press esc to skip tutorial)" << endl

<< endl;

cout << "Step 6 : Navigate to 2,3 and press space to insert pawn to 2,3" << endl;

displayBoard(i, j);

cout << "Controller : (" << i << "," << j << ")" << endl;

cout << "piece selected : P" << endl;

ch = getch();

if (ch == 27)

{

return 0;

}

else if (i == 2 && j == 3 && ch == 32)

{

break;

}

}

boardSymbols[2][3] = boardSymbols[1][3];

boardSymbols[1][3] = ' ';

clrscr();

displayBoard(i, j);

cout << "Controller : (" << i << "," << j << ")" << endl;

cout << "piece selected : P" << endl

<< endl;

cout << "GREAT CHESS V1.0 TUTORIAL COMPLETED !" << endl;

cout << "This is how you move the pieces :" << endl

<< "1. use w,a,s,d to navigate to the piece" << endl

<< "2.Press space to select the piece" << endl

<< "3.Navigate again to empty cell or enemy piece , press space again to insert" << endl;

cout << endl;

getch();

return 0;

}

}

void checkDemo(char p1[50], char p2[50]) //checks if demo should be called for a player

{

fstream f;

f.open("leaderboard.dat", ios::binary | ios::out | ios::app | ios::in);

player dummy;

int foundPlayer1 = 0;

int foundPlayer2 = 0;

f.read((char \*)&dummy, sizeof(dummy));

while (!f.eof())

{

if (strcmp(p1, dummy.name) == 0)

{

foundPlayer1 = 1;

}

if (strcmp(p2, dummy.name) == 0)

{

foundPlayer2 = 1;

}

f.read((char \*)&dummy, sizeof(dummy));

}

f.close();

if (foundPlayer1 == 0)

{

demo(p1);

}

if (foundPlayer2 == 0)

{

demo(p2);

}

else

{

cout << p1 << " , " << p2 << " Welcome back to ASCII CHESS !" << endl;

delay(2000);

}

}

void newGame() //initiates a new game: takes player's name , games name -> play().

{

char gameName[10];

cout << "enter game name :";

gets(gameName);

cout << endl;

player player1, player2;

strcpy(player1.gameName, gameName);

strcpy(player2.gameName, gameName);

cout << "player1 enter ur name:";

gets(player1.name);

cout << endl;

cout << "player2 enter ur name :";

gets(player2.name);

checkDemo(player1.name, player2.name);

clrscr();

loadDefault();

play(player1, player2); //starts the actual chess game

clrscr();

player1.games++;

player2.games++;

leaderboard\_write(player1);

leaderboard\_write(player2);

delay(1000);

cout << endl;

}

void displayMenu() //displays the start page of the program

{

fstream f;

f.open("mainMenu.txt", ios::in);

char x[500];

while (!f.eof())

{

f.getline(x, 500);

cout << x << endl;

}

f.close();

}

void leaderboard\_menu()

{

int c;

while (c != 4)

{

clrscr();

cout << "LEADERBOARD" << endl

<< endl;

cout << "1.Show leader board" << endl

<< "2.Search player" << endl

<< "3.Count number of players" << endl

<< "4.Back" << endl

<< endl;

cout << "enter choice :";

cin >> c;

switch (c)

{

case 1:

leaderboard\_read();

getch();

break;

case 2:

leaderboard\_search();

delay(2000);

break;

case 3:

cout << "number of players : " << leaderboard\_count() << endl;

getch();

case 4:

break;

}

}

}

void startGame() //gets the option and starts newGame() leaderboard\_menu() or exit the game.

{

char choice;

while (choice != 51)

{

clrscr();

displayMenu();

choice = getch();

switch (choice)

{

case 49: //49 is ascii for number 1

cout << "--------------------------------------------------------------------" << endl;

newGame();

cout << "--------------------------------------------------------------------" << endl;

displayMenu();

break;

case 50: //ascii for number 2

leaderboard\_menu();

break;

case 51:

cout << endl

<< " Thank you for playing Chess v1.0 !" << endl

<< " Made Dhruv Devasthale" << endl

<< endl;

break;

}

}

}

void displayBoard(int di, int dj) //displays the chess board

{

cout << " 0 1 2 3 4 5 6 7 " << endl;

for (int i = 0; i < 8; i++)

{

cout << " +---+---+---+---+---+---+---+---+" << endl;

cout << i << " | ";

for (int j = 0; j < 8; j++)

{

if (i == di && j == dj)

{

cout << "\b[" << boardSymbols[i][j] << "]| ";

}

else

{

cout << boardSymbols[i][j] << " | ";

}

}

cout << endl;

}

cout << " +---+---+---+---+---+---+---+---+" << endl;

}

void main()

{

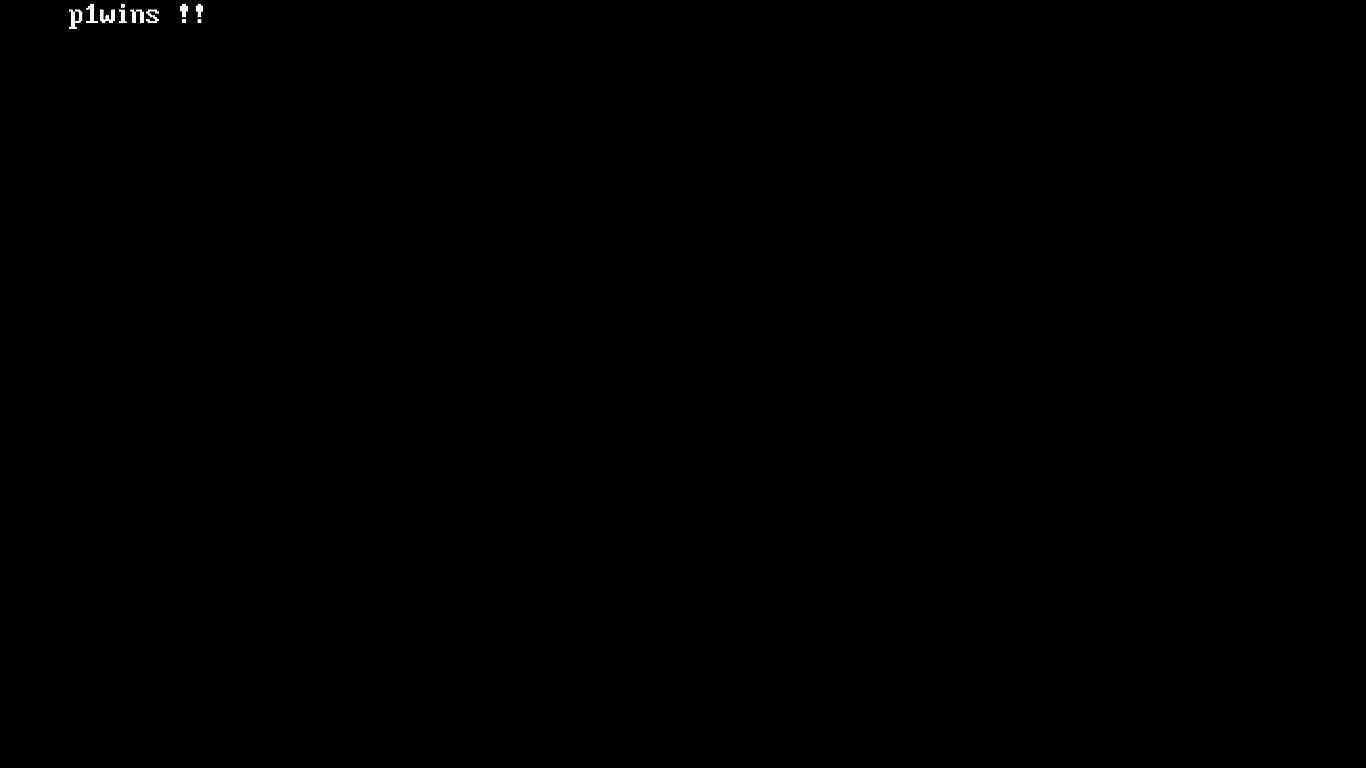
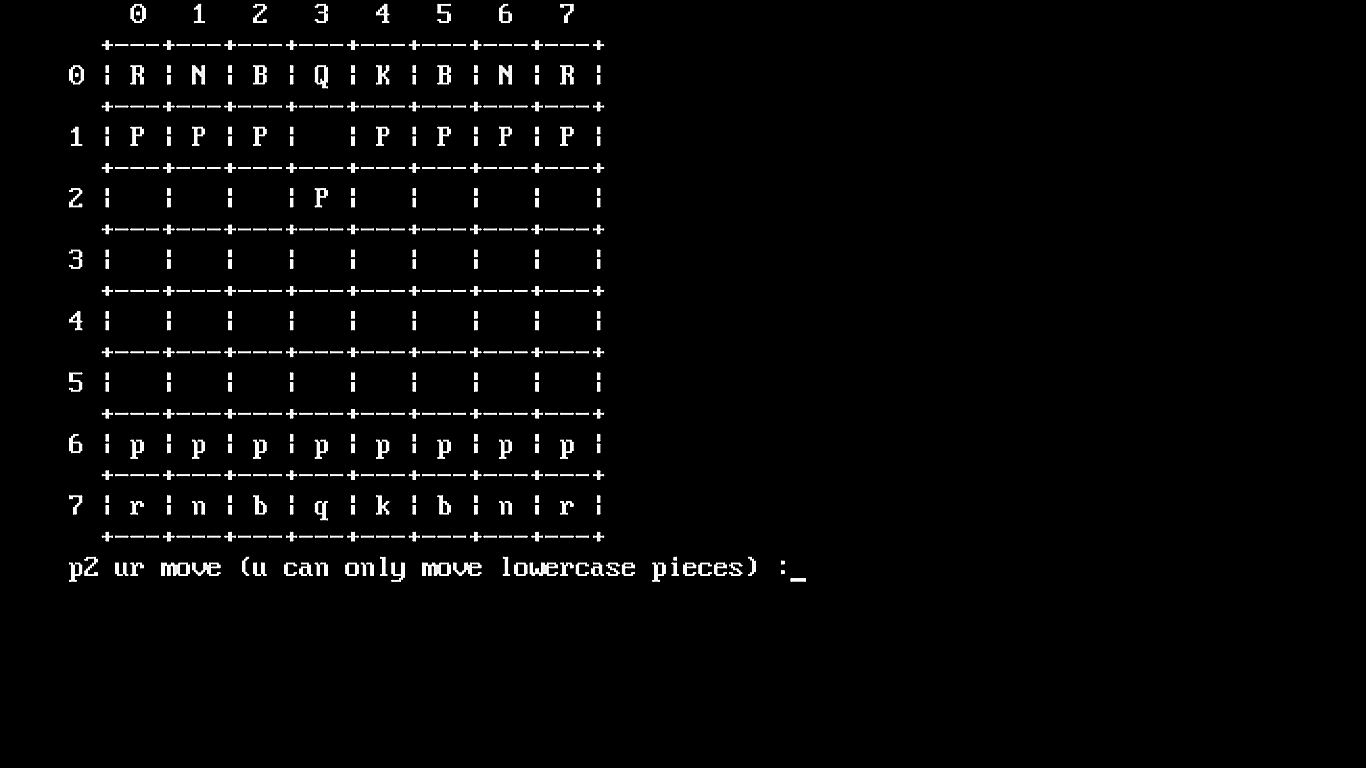
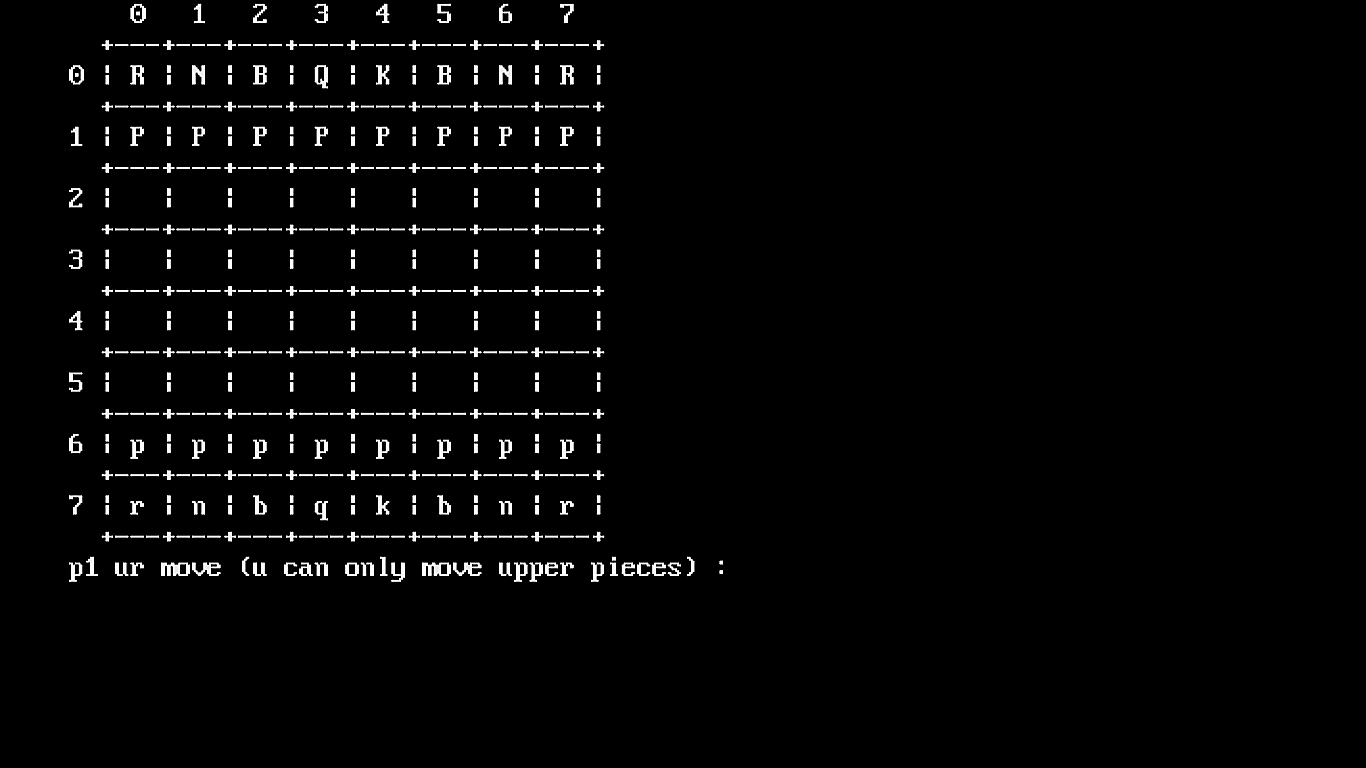
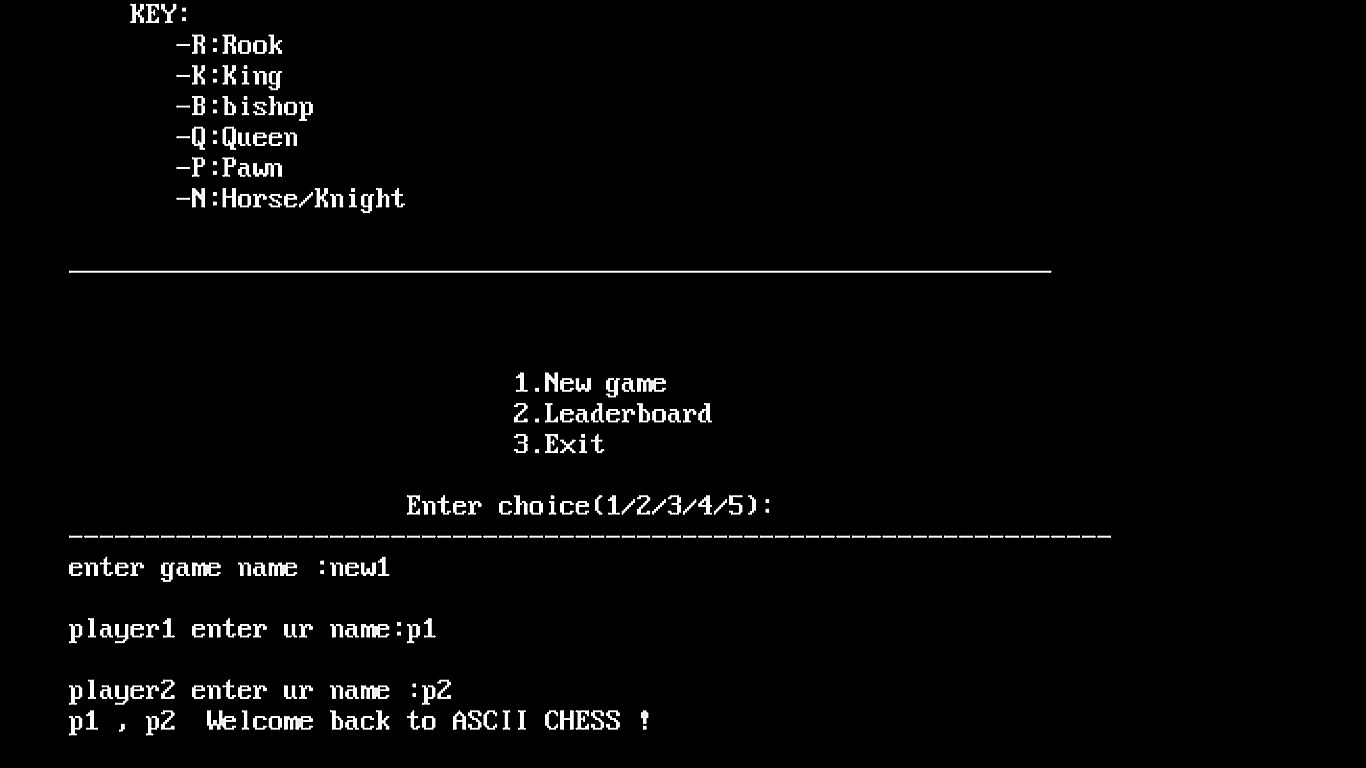
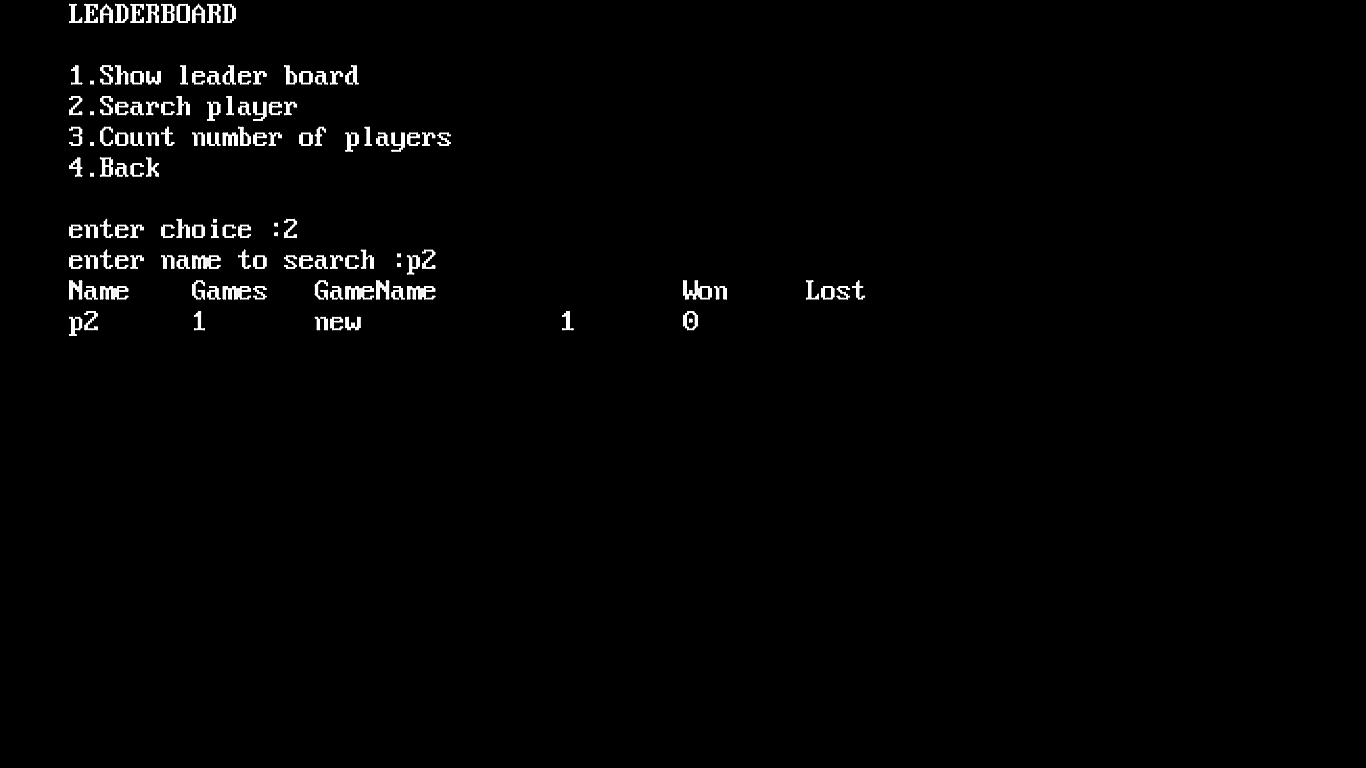
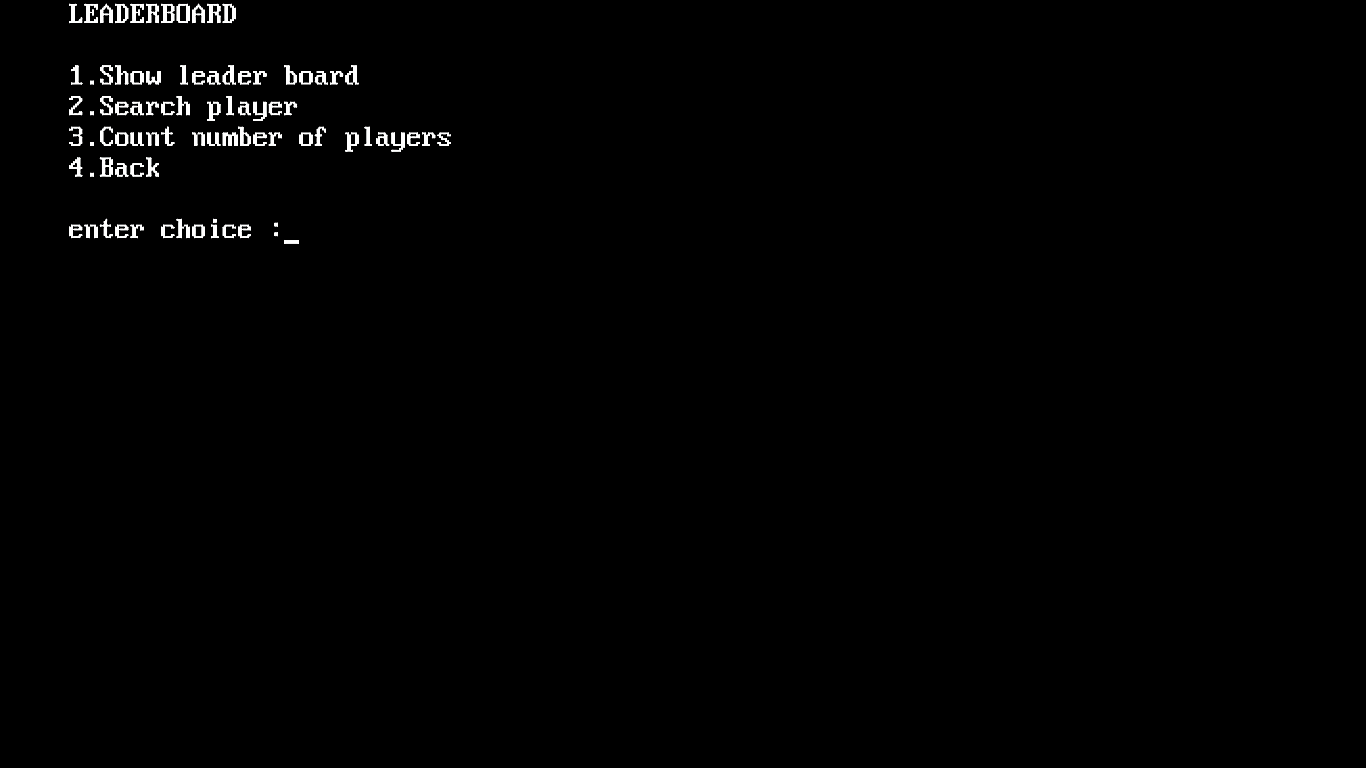
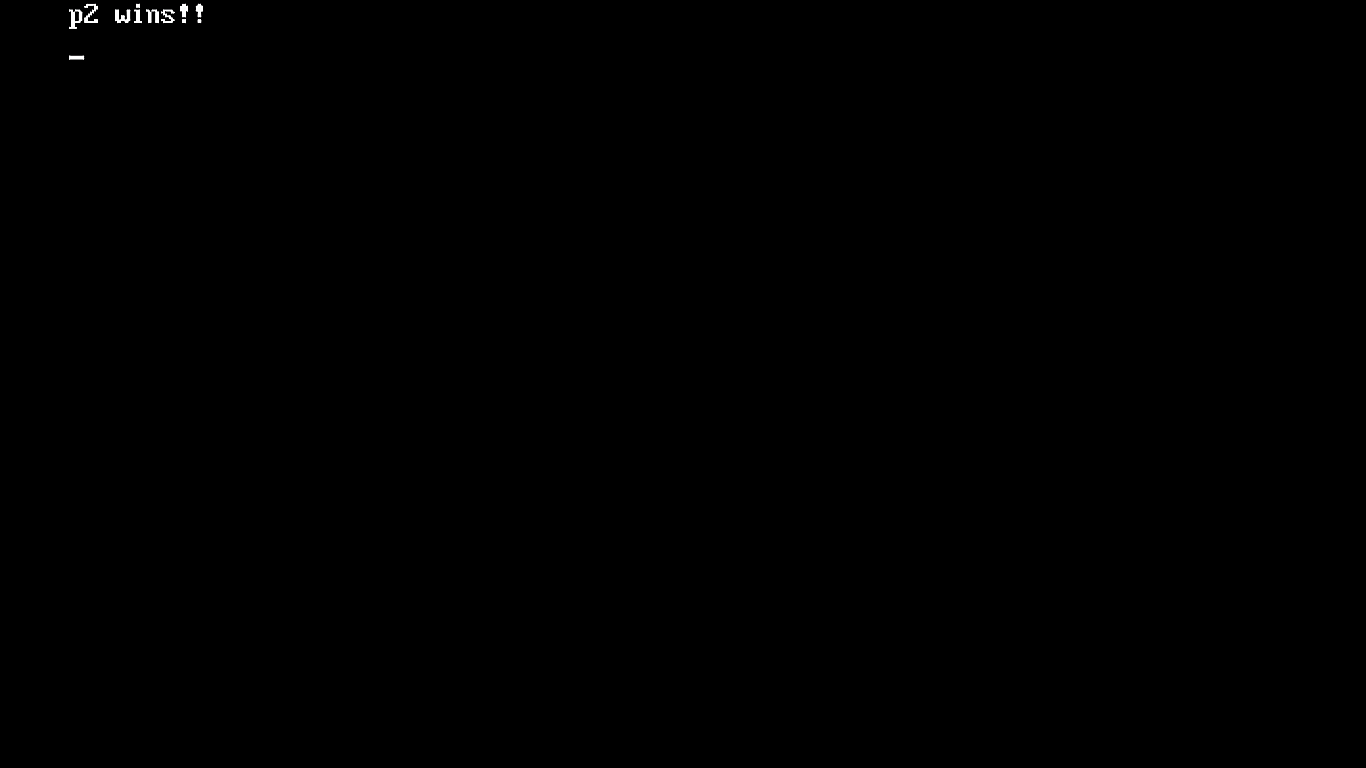
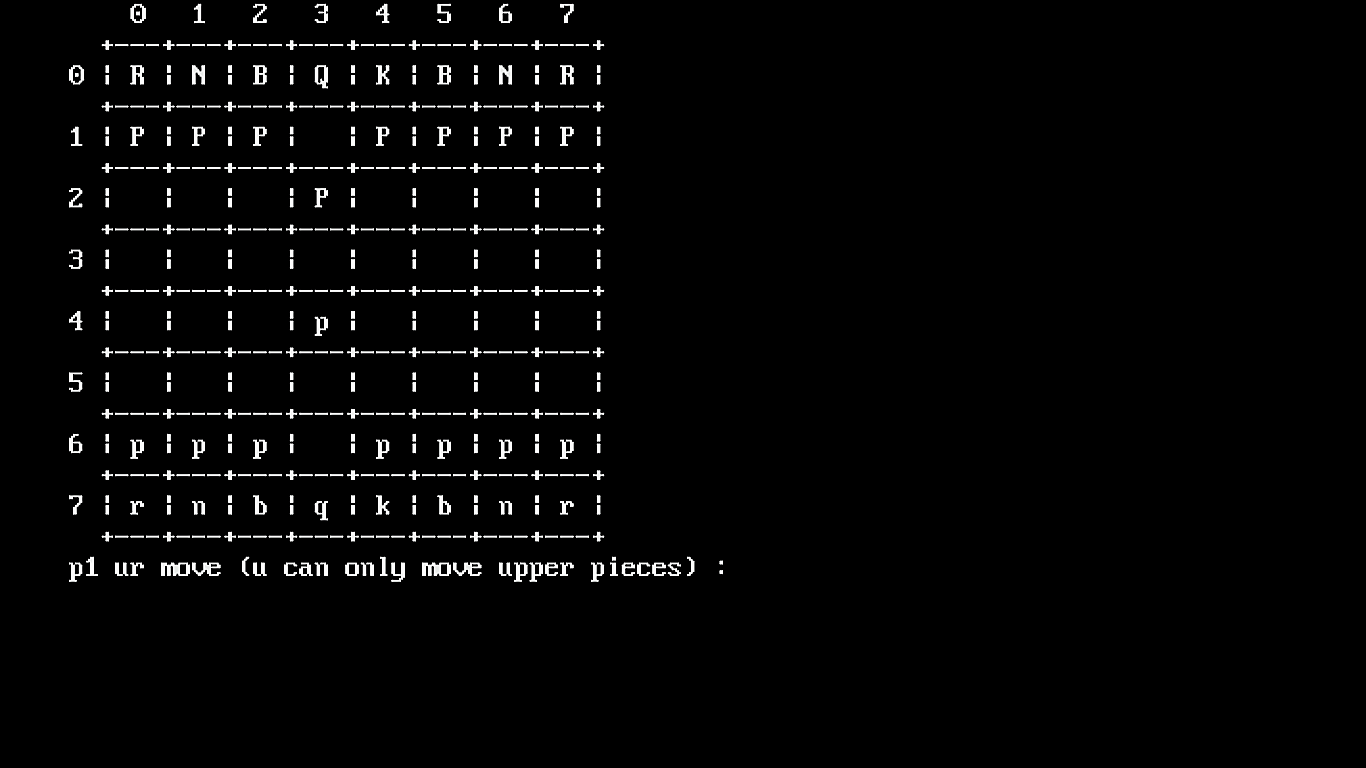
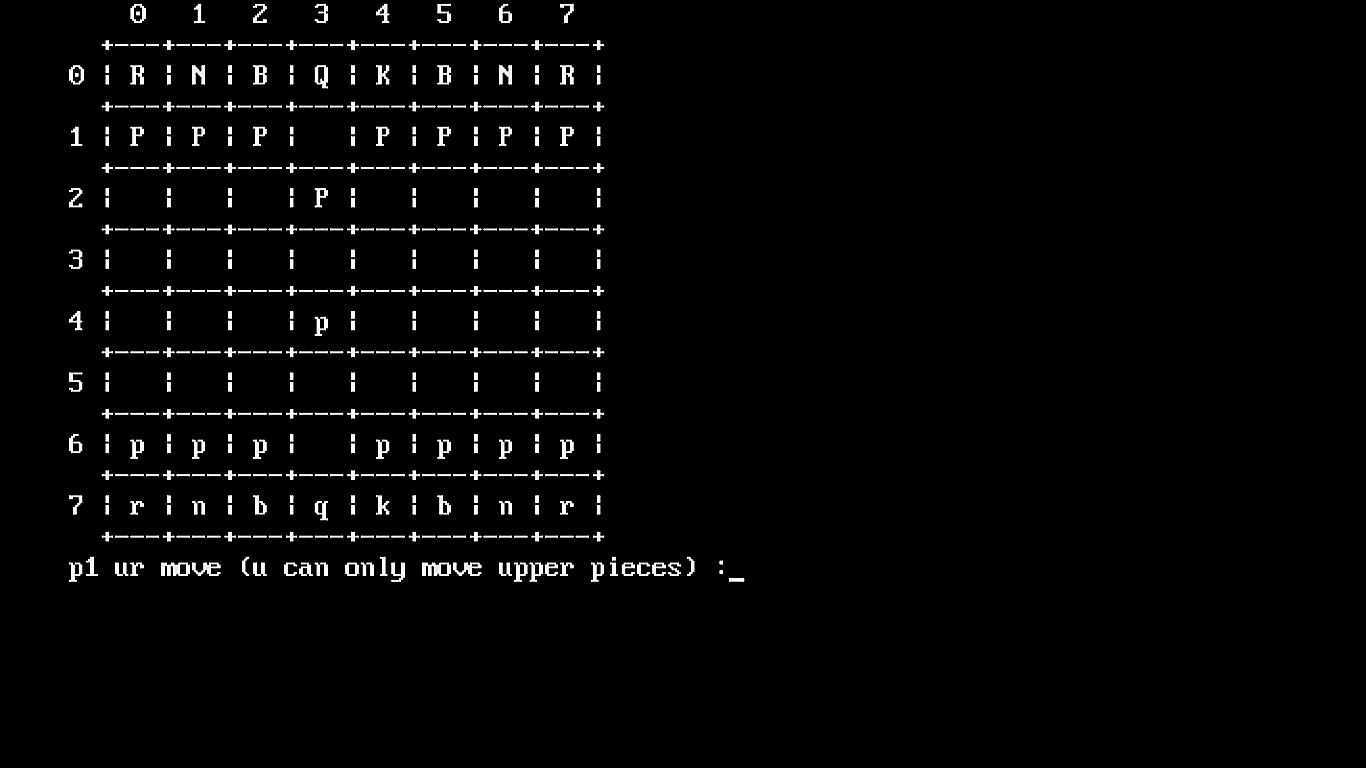
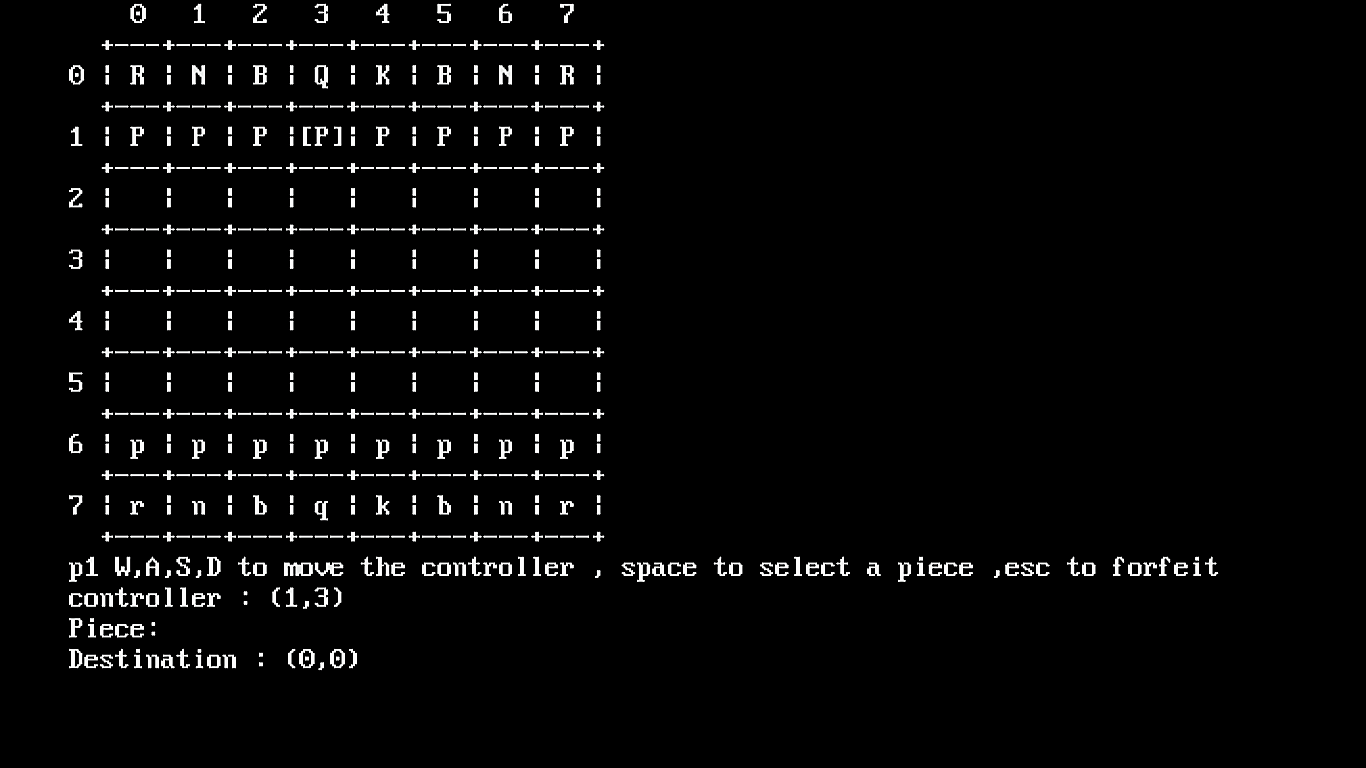
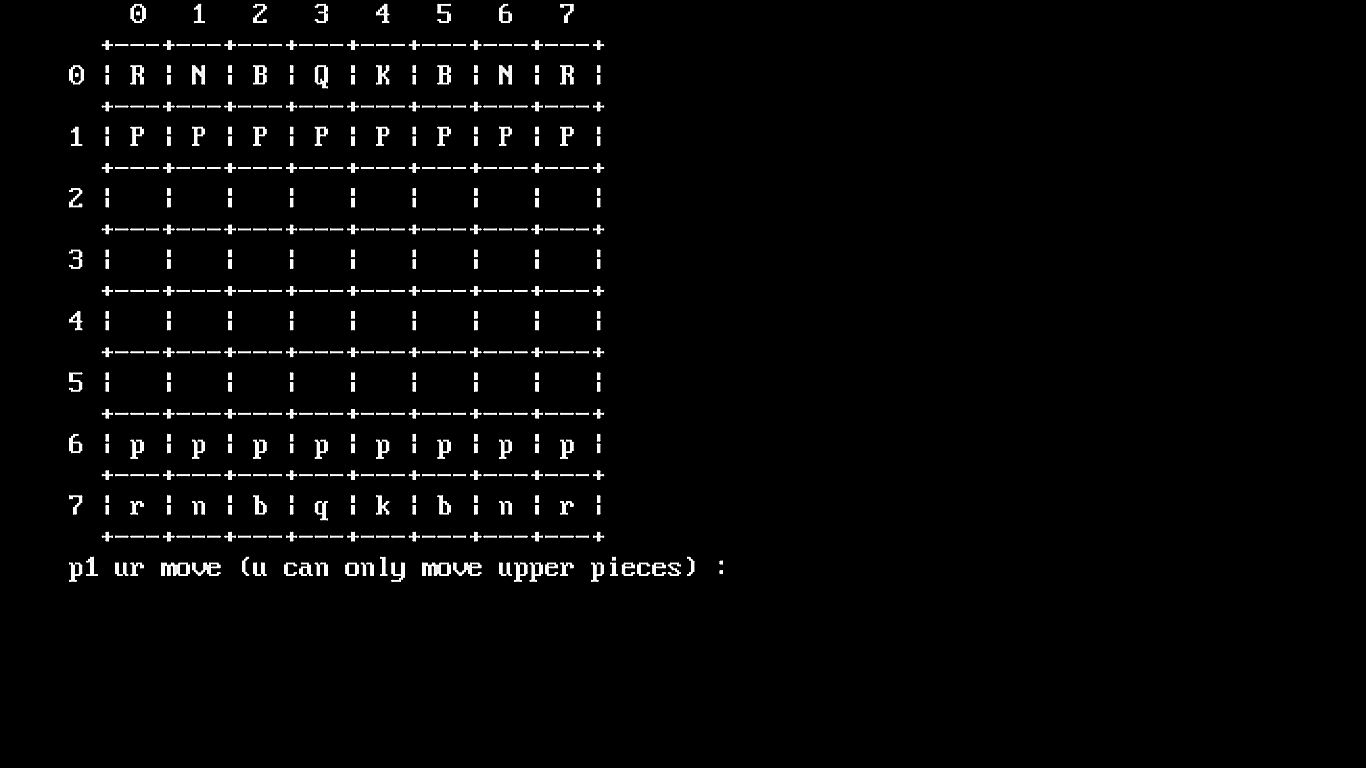
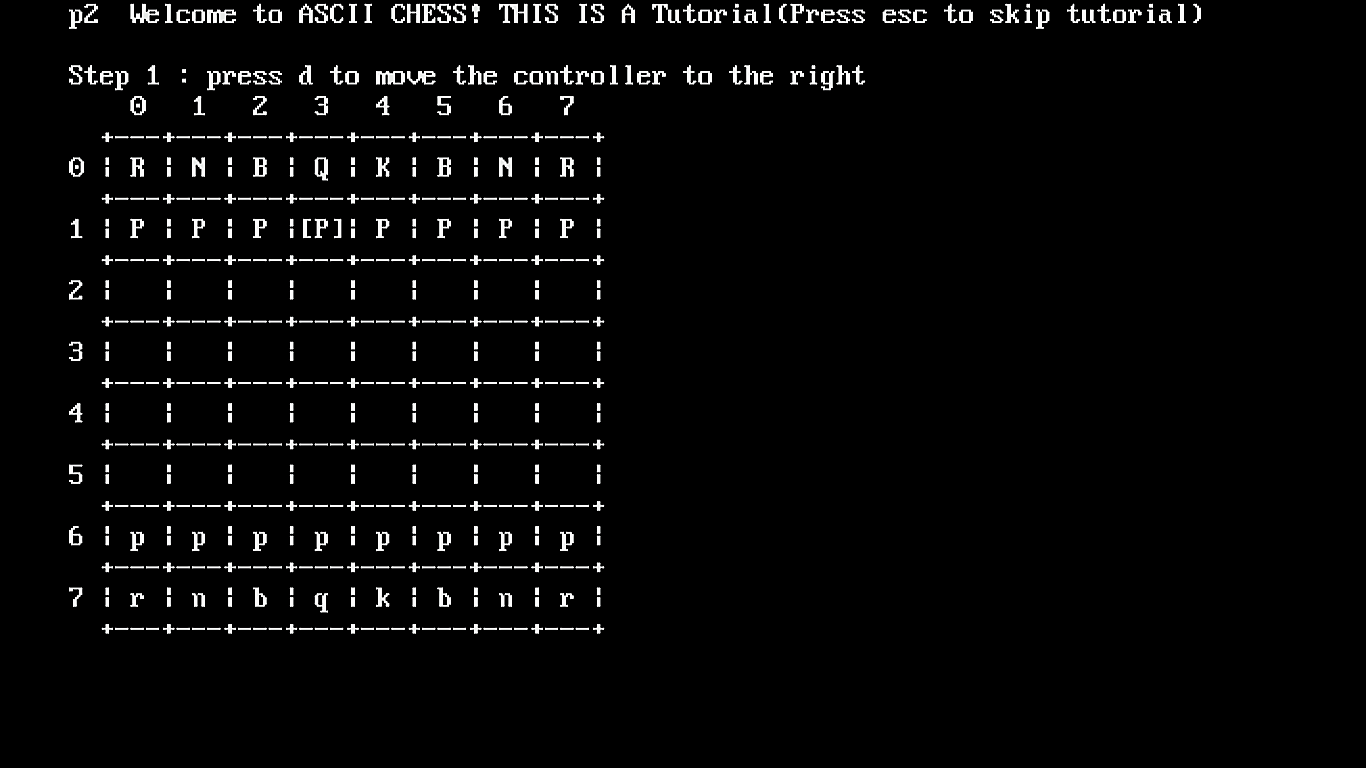
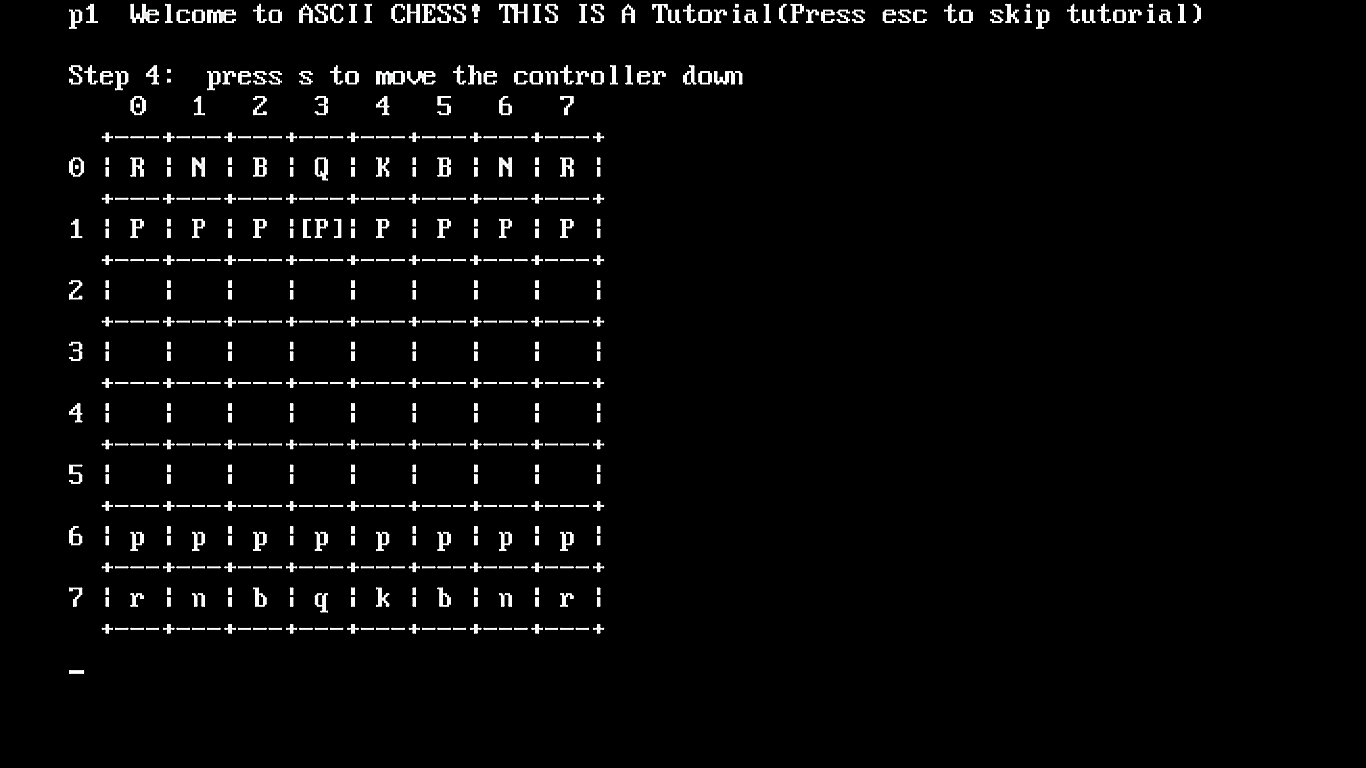
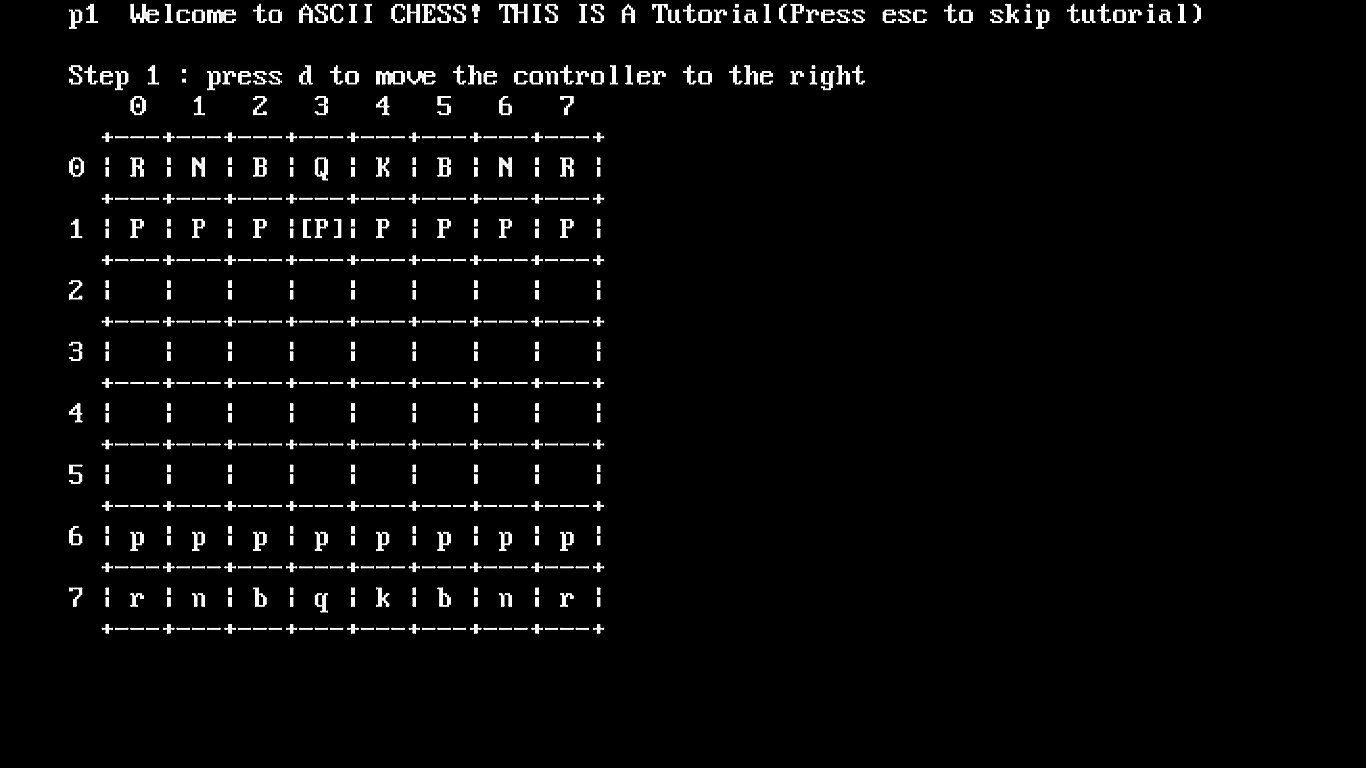
clrscr();

startGame();

getch();

}

OUTPUT



SCOPE FOR IMPROVEMENT

1.Improve Graphics

2.Implement all real-life chess rules

REFERENCES:

1. <https://gitlab.com/dd-16/12th>
2. <https://stackoverflow.com/>
3. <http://www.cplusplus.com/>
4. <http://www.tutorialspoint.com/>