**DOKUZ EYLUL UNIVERSITY**

**ENGINEERING FACULTY**

**DEPARTMENT OF COMPUTER ENGINEERING**

**CME1251 PROJECT BASED LEARNING – I**

**FINAL REPORT**

**DEU-CENG Chess**

**SHAPER**

**by**

**DENİZ KÜÇÜKKARA :2019510128**

**Lecturers**

**Özlem Yerlikaya**

**Yalçın Cebi**

**IZMIR**

**29.01.2021**

# 

# CHAPTER ONE

PROGRESS DESCRIPTION

Our task was to design a fluent chess game that can be played by two people, and with proper rules. We started our project by discussing and designing solution alternatives. Then we created the necessary variables and structures and placed our stones on our board in 2 different sets using 2 different arrays. And we were able to move our chess pieces with the Cursor movement. Then we started to design play mode we wrote functions to move the pawn and knight and updated them later in the project and tailored them to our needs. We encountered a lot of difficulties while doing these, especially designing the knight's movements and assigning them to the function was very difficult. Our chess pieces were moving on the board from now on, but our current task was to design them to move smoothly and eat the opponent's piece. But the problem was how to understand that the opponent's piece was the piece in it, we assigned one team to the 1s, the other team to the -1's and the empty board members it to 0, so we solved the problem of capturing your own team members. Now we came to the final part, completing the remaining parts of the game. There were a lot of tasks in this section, and when we got back to our code, there were a lot of places that needed to change.

# CHAPTER TWO

TASK SUMMARY

## Completed Tasks

Deniz Küçükkara I assigned the movement functions of the game. I made the movements of the all pieces, I made the table an array for the smoothness of the game, and I made another integer array that goes equal to the table.

I completed the play mode part by myself. I wrote functions to all stones. I got a checkmate check. I did the Notation part. I opened 2 array tables for the game board. I did print the notations to the file.

## Incomplete Tasks: Reasons and Explanations

Deniz Küçükkara : I dont have any imcomplete task.

## Additional Improvements ~~to the Project~~

Deniz Küçükkara : Appearance of the moving piece instead of showing the cursor during movements. Special "key "for cross movement.

# CHAPTER THREE

EXPLANATION of algorıthms

## Screenshots

Deniz Küçükkara :

To print a table:

We used two 2D arrays and I did a function that prints continuously to refresh them.

Piece control:

I made functions that found places to go when each piece was selected.

Dynamic:

To show the live motion of the part on the screen, I made a function that changes the elements of arrays with each press.

Checkmate :

I did a function that looked at every danger one by one and checked for any stones at the place of danger.





## Functions

Deniz Küçükkara: I opened a function for all pieces. I opened the function one by one for the direction of the pieces and the capture situations

## Algorithms and Solution Strategies

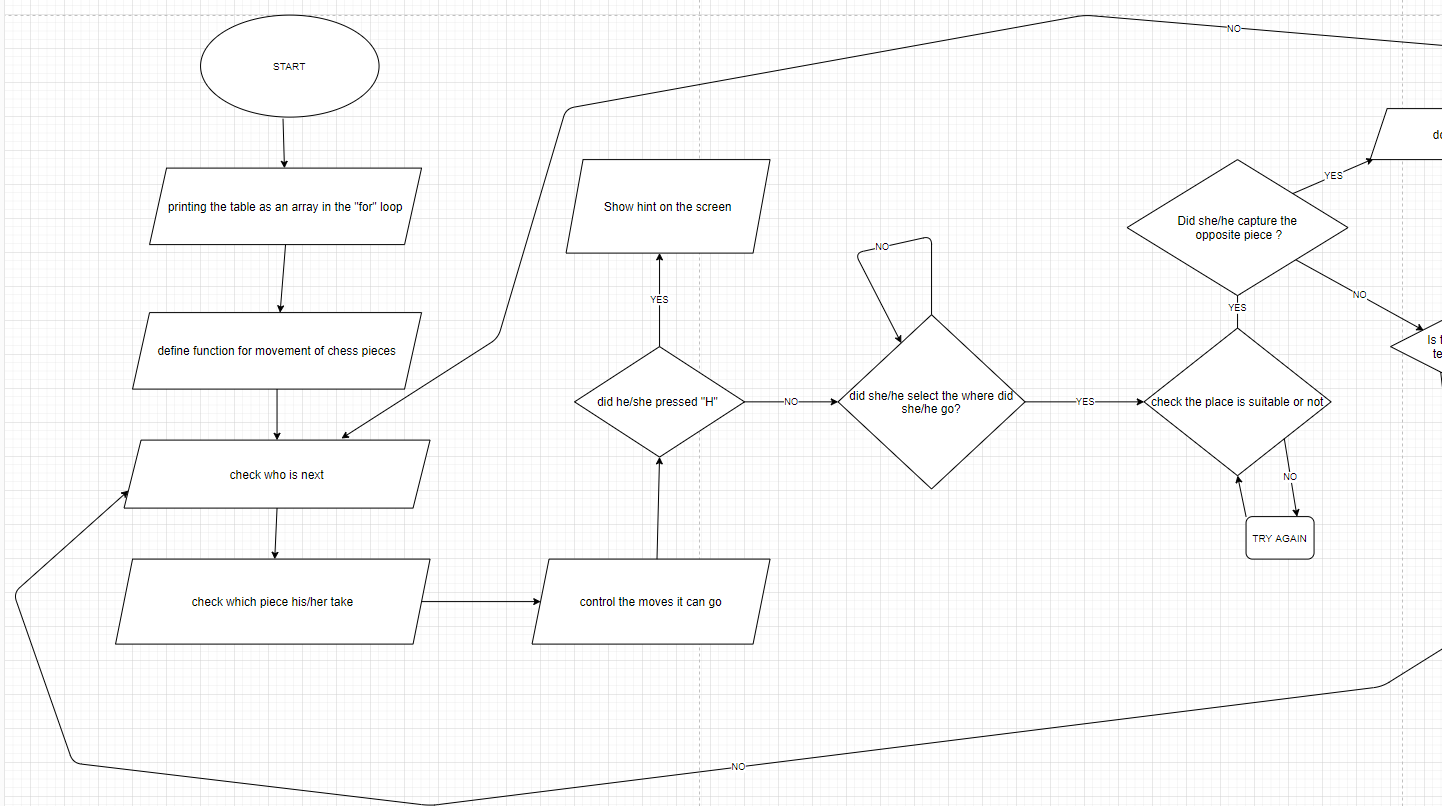
I created a variable named game\_counter to find out who has the game queue. I increased the value of this variable by 1 each time an operation was performed. If this value is a multiple of 2, then it is for the red team, if not for the blue team. I checked the notation that the user entered if there is an x in it, I set the flag to true, it showed that a swallowing operation will be done on this notation. I divided the notation the user entered according to the length and whether there is an x in it. I have identified the first and last columns of the fragmented substrings using letters like a b c. I detected the last line from the notation the user entered. I checked the first row on the game board with for loops and found it. I did the horse checks using the absolute value function. If the column has gone two units by absolute value, the row has to go 1 unit by absolute value. If the row has gone two units by absolute value, it has to go as the column by absolute value 1 unit. For the control of castling movements, I first determined whether it was long or short castling. I checked that the space between king and rook is empty. If it is empty I moved.

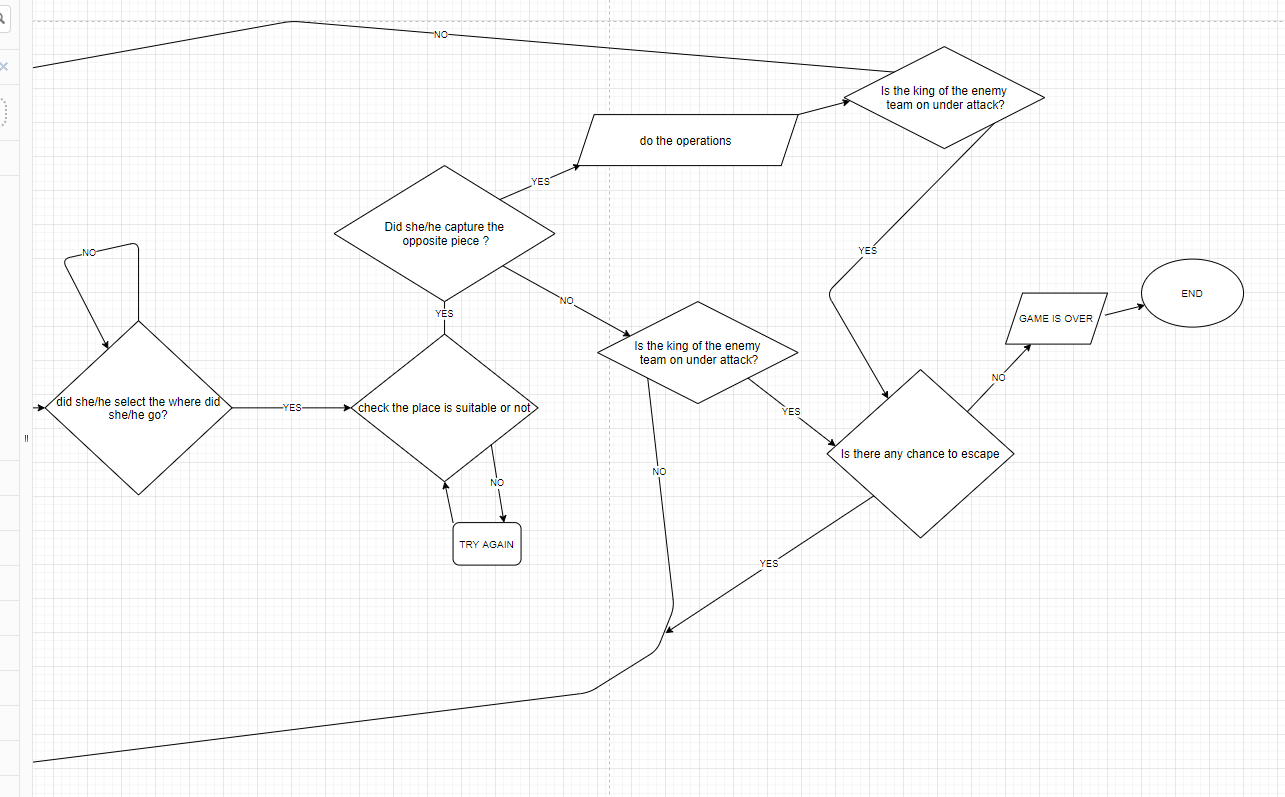
Deniz Küçükkara

First, I prepared a flowchart as mentioned below. Then I started to proceed in accordance with this flow chart. The first week I did the array and I did the movement

of the pieces without rules. The next week I made the rules of the pawn and the rook and drew up a coordinate map that in stantly shows where the pieces go. While doing these, I realized that the 2nd array with integer was required and I prepared it. Finally, I prepared the rook.

Flowchart.





# CHAPTER FOUR

PROBLEMS ENCOUNTERED

Deniz Küçükkara

I had a hard time doing direction functions to pieces. Because I didn't have much practice about functions. But thanks to this project, I was able to learn the functions correctly. Because my friend couldn't do the rook, I had to finish it on the last day. It forced me to write with no plans in my head, but I could handle it. Finally, it forced me to write the movement of the stones so I opened 2 arrays (letters and numbers) with the same value in the cursor position value. This problem disappeared when I type the cursor position in them. When I was making a checkmate, it was necessary to check every part of the king according to each piece and it was extremely difficult.

# CHAPTER FIVE

conclusıon

While doing this project, we spent most of us time researching and reviewing sample codes. Thanks to this project, we gained Analytical thinking skills. It allowed us to see the relationships between events or situations. It helped us think creatively. It would increase the problem-solving ability. It made us think systematically. We learned teamwork and became more familiar with C# and it’s structured.The game starts with 8\*8 board. Players take turns moving their game elements obeying therules. Each player's goal is to eat the opponent's king. If the king is lost or checkmate the game is over.

REFERENCES

[https://www.geeksforgeeks.org/csharp-programming- language/](https://www.geeksforgeeks.org/csharp-programming-%C2%A0%20%C2%A0%20%C2%A0language/)

<https://www.w3schools.com/cs/cs_intro.asp>

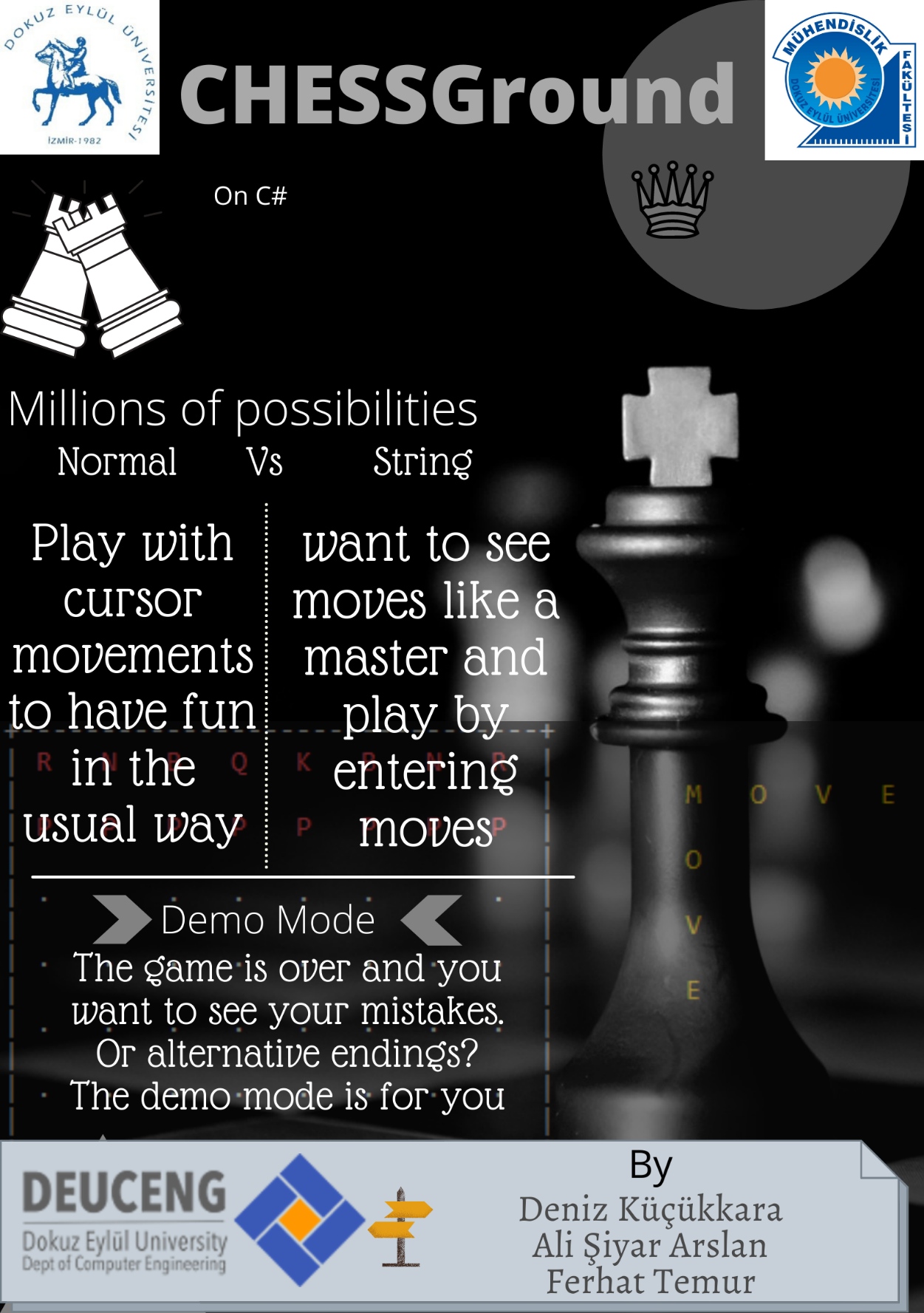
[https://www.canva.com](https://www.canva.com/)

<https://docs.microsoft.com/tr-tr/dotnet/csharp/>

<https://stackoverflow.com/>

**AppendIx A**

Poster/Web page of the Project



**AppendIx B**

Code of the Project

using System;

using System.IO;

namespace project33

{

class Program

{

static int cx, cy, temp\_b, firstplace\_x, firstplacey, tempcxpawn, tempcypawn, notation, gamecounter, KingPositionx, KingPositiony, sahk = 10, sahm = 10, ja;

static bool firstplacewriter, loop\_breaker, queenbool, queenmovements, updown, rightleft, rightcross, leftcross, Wrongpiece, enpassantm, enpassantk, Promotionbool, checkmate = false, sahka, ssahma;

static string[] moves = new string[9999];//This array for save notations to file

static void Notation(string situation, string firstletter)// This procedure for show notations in the command line

{

if (situation == "normal")//FOR NORMAL MOVES

{

if (notation % 2 == 0)

{

if (notation == 1)

notation++;

Console.SetCursorPosition(48, 4 + notation / 2);

Console.Write((notation) / 2 + 1 + ".");

Console.ForegroundColor = ConsoleColor.DarkCyan;

Console.SetCursorPosition(50, 4 + notation / 2);

Console.Write(firstletter + "" + Show1[cx] + "" + Show2[cy]);

}

if (notation % 2 == 1)

{

Console.ForegroundColor = ConsoleColor.DarkRed;

Console.SetCursorPosition(60, 4 + notation / 2);

Console.Write(firstletter + "" + Show1[cx] + "" + Show2[cy]);

}

if (firstletter == "")

moves[gamecounter] = (Show1[cx] + "" + Show2[cy] + " ");

else

moves[gamecounter] = (firstletter + Show1[firstplace\_x] + "" + Show1[cx] + "" + Show2[cy] + " ");

}

else if (situation == "promo")// FOR PROMOTION

{

if (notation % 2 == 0)

{

if (notation == 1)

notation++;

Console.SetCursorPosition(48, 4 + notation / 2);

Console.Write((notation) / 2 + 1 + ".");

Console.ForegroundColor = ConsoleColor.DarkCyan;

Console.SetCursorPosition(50, 4 + notation / 2);

Console.Write(Show1[cx] + "" + Show2[cy] + "" + firstletter);

}

if (notation % 2 == 1)

{

Console.ForegroundColor = ConsoleColor.DarkRed;

Console.SetCursorPosition(60, 4 + notation / 2);

Console.Write(Show1[cx] + "" + Show2[cy] + "" + firstletter);

}

moves[gamecounter] = (Show1[cx] + "" + Show2[cy] + firstletter + " ");

}

else if (situation == "yeme")// FOR CAPTURE

{

if (notation % 2 == 0)

{

;

Console.SetCursorPosition(48, 4 + notation / 2);

Console.Write((notation) / 2 + 1 + ".");

Console.ForegroundColor = ConsoleColor.DarkCyan;

Console.SetCursorPosition(50, 4 + notation / 2);

Console.Write(firstletter + "" + Show1[firstplace\_x] + "x" + Show1[cx] + "" + Show2[cy]);

}

if (notation % 2 == 1)

{

Console.ForegroundColor = ConsoleColor.DarkRed;

Console.SetCursorPosition(60, 4 + notation / 2);

Console.Write(firstletter + "" + Show1[firstplace\_x] + "x" + Show1[cx] + "" + Show2[cy]);

}

moves[gamecounter] = (firstletter + Show1[firstplace\_x] + "x" + "" + Show1[cx] + "" + Show2[cy] + " ");

}

else if (situation == "en")//FOR ENPASSANT

{

if (notation % 2 == 0)

{

Console.SetCursorPosition(48, 4 + notation / 2);

Console.Write((notation) / 2 + 1 + ".");

Console.ForegroundColor = ConsoleColor.DarkCyan;

Console.SetCursorPosition(50, 4 + notation / 2);

Console.Write(Show1[firstplace\_x] + "x" + Show1[cx] + "" + (Show2[tempcypawn] + "e.p."));

}

if (notation % 2 == 1)

{

Console.ForegroundColor = ConsoleColor.DarkRed;

Console.SetCursorPosition(60, 4 + notation / 2);

Console.Write(Show1[firstplace\_x] + "x" + Show1[cx] + "" + Show2[tempcypawn] + "e.p.");

}

}

else if (situation == "kısarok")//FOR ROK

{

int sayı = 0;

if (notation % 2 == 0)

{

Console.SetCursorPosition(48, 4 + notation / 2);

Console.Write((notation) / 2 + 1 + ".");

Console.ForegroundColor = ConsoleColor.DarkCyan;

sayı = 50;

}

if (notation % 2 == 1)

{

Console.ForegroundColor = ConsoleColor.DarkRed;

sayı = 60;

}

Console.SetCursorPosition(sayı, 4 + notation / 2);

Console.Write("0-0");

moves[gamecounter] = ("0-0 ");

}

else if (situation == "uzunrok")//FOR ROK

{

int sayı = 0;

if (notation % 2 == 0)

{

Console.SetCursorPosition(48, 4 + notation / 2);

Console.Write((notation) / 2 + 1 + ".");

Console.ForegroundColor = ConsoleColor.DarkCyan;

sayı = 50;

}

if (notation % 2 == 1)

{

Console.ForegroundColor = ConsoleColor.DarkRed;

sayı = 60;

}

Console.SetCursorPosition(sayı, 4 + notation / 2);

Console.Write("0-0-0");

moves[gamecounter] = ("0-0-0 ");

}

Console.ResetColor();

}

static string[] Show2 = new string[16];

static string[] Show1 = new string[33];

static string tempo1 = "E", tempo2 = "E", gamename = "";

static int tempo4 = 0, tempdenemea = 0, countera = 0;

static int[,] board2 = new int[8, 8] { {-1,-1,-1,-1,-1,-1,-1,-1 },//FOR MOVES makes the same movements as the main table

{-1,-1,-1,-1,-1,-1,-1,-1 },

{0,0,0,0,0,0,0,0 },

{0,0,0,0,0,0,0,0 },

{0,0,0,0,0,0,0,0 },

{0,0,0,0,0,0,0,0 },

{1,1,1,1,1,1,1,1 },

{1,1,1,1,1,1,1,1 }, };

static void RookCheckMate(string rook, int enemyteam)// Check Rook

{

for (int i = KingPositionx + 1; i < board.GetLength(0); i++)

{

if (board2[i, ja] == enemyteam)

break;

else if (board2[i, ja] == -enemyteam && board[i, ja] != rook)

break;

else if (board[i, ja] == rook)

{

checkmate = true;

Console.SetCursorPosition(1, 30);

Console.WriteLine("Check");

break;

}

else

checkmate = false;

}

if (KingPositionx > 0 && ja > 0)

{

for (int i = KingPositionx; i > 0; i--)

{

if (board2[i, ja] == enemyteam && (board[KingPositionx, i] != "KK" || board[KingPositionx, i] != "KM"))

break;

else if (board2[i, ja] == -enemyteam && board[i, ja] != rook)

break;

else if (board[i, ja] == rook)

{

checkmate = true;

Console.SetCursorPosition(1, 30);

Console.WriteLine("Check");

break;

}

else

checkmate = false;

}

for (int i = KingPositiony; i > 0; i--)

{

if (board2[KingPositionx, i] == enemyteam && (board[KingPositionx, i] != "KK" || board[KingPositionx, i] != "KM"))

break;

else if (board2[KingPositionx, i] == -enemyteam && board[KingPositionx, i] != rook)

break;

else if (board[KingPositionx, i] == rook)

{

checkmate = true;

Console.SetCursorPosition(1, 30);

Console.WriteLine("Check");

break;

}

else

checkmate = false;

}

for (int i = KingPositiony; i < board.GetLength(0); i++)

{

if (board2[KingPositionx, i] == enemyteam)

break;

else if (board2[KingPositionx, i] == -enemyteam && board[KingPositionx, i] != rook)

break;

else if (board[KingPositionx, i] == rook)

{

checkmate = true;

Console.SetCursorPosition(1, 30);

Console.WriteLine("Check");

break;

}

else

checkmate = false;

}

}

}

static void RealMove(int Up, int Right, int Down, int Left)

{

if (Up == 8 && cy > 1)//1UP

{

tempo4 = board2[((cy - 1) / 2) - 1, ((cx - 1) / 4)];

tempo1 = board[((cy - 1) / 2) - 1, ((cx - 1) / 4)];

Yön(cx, cy, "up");

cy -= 2;

board[((cy - 1) / 2) + 1, ((cx - 1) / 4)] = tempo2;

board2[((cy - 1) / 2) + 1, ((cx - 1) / 4)] = tempdenemea;

}

if (Right == 6 && cx < 32)//1RIGHT

{

tempo4 = board2[((cy - 1) / 2), ((cx - 1) / 4) + 1];

tempo1 = board[((cy - 1) / 2), ((cx - 1) / 4) + 1];

Yön(cx, cy, "right");

cx += 4;

board[((cy - 1) / 2), ((cx - 1) / 4) - 1] = tempo2;

board2[((cy - 1) / 2), ((cx - 1) / 4) - 1] = tempdenemea;

}

if (Down == 2 && cy < 15)//1DOWN

{

tempo4 = board2[((cy - 1) / 2) + 1, ((cx - 1) / 4)];

tempo1 = board[((cy - 1) / 2) + 1, ((cx - 1) / 4)];

Yön(cx, cy, "down");

cy += 2;

board[((cy - 1) / 2) - 1, ((cx - 1) / 4)] = tempo2;

board2[((cy - 1) / 2) - 1, ((cx - 1) / 4)] = tempdenemea;

}

if (Left == 4 && cx > 4)//1LEFT

{

tempo4 = board2[((cy - 1) / 2), ((cx - 1) / 4) - 1];

tempo1 = board[((cy - 1) / 2), ((cx - 1) / 4) - 1];

Yön(cx, cy, "left");

cx -= 4;

board[((cy - 1) / 2), ((cx - 1) / 4) + 1] = tempo2;

board2[((cy - 1) / 2), ((cx - 1) / 4) + 1] = tempdenemea;

}

}

static string[,] board = new string[8, 8] { { "RK","NK","BK","QK","KK","BK","NK","RK" },

{"PK","PK","PK", "PK","PK","PK","PK", "PK"},

{"E","E","E","E","E","E","E","E" },

{"E","E","E","E","E","E","E","E" },

{"E","E","E","E","E","E","E","E" } ,

{"E","E","E","E","E","E","E","E" } ,

{"PM","PM","PM","PM","PM","PM","PM","PM" },

{"RM","NM","BM","QM","KM","BM","NM","RM" } }; //2D array for board

static void Promotion(string teamletter)//For promotion

{

Console.SetCursorPosition(70, 1);

Console.Write("!Promotion!");

Console.SetCursorPosition(70, 3);

Console.Write("Press the letter of the piece you want to convert");

Console.SetCursorPosition(70, 5);

Console.Write("R(ROOK) Q(QUEEN) B(BISHOP) N(KNIGHT)");

ConsoleKeyInfo keyInfo2 = Console.ReadKey(true);

if (keyInfo2.Key == ConsoleKey.N)

{

board[((cy - 1) / 2), ((cx - 1) / 4)] = "N" + teamletter;

Promotionbool = true;

Notation("promo", "N");

}

else if (keyInfo2.Key == ConsoleKey.R)

{

board[((cy - 1) / 2), ((cx - 1) / 4)] = "R" + teamletter;

Promotionbool = true;

Notation("promo", "R");

}

else if (keyInfo2.Key == ConsoleKey.Q)

{

board[((cy - 1) / 2), ((cx - 1) / 4)] = "Q" + teamletter;

Promotionbool = true;

Notation("promo", "Q");

}

else if (keyInfo2.Key == ConsoleKey.B)

{

board[((cy - 1) / 2), ((cx - 1) / 4)] = "B" + teamletter;

Promotionbool = true;

Notation("promo", "B");

}

else

{

Console.SetCursorPosition(70, 7);

Console.WriteLine("WRONG KEY TRY AGAIN");

Promotionbool = false;

}

Console.SetCursorPosition(70, 1);

Console.Write(" ");

Console.SetCursorPosition(70, 3);

Console.Write(" ");

Console.SetCursorPosition(70, 5);

Console.Write(" ");

Console.SetCursorPosition(70, 7);

Console.Write(" ");

Rewrite();

}

static void Wrong\_move(int takım)

{

if (takım == 1)

Console.ForegroundColor = ConsoleColor.DarkCyan;

else if (takım == -1)

Console.ForegroundColor = ConsoleColor.DarkRed;

Console.SetCursorPosition(12, 20);

Console.WriteLine("Wrong Place Try Again.");

Console.ResetColor();

if (takım == 0)

{

Console.SetCursorPosition(12, 20);

Console.WriteLine(" ");

}

}

static void Rewrite()

{

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

{

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

{

Console.SetCursorPosition(4 \* x + 4, 2 \* y + 1);

if (board[y, x] == "E") Console.Write(".");

else if (board[y, x] == "PK")

{

Console.ForegroundColor = ConsoleColor.Red;

Console.Write("P");

Console.ResetColor();

}

else if (board[y, x] == "PM")

{

Console.ForegroundColor = ConsoleColor.Blue;

Console.Write("P");

Console.ResetColor();

}

else if (board[y, x] == "RK")

{

Console.ForegroundColor = ConsoleColor.Red;

Console.Write("R");

Console.ResetColor();

}

else if (board[y, x] == "RM")

{

Console.ForegroundColor = ConsoleColor.Blue;

Console.Write("R");

Console.ResetColor();

}

else if (board[y, x] == "NK")

{

Console.ForegroundColor = ConsoleColor.Red;

Console.Write("N");

Console.ResetColor();

}

else if (board[y, x] == "NM")

{

Console.ForegroundColor = ConsoleColor.Blue;

Console.Write("N");

Console.ResetColor();

}

else if (board[y, x] == "BK")

{

Console.ForegroundColor = ConsoleColor.Red;

Console.Write("B");

Console.ResetColor();

}

else if (board[y, x] == "BM")

{

Console.ForegroundColor = ConsoleColor.Blue;

Console.Write("B");

Console.ResetColor();

}

else if (board[y, x] == "QK")

{

Console.ForegroundColor = ConsoleColor.Red;

Console.Write("Q");

Console.ResetColor();

}

else if (board[y, x] == "QM")

{

Console.ForegroundColor = ConsoleColor.Blue;

Console.Write("Q");

Console.ResetColor();

}

else if (board[y, x] == "KK")

{

Console.ForegroundColor = ConsoleColor.Red;

Console.Write("K");

Console.ResetColor();

}

else if (board[y, x] == "KM")

{

Console.ForegroundColor = ConsoleColor.Blue;

Console.Write("K");

Console.ResetColor();

}

else Console.Write(board[y, x]);

}

}

}

static void Pawn(int firstmove, int cynum)//For pawn movements

{

ConsoleKeyInfo keyInfo2 = Console.ReadKey(true);

Wrong\_move(0);

if (keyInfo2.Key == ConsoleKey.N && firstplace\_x - cx == 0 && firstplacey - cy == 0)

Wrongpiece = true;

if (keyInfo2.Key == ConsoleKey.UpArrow)

RealMove(8, 0, 0, 0);

if (keyInfo2.Key == ConsoleKey.RightArrow)

RealMove(0, 6, 0, 0);

if (keyInfo2.Key == ConsoleKey.DownArrow)

RealMove(0, 0, 2, 0);

if (keyInfo2.Key == ConsoleKey.LeftArrow)

RealMove(0, 0, 0, 4);

if (keyInfo2.Key == ConsoleKey.Enter)

{

if (firstplacey == firstmove && Math.Abs(cx - firstplace\_x) == 0 && Math.Abs(cy - firstplacey) == 4 && tempo4 == 0)

{

tempcxpawn = cx;

tempcypawn = cy;

if (cynum / 2 == 1)

enpassantk = true;

if (cynum / 2 == -1)

enpassantm = true;

loop\_breaker = true;

Notation("normal", "");

}

else if (Math.Abs(cx - firstplace\_x) == 0 && Math.Abs(cy - firstplacey) == 2 && tempo4 == 0)

{

loop\_breaker = true;

Notation("normal", "");

}

else if (Math.Abs(cx - firstplace\_x) == 4 && Math.Abs(cy - firstplacey) == 2 && tempo4 == (cynum) && tempo1 == "E")

{

Notation("en", "");

board[((cy - 1) / 2) - (cynum / 2), ((cx - 1) / 4)] = "E";

board2[((cy - 1) / 2) - (cynum / 2), ((cx - 1) / 4)] = 0;

loop\_breaker = true;

}

else if (Math.Abs(cx - firstplace\_x) == 4 && Math.Abs(cy - firstplacey) == 2 && tempo4 == (cynum / 2))

{

if (cy != 1 && cy != 15)

Notation("yeme", "");

loop\_breaker = true;

}

else

Wrong\_move(-(cynum / 2));

}

Rewrite();

tempo2 = tempo1;

tempdenemea = tempo4;

}

static void King(int takım, string takımkalesi)//For king movements and ROK

{

ConsoleKeyInfo keyInfo2 = Console.ReadKey(true);

Wrong\_move(0);

if (keyInfo2.Key == ConsoleKey.K && board2[((firstplacey - 1) / 2), ((firstplace\_x - 1) / 4) + 1] == 0 && board2[((firstplacey - 1) / 2), ((firstplace\_x - 1) / 4) + 2] == 0 && board[((firstplacey - 1) / 2), ((firstplace\_x - 1) / 4) + 3] == "R" + takımkalesi)

{

RealMove(0, 6, 0, 0);

RealMove(0, 6, 0, 0);

board[((firstplacey - 1) / 2), ((firstplace\_x - 1) / 4) + 1] = "R" + takımkalesi;

board2[((firstplacey - 1) / 2), ((firstplace\_x - 1) / 4) + 1] = takım;

board[((firstplacey - 1) / 2), ((firstplace\_x - 1) / 4) + 3] = "E";

board2[((firstplacey - 1) / 2), ((firstplace\_x - 1) / 4) + 3] = 0;

Notation("kısarok", "");

loop\_breaker = true;

}

if (keyInfo2.Key == ConsoleKey.U && board2[((firstplacey - 1) / 2), ((firstplace\_x - 1) / 4) - 1] == 0 && board2[((firstplacey - 1) / 2), ((firstplace\_x - 1) / 4) - 2] == 0 && board2[((firstplacey - 1) / 2), ((firstplace\_x - 1) / 4) - 3] == 0 && board[((firstplacey - 1) / 2), ((firstplace\_x - 1) / 4) - 4] == "R" + takımkalesi)

{

RealMove(0, 0, 0, 4);

RealMove(0, 0, 0, 4);

board[((firstplacey - 1) / 2), ((firstplace\_x - 1) / 4) - 1] = "R" + takımkalesi;

board2[((firstplacey - 1) / 2), ((firstplace\_x - 1) / 4) - 1] = takım;

board[((firstplacey - 1) / 2), ((firstplace\_x - 1) / 4) - 4] = "E";

board2[((firstplacey - 1) / 2), ((firstplace\_x - 1) / 4) - 4] = 0;

Notation("uzunrok", "");

loop\_breaker = true;

}

if (keyInfo2.Key == ConsoleKey.N && firstplace\_x - cx == 0 && firstplacey - cy == 0)

Wrongpiece = true;

if (keyInfo2.Key == ConsoleKey.UpArrow)

RealMove(8, 0, 0, 0);

if (keyInfo2.Key == ConsoleKey.RightArrow)

RealMove(0, 6, 0, 0);

if (keyInfo2.Key == ConsoleKey.DownArrow)

RealMove(0, 0, 2, 0);

if (keyInfo2.Key == ConsoleKey.LeftArrow)

RealMove(0, 0, 0, 4);

if (keyInfo2.Key == ConsoleKey.Enter)

{

if (Math.Abs(cx - firstplace\_x) == 0 && Math.Abs(cy - firstplacey) == 2 && tempo4 != takım)

{

if (tempo4 == -takım)

Notation("yeme", "N");

else

Notation("normal", "N");

loop\_breaker = true;

}

else if (Math.Abs(cx - firstplace\_x) == 4 && Math.Abs(cy - firstplacey) == 0 && tempo4 != takım)

{

if (tempo4 == -takım)

Notation("yeme", "N");

else

Notation("normal", "N");

loop\_breaker = true;

}

if (Math.Abs(cx - firstplace\_x) == 4 && Math.Abs(cy - firstplacey) == 2 && tempo4 != takım)

{

if (tempo4 == -takım)

Notation("yeme", "N");

else

Notation("normal", "N");

loop\_breaker = true;

}

else

Wrong\_move(takım);

}

Rewrite();

tempo2 = tempo1;

tempdenemea = tempo4;

}

static void Horse(int Team)//For horse movements

{

ConsoleKeyInfo keyInfo2 = Console.ReadKey(true);

Wrong\_move(0);

if (keyInfo2.Key == ConsoleKey.N && firstplace\_x - cx == 0 && firstplacey - cy == 0)

Wrongpiece = true;

if (keyInfo2.Key == ConsoleKey.UpArrow)

RealMove(8, 0, 0, 0);

if (keyInfo2.Key == ConsoleKey.RightArrow)

RealMove(0, 6, 0, 0);

if (keyInfo2.Key == ConsoleKey.DownArrow)

RealMove(0, 0, 2, 0);

if (keyInfo2.Key == ConsoleKey.LeftArrow)

RealMove(0, 0, 0, 4);

if (keyInfo2.Key == ConsoleKey.Enter)

{

if (Math.Abs(cx - firstplace\_x) == 8 && Math.Abs(cy - firstplacey) == 2 && tempo4 != Team)// \_\_| yada |\_\_

{

if (tempo4 == -Team)

Notation("yeme", "N");

else

Notation("normal", "N");

loop\_breaker = true;

}

else if (Math.Abs(cx - firstplace\_x) == 4 && Math.Abs(cy - firstplacey) == 4 && tempo4 != Team)// L yada \_|

{

if (tempo4 == -Team)

Notation("yeme", "N");

else

Notation("normal", "N");

loop\_breaker = true;

}

else

Wrong\_move(Team);

}

Rewrite();

tempo2 = tempo1;

tempdenemea = tempo4;

}

static void Rook(int enemyteam)

{

ConsoleKeyInfo keyInfo2 = Console.ReadKey(true);

Wrong\_move(0);

if (keyInfo2.Key == ConsoleKey.N && firstplace\_x - cx == 0 && firstplacey - cy == 0)

Wrongpiece = true;

if (temp\_b == enemyteam && countera > 1)

queenbool = false;

else if (rightleft == false && cy > 1 && keyInfo2.Key == ConsoleKey.UpArrow && board2[((cy - 1) / 2) - 1, (cx - 1) / 4] != -enemyteam)

{

temp\_b = board2[((cy - 1) / 2) - 1, (cx - 1) / 4];

updown = true;

Yön(cx, cy, "up");

cy -= 2;

Console.SetCursorPosition(cx, cy);

}

else if (rightleft == false && cy < 15 && keyInfo2.Key == ConsoleKey.DownArrow && board2[((cy - 1) / 2) + 1, (cx - 1) / 4] != -enemyteam)

{

temp\_b = board2[((cy - 1) / 2) + 1, (cx - 1) / 4];

updown = true;

Yön(cx, cy, "down");

cy += 2;

Console.SetCursorPosition(cx, cy);

}

else if (updown == false && keyInfo2.Key == ConsoleKey.RightArrow && cx < 30 && board2[((cy - 1) / 2), ((cx - 1) / 4) + 1] != -enemyteam)

{

temp\_b = board2[((cy - 1) / 2), ((cx - 1) / 4) + 1];

rightleft = true;

Yön(cx, cy, "right");

cx += 4;

Console.SetCursorPosition(cx, cy);

}

else if (updown == false && cx > 4 && keyInfo2.Key == ConsoleKey.LeftArrow && board2[((cy - 1) / 2), ((cx - 1) / 4) - 1] != -enemyteam)

{

temp\_b = board2[((cy - 1) / 2), ((cx - 1) / 4) - 1];

rightleft = true;

Yön(cx, cy, "left");

cx -= 4;

Console.SetCursorPosition(cx, cy);

}

if (firstplace\_x - cx == 0 && firstplacey - cy == 0)

{

queenmovements = true;

updown = false;

rightleft = false;

}

if (keyInfo2.Key == ConsoleKey.Enter)

{

if (firstplace\_x - cx == 0 && firstplacey - cy == 0)

Wrong\_move(-enemyteam);

else

queenbool = false;

}

Rewrite();

tempo2 = tempo1;

tempdenemea = tempo4;

}

static void Bishop(int enemy\_team)

{

ConsoleKeyInfo keyInfo2 = Console.ReadKey(true);

Wrong\_move(0);

if (keyInfo2.Key == ConsoleKey.N && firstplace\_x - cx == 0 && firstplacey - cy == 0)

Wrongpiece = true;

if (temp\_b == enemy\_team && countera > 1)

queenbool = false;

else if (leftcross == false && cy > 1 && cx < 30 && keyInfo2.Key == ConsoleKey.E && board2[((cy - 1) / 2) - 1, ((cx - 1) / 4) + 1] != -enemy\_team)//sagcapraz yukarı

{

temp\_b = board2[((cy - 1) / 2) - 1, ((cx - 1) / 4) + 1];

Console.SetCursorPosition(46, 5);

rightcross = true;

Yön(cx, cy, "ur");

cy -= 2;

cx += 4;

Console.SetCursorPosition(cx, cy);

}

else if (leftcross == false && cy < 15 && cx > 4 && keyInfo2.Key == ConsoleKey.Z && board2[((cy - 1) / 2) + 1, ((cx - 1) / 4) - 1] != -enemy\_team)//sagcaprz asagi

{

temp\_b = board2[((cy - 1) / 2) + 1, ((cx - 1) / 4) - 1];

Console.SetCursorPosition(46, 5);

rightcross = true;

Yön(cx, cy, "dl");

cy += 2;

cx -= 4;

Console.SetCursorPosition(cx, cy);

}

else if (rightcross == false && cx > 4 && keyInfo2.Key == ConsoleKey.Q && board2[((cy - 1) / 2) - 1, ((cx - 1) / 4) - 1] != -enemy\_team)//solcapraz yukarı

{

temp\_b = board2[((cy - 1) / 2) - 1, ((cx - 1) / 4) - 1];

Console.SetCursorPosition(46, 5);

leftcross = true;

Yön(cx, cy, "ul");

cy -= 2;

cx -= 4;

Console.SetCursorPosition(cx, cy);

}

else if (rightcross == false && cy < 15 && cx < 30 && keyInfo2.Key == ConsoleKey.C && board2[((cy - 1) / 2) + 1, ((cx - 1) / 4) + 1] != -enemy\_team)//solcaprz asagi

{

temp\_b = board2[((cy - 1) / 2) + 1, ((cx - 1) / 4) + 1];

Console.SetCursorPosition(46, 5);

leftcross = true;

Yön(cx, cy, "dr");

cy += 2;

cx += 4;

Console.SetCursorPosition(cx, cy);

}

if (firstplace\_x - cx == 0 && firstplacey - cy == 0)

{

queenmovements = false;

rightcross = false;

leftcross = false;

}

if (keyInfo2.Key == ConsoleKey.Enter)

{

if (firstplace\_x - cx == 0 && firstplacey - cy == 0)

Wrong\_move(-enemy\_team);

else

queenbool = false;

}

}

static void Queen(bool rightleft, bool updown, int enemyteam, bool rightcross, bool leftcross)

{

while (queenbool == true)

{

if (Wrongpiece == true)

break;

if (queenmovements == true)

Bishop(enemyteam);

if (queenmovements == false)

Rook(enemyteam);

}

if (Wrongpiece != true)

{

if (temp\_b == enemyteam)

Notation("yeme", "Q");

else

Notation("normal", "Q");

}

}

static void Yön(int x, int y, string finderr)

{

int Yonfinder = 0;

int Yınfinder2 = 0;

if (finderr == "up")

Yonfinder = -1;

if (finderr == "down")

Yonfinder = 1;

if (finderr == "right")

Yınfinder2 = +1;

if (finderr == "left")

Yınfinder2 = -1;

if (finderr == "ur")

{

Yonfinder = -1;//y

Yınfinder2 = 1;//x

}

if (finderr == "dl")

{

Yonfinder = 1;

Yınfinder2 = -1;

}

if (finderr == "ul")

{

Yonfinder = -1;

Yınfinder2 = -1;

}

if (finderr == "dr")

{

Yonfinder = 1;

Yınfinder2 = 1;

}

board[((y - 1) / 2) + Yonfinder, ((x - 1) / 4) + Yınfinder2] = board[(y - 1) / 2, ((x - 1) / 4)];

board[(y - 1) / 2, (x - 1) / 4] = "E";

board2[((y - 1) / 2) + Yonfinder, ((x - 1) / 4) + Yınfinder2] = board2[(y - 1) / 2, ((x - 1) / 4)];

board2[(y - 1) / 2, (x - 1) / 4] = 0;

Rewrite();

}

static void Yön2(int x, int y, int X, int Y)

{

board[Y, X] = board[y, x];

board[y, x] = "E";

board2[Y, X] = board2[y, x];

board2[y, x] = 0;

Rewrite();

}

static void Yön3(int x, int y, int X, int Y)

{

string ffirst = board[y, x];

board[y, x] = board[Y, X];

board[Y, X] = ffirst;

Rewrite();

}

static void Main(string[] args)

{

string dosya = "";

Console.WriteLine("Which game style do you want to play? 1-Play Mode 2-String Mode 3-Demo Mode");//For deciding game mode

string check = Console.ReadLine();

if (check == "1" || check == "2")

{

Console.WriteLine("Create Game .txt name");//for creating txt file

gamename = Console.ReadLine();

StreamWriter f1 = File.CreateText(gamename + ".txt");

f1.Close();

}

if (check == "3")

{

Console.WriteLine("Enter txt file name");

dosya = Console.ReadLine();

}

Console.Clear();

Console.SetCursorPosition(44, 2);

Console.ForegroundColor = ConsoleColor.DarkYellow;

Console.Write("M O V E");

Console.SetCursorPosition(44, 4);

Console.Write("O");

Console.SetCursorPosition(44, 6);

Console.Write("V");

Console.SetCursorPosition(44, 8);

Console.Write("E");

Console.ResetColor();

Console.SetCursorPosition(0, 0);

int count = 1;

string[] Shower2 = new string[] { "a", "b", "c", "d", "e", "f", "g", "h" };

for (int i = 0; i < Show1.Length; i++)

{

if (i % 4 == 0 && i > 3)

Show1[i] = Shower2[i / 4 - 1];

else

Show1[i] = null;

}

string[] Show22 = new string[] { "8", "7", "6", "5", "4", "3", "2", "1" };

for (int i = 1; i < Show2.Length; i += 2)

{

Show2[i] = Show22[i - count];

count++;

}

cx = 4;

cy = 1;

int temp2cypawn = 200, temp2cxpawn = 200;

int k = 9;

for (int i = 0; i <= 16; i++)

{

System.Threading.Thread.Sleep(50);

if (i == 0 || i == 16) Console.WriteLine(" +--------------------------------+");

if (i % 2 == 1)

{

k--;

Console.WriteLine(k + " | |");

}

else

{

if (i < 16 && i > 0 && i % 2 == 0) Console.WriteLine(" | |");

}

}

int game\_counter = 0;

Rewrite();

Console.SetCursorPosition(3, 17);

Console.Write(" a b c d e f g h ");

int ccount = 0;

string c = "";

Console.SetCursorPosition(0, 20);

if (check == "2" || check == "3")

{

Console.SetCursorPosition(0, 20);

Console.WriteLine(" ");

Console.SetCursorPosition(0, 21);

Console.WriteLine(" ");

while (true)

{

//-----

string a = "";

Console.SetCursorPosition(40, 15);

Console.WriteLine(" ");

Console.SetCursorPosition(65, 20);

Console.WriteLine(" ");

if (game\_counter % 2 == 0)//For show trun

{

Console.SetCursorPosition(40, 15);

Console.ForegroundColor = ConsoleColor.Blue;

Console.WriteLine("play turn is in blue");

Console.ResetColor();

}

if (game\_counter % 2 == 1)//For show turn

{

Console.SetCursorPosition(40, 15);

Console.ForegroundColor = ConsoleColor.Red;

Console.WriteLine("play turn is in red");

Console.ResetColor();

}

Console.SetCursorPosition(0, 23);

Console.WriteLine(" ");

if (check == "2")//string mode

{

Console.SetCursorPosition(0, 22);

Console.WriteLine("enter a string");

a = Console.ReadLine();

c = a;

}

else if (check == "3")//Demo mod

{

StreamReader f = File.OpenText(dosya + ".txt");

string sstr = f.ReadLine();

f.Close();

string[] words = sstr.Split(' ');

ConsoleKeyInfo keyInfo = Console.ReadKey(true);

if (keyInfo.Key == ConsoleKey.Spacebar)

{

a = words[ccount];

c = a;

ccount++;

}

if (keyInfo.Key == ConsoleKey.Enter)

{

check = "1";

break;

}

}

int yy = 0, xx = 0, xxx = 0, yyy = 0;

int locx = 0, line = 0;

bool flag = false, flag2 = true;

string from = "", color = "";

string where = "", column = "", columnf = "", fil\_kontrol = "";

for (int i = 0; i < a.Length; i++)

{

if (a[i] == 'x')

{

flag = true;

locx = i;

break;

}

}

if (flag == true)//capture

{

if (a.Length == 5)//break the string

{

where = a.Substring(a.Length - 2, 2);

from = a.Substring(0, 2);

column = where.Substring(0, 1);

line = Convert.ToInt32(where.Substring(1, 1));

columnf = from.Substring(1, 1);

}

if (a.Length == 4 && locx == 1)//break the string for pawn

{

where = a.Substring(a.Length - 2, 2);

from = a.Substring(0, 1);

column = where.Substring(0, 1);

line = Convert.ToInt32(where.Substring(1, 1));

columnf = from.Substring(0, 1);

}

if (locx == 1)

from = a.Substring(0, 1);

else if (locx == 2)

from = a.Substring(0, 2);

where = a.Substring(a.Length - 2, 2);

}

else if (flag == false)//break the string for no capture

{

if (a.Length == 2)//pawn

{

where = a.Substring(0, 2);

column = where.Substring(0, 1);

line = Convert.ToInt32(where.Substring(1, 1));

}

else if (a.Length == 3)

{

where = a.Substring(a.Length - 2, 2);

from = a.Substring(0, 1);

column = where.Substring(0, 1);

line = Convert.ToInt32(where.Substring(1, 1));

}

else if (a.Length == 4)

{

where = a.Substring(a.Length - 2, 2);

from = a.Substring(0, 2);

column = where.Substring(0, 1);

columnf = from.Substring(1, 1);

line = Convert.ToInt32(where.Substring(1, 1));

}

}

if (columnf == "a")//first point

{

xx = 0;

}

if (columnf == "b")

{

xx = 1;

}

if (columnf == "c")

{

xx = 2;

}

if (columnf == "d")

{

xx = 3;

}

if (columnf == "e")

{

xx = 4;

}

if (columnf == "f")

{

xx = 5;

}

if (columnf == "g")

{

xx = 6;

}

if (columnf == "h")

{

xx = 7;

}

if (column == "a")//last point

{

xxx = 0;

}

if (column == "b")

{

xxx = 1;

}

if (column == "c")

{

xxx = 2;

}

if (column == "d")

{

xxx = 3;

}

if (column == "e")

{

xxx = 4;

}

if (column == "f")

{

xxx = 5;

}

if (column == "g")

{

xxx = 6;

}

if (column == "h")

{

xxx = 7;

}

yyy = 8 - line;

Console.SetCursorPosition(0, 55);

Console.WriteLine(xxx + " last " + yyy);//--

if (game\_counter % 2 == 0)//COLOR--

{

if (a == "0-0" || a == "0-0-0")

{

color = color + a + "M";

}

else if (from == "")

{

color = color + "PM";

}

else if (from == from.Substring(0, 1).ToLower())

{

color = color + "PM";

}

else

{

color = color + from.Substring(0, 1) + "M";

}

}

if (game\_counter % 2 == 1)

{

if (a == "0-0" || a == "0-0-0")

{

color = color + a + "M";

}

else if (from == "")

{

color = color + "PK";

}

else if (from == from.Substring(0, 1).ToLower())

{

color = color + "PK";

}

else

{

color = color + from.Substring(0, 1) + "K";

}

}

for (int i = 0; i < 8; i++)//first point calculator (y)

{

if (a.Length == 2)

{

if (board[i, xxx] == color)//find the given game element in that column

{

yy = i;

if (flag == false && a.Length == 2)

{

xx = xxx;

}

}

}

else

{

if (board[i, xx] == color)//find the given game element in that column

{

yy = i;

if (flag == false && a.Length == 2)

{

xx = xxx;

}

}

}

}

if (color == "PM" && flag == false && board2[yyy, xxx] == 0)//pawn MAVİ no capture

{

if (yy == 6 && (yy - yyy == 2))//check two steps ahead for the first move

{

Yön2(xx, yy, xxx, yyy);

game\_counter++;

}

if (yy - yyy == 1)//check one steps ahead for the move

{

Yön2(xx, yy, xxx, yyy);

game\_counter++;

}

if (yyy == 0)

{

Console.SetCursorPosition(65, 20);

Console.Write("Enter a game item you want :");

string b = Console.ReadLine();

board[yyy, xxx] = b + "M";

Rewrite();

}

}

if (color == "PM" && flag == true)//pawn MAVİ capture

{

if (yy - yyy == 1 && (((xxx - xx == 1 && board2[yyy, xxx] == -1)) || ((xxx - xx == -1 && board2[yyy, xxx] == -1))))

{

Yön2(xx, yy, xxx, yyy);

game\_counter++;

}

Console.ForegroundColor = ConsoleColor.DarkCyan;

Console.SetCursorPosition(48, 4 + game\_counter / 2);

Console.WriteLine(game\_counter + 1 / 2 + ". " + c);

Console.ResetColor();

}

if (color == "PK" && flag == false && board2[yyy, xxx] == 0)//pawn KIRMIZI no capture

{

if (yy == 1 && (yy - yyy == -2))//check two steps ahead for the first move

{

Yön2(xx, yy, xxx, yyy);

game\_counter++;

}

if (yy - yyy == -1)//check one steps ahead for the move

{

Yön2(xx, yy, xxx, yyy);

game\_counter++;

}

if (yyy == 7)

{

Console.SetCursorPosition(65, 20);

Console.Write("Enter a game item you want :");

string b = Console.ReadLine();

board[yyy, xxx] = b + "K";

Rewrite();

}

}

if (color == "PK" && flag == true && board2[yyy, xxx] == 1)//pawn KIRMIZI capture

{

if (yy - yyy == -1 && xxx - xx == 1)//check two steps ahead for the first move

{

Yön2(xx, yy, xxx, yyy);

game\_counter++;

}

if (yy - yyy == -1 && xxx - xx == -1)//check one steps ahead for the move

{

Yön2(xx, yy, xxx, yyy);

game\_counter++;

}

}

if (color == "NM" && flag == false && ((Math.Abs(yyy - yy) == 2 && Math.Abs(xxx - xx) == 1) || (Math.Abs(yyy - yy) == 1 && Math.Abs(xxx - xx) == 2)) && board2[yyy, xxx] == 0)// mavi at no capture

{

Yön2(xx, yy, xxx, yyy);

game\_counter++;

}

if (color == "NM" && flag == true && ((Math.Abs(yyy - yy) == 2 && Math.Abs(xxx - xx) == 1) || (Math.Abs(yyy - yy) == 1 && Math.Abs(xxx - xx) == 2)) && board2[yyy, xxx] == -1)// mavi at capture

{

Yön2(xx, yy, xxx, yyy);

game\_counter++;

}

if (color == "NK" && flag == false && ((Math.Abs(yyy - yy) == 2 && Math.Abs(xxx - xx) == 1) || (Math.Abs(yyy - yy) == 1 && Math.Abs(xxx - xx) == 2)) && board2[yyy, xxx] == 0)// kırmızı at no capture

{

Yön2(xx, yy, xxx, yyy);

game\_counter++;

}

if (color == "NK" && flag == true && ((Math.Abs(yyy - yy) == 2 && Math.Abs(xxx - xx) == 1) || (Math.Abs(yyy - yy) == 1 && Math.Abs(xxx - xx) == 2)) && board2[yyy, xxx] == 1)// kırmızı at capture

{

Yön2(xx, yy, xxx, yyy);

game\_counter++;

}

if ((color == "RM" || color == "RK") && ((flag == false && board2[yyy, xxx] == 0) || (flag == true && board2[yyy, xxx] == -1 && board2[yy, xx] == 1) || (flag == true && board2[yyy, xxx] == 1 && board2[yy, xx] == -1)))// kale

{

int bigger, lower;

if (yy == yyy && xx != xxx)

{

if (xx > xxx)

{

bigger = xx;

lower = xxx;

}

else

{

bigger = xxx;

lower = xx;

}

for (int i = lower + 1; i < bigger; i++)

{

if (board2[yy, i] != 0)

{

flag2 = false;

}

}

if (flag2 == true)//check if it can go

{

Yön2(xx, yy, xxx, yyy);

game\_counter++;

}

}

if (yy != yyy && xx == xxx)

{

if (yy > yyy)

{

bigger = yy;

lower = yyy;

}

else

{

bigger = yyy;

lower = yy;

}

for (int i = lower + 1; i < bigger; i++)

{

if (board2[i, xx] != 0)

{

flag2 = false;

}

}

if (flag2 == true)//check if it can go

{

Yön2(xx, yy, xxx, yyy);

game\_counter++;

}

}

}

if ((color == "BM" || color == "BK") && ((flag == false && board2[yyy, xxx] == 0) || (flag == true && board2[yyy, xxx] == -1 && board2[yy, xx] == 1) || (flag == true && board2[yyy, xxx] == 1 && board2[yy, xx] == -1)))// fil

{

int biggerx, lowerx, biggery, lowery;

if (Math.Abs(xxx - xx) == Math.Abs(yyy - yy))

{

if (xx > xxx)//direction detection

{

biggerx = xx;

lowerx = xxx;

fil\_kontrol = "sol";

}

else//direction detection

{

biggerx = xxx;

lowerx = xx;

fil\_kontrol = "sag";

}

if (yy > yyy)//direction detection

{

biggery = yy;

lowery = yyy;

}

else//direction detection

{

biggery = yyy;

lowery = yy;

}

if (fil\_kontrol == "sag")

{

int j = biggery - 1;

for (int i = lowerx + 1; i < biggerx; i++)

{

if (board2[j, i] != 0)

{

flag2 = false;

}

j--;

if (j == lowery)

{

break;

}

}

}

if (fil\_kontrol == "sol")

{

int j = lowery + 1;

for (int i = lowerx + 1; i < biggerx; i++)

{

if (board2[j, i] != 0)

{

flag2 = false;

}

j++;

if (j == biggery)

{

break;

}

}

}

if (flag2 == true)//check if it can go

{

Yön2(xx, yy, xxx, yyy);

game\_counter++;

}

}

}

if ((color == "QM" || color == "QK") && ((flag == false && board2[yyy, xxx] == 0) || (flag == true && board2[yyy, xxx] == -1 && board2[yy, xx] == 1) || (flag == true && board2[yyy, xxx] == 1 && board2[yy, xx] == -1)))// queen

{

int biggerx, lowerx, biggery, lowery;

if (Math.Abs(xxx - xx) == Math.Abs(yyy - yy))

{

if (xx > xxx)

{

biggerx = xx;

lowerx = xxx;

fil\_kontrol = "sol";

}

else

{

biggerx = xxx;

lowerx = xx;

fil\_kontrol = "sag";

}

if (yy > yyy)

{

biggery = yy;

lowery = yyy;

}

else

{

biggery = yyy;

lowery = yy;

}

if (fil\_kontrol == "sag")

{

int j = biggery - 1;

for (int i = lowerx + 1; i < biggerx; i++)

{

if (board2[j, i] != 0)

{

flag2 = false;

}

j--;

if (j == lowery)

{

break;

}

}

}

if (fil\_kontrol == "sol")

{

int j = lowery + 1;

for (int i = lowerx + 1; i < biggerx; i++)

{

if (board2[j, i] != 0)

{

flag2 = false;

}

j++;

if (j == biggery)

{

break;

}

}

}

if (flag2 == true)//check if it can go

{

Yön2(xx, yy, xxx, yyy);

game\_counter++;

}

}

else

{

int bigger, lower;

if (yy == yyy && xx != xxx)

{

if (xx > xxx)

{

bigger = xx;

lower = xxx;

}

else

{

bigger = xxx;

lower = xx;

}

for (int i = lower + 1; i < bigger; i++)

{

if (board2[yy, i] != 0)

{

flag2 = false;

}

}

if (flag2 == true)//check if it can go

{

Yön2(xx, yy, xxx, yyy);

game\_counter++;

}

}

if (yy != yyy && xx == xxx)

{

if (yy > yyy)

{

bigger = yy;

lower = yyy;

}

else

{

bigger = yyy;

lower = yy;

}

for (int i = lower + 1; i < bigger; i++)

{

if (board2[i, xx] != 0)

{

flag2 = false;

}

}

if (flag2 == true)//check if it can go

{

Yön2(xx, yy, xxx, yyy);

game\_counter++;

}

}

}

}

if ((color == "KM" || color == "KK") && ((flag == false && board2[yyy, xxx] == 0) || (flag == true && board2[yyy, xxx] == -1 && board2[yy, xx] == 1) || (flag == true && board2[yyy, xxx] == 1 && board2[yy, xx] == -1)))// king

{

if ((yy == yyy && xx != xxx && Math.Abs(xx - xxx) == 1) || (yy != yyy && xx == xxx && Math.Abs(yy - yyy) == 1))//check if it can go

{

Yön2(xx, yy, xxx, yyy);

game\_counter++;

}

if (yy != yyy && xx != xxx && Math.Abs(yy - yyy) == 1 && Math.Abs(xx - xxx) == 1)//check if it can go

{

Yön2(xx, yy, xxx, yyy);

game\_counter++;

}

}

if ((color == "0-0M") && board2[7, 5] == 0 && board2[7, 6] == 0 && board2[7, 7] == 1 && board[7, 7] == "RM")//rok mavi kısa

{

Yön3(4, 7, 7, 7);

game\_counter++;

}

if ((color == "0-0K") && board2[0, 5] == 0 && board2[0, 6] == 0 && board2[0, 7] == -1 && board[0, 7] == "RK")//rok kırmızı kısa

{

Yön3(4, 0, 7, 0);

game\_counter++;

}

if ((color == "0-0-0M") && board2[7, 1] == 0 && board2[7, 2] == 0 && board2[7, 3] == 0 && board2[7, 0] == 1 && board[7, 0] == "RM")//rok mavi uzun

{

Yön3(4, 7, 0, 7);

game\_counter++;

}

if ((color == "0-0-0K") && board2[0, 1] == 0 && board2[0, 2] == 0 && board2[0, 3] == 0 && board2[0, 0] == -1 && board[0, 0] == "RK")//rok kırmızı uzun

{

Yön3(4, 0, 0, 0);

game\_counter++;

}

if (game\_counter % 2 == 1)

{

Console.SetCursorPosition(48, 4 + game\_counter / 2);

Console.WriteLine((game\_counter + 1) / 2 + ". ");

Console.ForegroundColor = ConsoleColor.DarkCyan;

Console.SetCursorPosition(50, 4 + game\_counter / 2);

Console.WriteLine(c);

StreamWriter f3 = File.AppendText(gamename + ".txt");//For adding notations to file

f3.Write(c + " ");

f3.Close();

}

else if (game\_counter % 2 == 0)

{

Console.ForegroundColor = ConsoleColor.DarkRed;

Console.SetCursorPosition(60, 4 + (game\_counter / 2) - 1);

Console.WriteLine(c);

StreamWriter f3 = File.AppendText(gamename + ".txt");//For adding notations to file

f3.Write(c + " ");

f3.Close();

}

Console.ResetColor();

}

}

if (check == "1")

{

while (true)

{

Console.SetCursorPosition(0, 20);

Console.WriteLine(" ");

Console.SetCursorPosition(0, 21);

Console.WriteLine(" ");

if (game\_counter % 2 == 0)

{

Console.SetCursorPosition(12, 19);

Console.ForegroundColor = ConsoleColor.DarkCyan;

Console.WriteLine("Blue's Turn");

if (enpassantm == true)//For checking enpassant

if (board2[(((temp2cypawn - 1) / 2) + 1), ((temp2cxpawn - 1) / 4)] == 2)

board2[(((temp2cypawn - 1) / 2) + 1), ((temp2cxpawn - 1) / 4)] = 0;

enpassantm = false;

if (enpassantk == true)

{

board2[(((tempcypawn - 1) / 2) - 1), ((tempcxpawn - 1) / 4)] = -2;

temp2cypawn = tempcypawn;

temp2cxpawn = tempcxpawn;

}

}

else

{

Console.SetCursorPosition(12, 19);

Console.ForegroundColor = ConsoleColor.DarkRed;

Console.WriteLine(" Red's Turn");

if (enpassantk == true)

if (board2[(((temp2cypawn - 1) / 2) - 1), ((temp2cxpawn - 1) / 4)] == -2)

board2[(((temp2cypawn - 1) / 2) - 1), ((temp2cxpawn - 1) / 4)] = 0;

enpassantk = false;

if (enpassantm == true)

{

board2[(((tempcypawn - 1) / 2) + 1), ((tempcxpawn - 1) / 4)] = 2;

temp2cypawn = tempcypawn;

temp2cxpawn = tempcxpawn;

}

}

Console.ResetColor();

for (int y = 0; y < 8; y++)//For writing board

{

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

{

Console.SetCursorPosition(4 \* x + 4, 2 \* y + 1);

if (board[y, x] == "E") Console.Write(".");

else if (board[y, x] == "PK")

{

Console.ForegroundColor = ConsoleColor.Red;

Console.Write("P");

Console.ResetColor();

}

else if (board[y, x] == "PM")

{

Console.ForegroundColor = ConsoleColor.Blue;

Console.Write("P");

Console.ResetColor();

}

else if (board[y, x] == "RK")

{

Console.ForegroundColor = ConsoleColor.Red;

Console.Write("R");

Console.ResetColor();

}

else if (board[y, x] == "RM")

{

Console.ForegroundColor = ConsoleColor.Blue;

Console.Write("R");

Console.ResetColor();

}

else if (board[y, x] == "NK")

{

Console.ForegroundColor = ConsoleColor.Red;

Console.Write("N");

Console.ResetColor();

}

else if (board[y, x] == "NM")

{

Console.ForegroundColor = ConsoleColor.Blue;

Console.Write("N");

Console.ResetColor();

}

else if (board[y, x] == "BK")

{

Console.ForegroundColor = ConsoleColor.Red;

Console.Write("B");

Console.ResetColor();

}

else if (board[y, x] == "BM")

{

Console.ForegroundColor = ConsoleColor.Blue;

Console.Write("B");

Console.ResetColor();

}

else if (board[y, x] == "QK")

{

Console.ForegroundColor = ConsoleColor.Red;

Console.Write("Q");

Console.ResetColor();

}

else if (board[y, x] == "QM")

{

Console.ForegroundColor = ConsoleColor.Blue;

Console.Write("Q");

Console.ResetColor();

}

else if (board[y, x] == "KK")

{

Console.ForegroundColor = ConsoleColor.Red;

Console.Write("K");

Console.ResetColor();

}

else if (board[y, x] == "KM")

{

Console.ForegroundColor = ConsoleColor.Blue;

Console.Write("K");

Console.ResetColor();

}

else Console.Write(board[y, x]);

}

}

Console.SetCursorPosition(cx, cy);

Console.CursorVisible = true;

ConsoleKeyInfo keyInfo = Console.ReadKey(true);//For normal movements

if (keyInfo.Key == ConsoleKey.RightArrow && cx < 29)

cx += 4;

if (keyInfo.Key == ConsoleKey.LeftArrow && cx > 4)

cx -= 4;

if (keyInfo.Key == ConsoleKey.UpArrow && cy > 2)

cy -= 2;

if (keyInfo.Key == ConsoleKey.DownArrow && cy < 15)

cy += 2;

Console.SetCursorPosition(cx, cy);

if (keyInfo.Key == ConsoleKey.Enter)// Select piece

{

notation = game\_counter;

firstplacewriter = false;//writing first place

if (firstplacewriter == false)

{

firstplace\_x = cx;

firstplacey = cy;

}

firstplacewriter = true;

StreamWriter f2 = File.AppendText(gamename + ".txt");

queenbool = true;

queenmovements = true;

tempo1 = "E";

tempo2 = "E";

tempo4 = 0;

tempdenemea = 0;

countera = 0;

bool updown = false;

bool rightleft = false;

bool leftcross = false;

bool rightcross = false;

Promotionbool = false;

loop\_breaker = false;

Console.CursorVisible = false;

while (true)

{

Wrongpiece = false;

if ((board[(cy - 1) / 2, (cx - 1) / 4] == "E"))

{

Console.SetCursorPosition(12, 21);

Console.ForegroundColor = ConsoleColor.DarkYellow;

Console.Write("You do not choose any piece");

Console.ResetColor();

break;

}

else

{

Console.SetCursorPosition(12, 21);

Console.WriteLine(" ");

}

if (game\_counter % 2 == 0 && board[(cy - 1) / 2, (cx - 1) / 4] == "PM")//For Blue Pawn moves

{

while (loop\_breaker == false)

{

Pawn(13, -2);

if (Wrongpiece == true)

break;

}

if (Wrongpiece == true)

break;

if (cy > 1 && cx < 32 && cy < 15 && cx > 4)//ALTTAKİ İFİN KOŞULU

if (board[((cy - 1) / 2) - 1, ((cx - 1) / 4) + 1] == "KK" || board[((cy - 1) / 2) - 1, ((cx - 1) / 4) - 1] == "KK")//ŞAHMATPİYON

{

checkmate = true;

Console.SetCursorPosition(1, 30);

Console.WriteLine("Check");

}

if (cy == 1)

{

while (Promotionbool == false)

Promotion("M");

}

game\_counter++;

break;

}

if (game\_counter % 2 != 0 && board[(cy - 1) / 2, (cx - 1) / 4] == "PM")//For Wrong enter

break;

if (game\_counter % 2 == 1 && board[(cy - 1) / 2, (cx - 1) / 4] == "PK")//For Red Pawn moves

{

while (loop\_breaker == false)

{

Pawn(3, 2);

if (Wrongpiece == true)

break;

}

if (Wrongpiece == true)

break;

if (cy == 15)

{

while (Promotionbool == false)

Promotion("K");

}

game\_counter++;

break;

}//

if (game\_counter % 2 != 1 && board[(cy - 1) / 2, (cx - 1) / 4] == "PK")

break;

if (game\_counter % 2 == 0 && board[(cy - 1) / 2, (cx - 1) / 4] == "RM")//For Blue Rook moves

{

while (queenbool == true)

{

countera++;

if (Wrongpiece == true)

break;

Rook(-1);

}

if (Wrongpiece == true)

break;

if (cy > 1 && cx < 32 && cy < 15 && cx > 4)//ALTTAKİ İFİN KOŞULU

if (board[((cy - 1) / 2) - 1, ((cx - 1) / 4) + 1] == "KM" || board[((cy - 1) / 2) - 1, ((cx - 1) / 4) - 1] == "KM")//ŞAHMATPİYON

{

checkmate = true;

Console.SetCursorPosition(1, 30);

Console.WriteLine("Check");

}

if (temp\_b == -1)

Notation("yeme", "R");

else

Notation("normal", "R");

game\_counter++;

break;

}

if (game\_counter % 2 != 0 && board[(cy - 1) / 2, (cx - 1) / 4] == "RM")

break;

if (game\_counter % 2 == 1 && board[(cy - 1) / 2, (cx - 1) / 4] == "RK")//For Red Rook moves

{

while (queenbool == true)

{

countera++;

if (Wrongpiece == true)

break;

Rook(1);

}

if (Wrongpiece == true)

break;

if (temp\_b == 1)

Notation("yeme", "R");

else

Notation("normal", "R");

game\_counter++;

break;

}

if (game\_counter % 2 != 1 && board[(cy - 1) / 2, (cx - 1) / 4] == "RK")

break;

if (game\_counter % 2 == 0 && board[(cy - 1) / 2, (cx - 1) / 4] == "BM")////For Blue Bishop moves

{

while (queenbool == true)

{

countera++;

if (Wrongpiece == true)

break;

Bishop(-1);

}

if (Wrongpiece == true)

break;

ja = cx;

for (int i = (cy - 3) / 2; i < 1; i -= 2)//ŞAHMAT FİL

{

i = (i - 1) / 2;

if (ja < 32)

{

ja += 4;

if (board2[i, ((ja - 1) / 4)] == 1 && board[i, ((ja - 1) / 4)] != "BM")

break;

else if (board2[i, ((ja - 1) / 4)] == -1 && board[i, ((ja - 1) / 4)] != "KK")

break;

else if (board[i, ((ja - 1) / 4)] == "KK")

{

checkmate = true;

Console.SetCursorPosition(1, 30);

Console.WriteLine("Check");

break;

}

else

checkmate = false;

}

}

if (temp\_b == -1)

Notation("yeme", "B");

else

Notation("normal", "B");

game\_counter++;

break;

}

if (game\_counter % 2 != 0 && board[(cy - 1) / 2, (cx - 1) / 4] == "BM")

break;

if (game\_counter % 2 == 1 && board[(cy - 1) / 2, (cx - 1) / 4] == "BK")//For Red Bishop moves

{

while (queenbool == true)

{

countera++;

if (Wrongpiece == true)

break;

Bishop(1);

}

if (Wrongpiece == true)

break;

if (temp\_b == 1)

Notation("yeme", "B");

else

Notation("normal", "B");

game\_counter++;

break;

}

if (game\_counter % 2 != 1 && board[(cy - 1) / 2, (cx - 1) / 4] == "BK")

break;

if (game\_counter % 2 == 0 && board[(cy - 1) / 2, (cx - 1) / 4] == "QM")//queen

{

Queen(rightleft, updown, -1, rightcross, leftcross);

if (Wrongpiece == true)

break;

game\_counter++;

break;

}

if (game\_counter % 2 != 0 && board[(cy - 1) / 2, (cx - 1) / 4] == "QM")

break;

if (game\_counter % 2 == 1 && board[(cy - 1) / 2, (cx - 1) / 4] == "QK")//queen

{

Queen(rightleft, updown, 1, rightcross, leftcross);

if (Wrongpiece == true)

break;

game\_counter++;

break;

}

if (game\_counter % 2 != 1 && board[(cy - 1) / 2, (cx - 1) / 4] == "QK")

break;

if (game\_counter % 2 == 0 && (board[(cy - 1) / 2, (cx - 1) / 4] == "NM"))

{

while (loop\_breaker == false)

{

if (Wrongpiece == true)

break;

Horse(1);

}

if (Wrongpiece == true)

break;

game\_counter++;

break;

}

if (game\_counter % 2 != 0 && (board[(cy - 1) / 2, (cx - 1) / 4] == "NM"))

break;

if (game\_counter % 2 == 1 && (board[(cy - 1) / 2, (cx - 1) / 4] == "NK"))

{

while (loop\_breaker == false)

{

if (Wrongpiece == true)

break;

Horse(-1);

}

if (Wrongpiece == true)

break;

game\_counter++;

break;

}

if (game\_counter % 2 != 1 && (board[(cy - 1) / 2, (cx - 1) / 4] == "NK"))

break;

if (game\_counter % 2 == 0 && (board[(cy - 1) / 2, (cx - 1) / 4] == "KM"))

{

while (loop\_breaker == false)

{

if (Wrongpiece == true)

break;

King(1, "M");

}

if (Wrongpiece == true)

break;

game\_counter++;

break;

}

if (game\_counter % 2 != 0 && (board[(cy - 1) / 2, (cx - 1) / 4] == "KM"))

break;

if (game\_counter % 2 == 1 && (board[(cy - 1) / 2, (cx - 1) / 4] == "KK"))

{

while (loop\_breaker == false)

{

if (Wrongpiece == true)

break;

King(-1, "K");

}

if (Wrongpiece == true)

break;

game\_counter++;

break;

}

if (game\_counter % 2 == 1 && (board[(cy - 1) / 2, (cx - 1) / 4] == "KM"))

break;

}

if (gamecounter > 1)//For writing file

{

if (moves[gamecounter] == moves[gamecounter - 1])

{

f2.Write(moves[gamecounter]);

}

}

else

f2.Write(moves[gamecounter]);

f2.Close();

sahk = 0;

bool sahka = false;//For game ending

bool ssahma = false;

sahm = 0;

for (int i = 0; i < board.GetLength(0); i++)

{

for (int j = 0; j < board.GetLength(1); j++)

{

if (board[i, j] == "KK")

{

sahk++;

break;

}

}

}

if (sahk == 0)

sahka = true;

for (int i = 0; i < board.GetLength(0); i++)

{

for (int j = 0; j < board.GetLength(1); j++)

{

if (board[i, j] == "KM")

{

sahm++;

break;

}

}

}

if (sahk == 0)

ssahma = true;

if (sahka == true || ssahma == true)

break;

}

if (sahka ==true || ssahma ==true)

break;

}

}

Console.SetCursorPosition(70, 5);

Console.WriteLine("GAME OVER");

Console.ReadLine();

}

}

}