VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

ALGORITHMS

J Component- Review 1

"Chess: Safe Move Predictor"

Slot : L39+L40

SUBMITTED BY:

1. Anushka Dixit [19BCE0577]
2. Sparsh Arya [17BEC0656]
3. Chirag Shenoy [19BCE0356]
4. Ramya K [19BCE0937]
5. AnimishaGadde [19BCE0659]

SUBMITTED TO: Dr.Gayathri P.

ABSTRACT

In this project, we intend to solve a chess problem which predicts the set of moves that will ensure optimum player performance indicating dangerous moves along with safe ones. This application is written in C++ and run via a C++ compiler. All 64 squares of the chess board are marked on a co-ordinate system. A given chess configuration is taken as input from the user. The user is also allowed to select a piece for movement. The range of spaces that a piece can move on the next turn is recorded and pushed onto a stack while ensuring required constraints.

When the user inputs the chess piece to be moved, a background algorithm runs to predict the squares that will ensure the safety of that piece in the nearby turn. The safe moves will be indicated in green squares and the dangerous ones in red. A graphical user interface will be provided to ease out visualization and ensure better prediction.

AIM

To make the prediction of safe and dangerous moves of a chess piece given a certain board configuration using data structures.

OBJECTIVE

Chess is a game that grabs interest of a widespread audience across all age groups, with many new learners every day.

Many beginners, experience difficulties in deciding optimum chess moves to ensure best player performance. Amateurs also face difficulties in remembering piece movement, this code shall be of some help to them.
With this project, we intend to improve player performance while predicting moves that are safe from the upcoming opponent threats.

SCOPEAND APPLICATIONS

Inside Chess

The project can be used to train beginner and novice chess players. This game shall help players prepare for competitions and tournaments even in the absence of an opponent. Predictive strategies used in the algorithm will help players gain confidence in playing chess. This will enhance their visualization capacities along with great improvements in logical reasoning capabilities.

Outside Chess

The model applies in various situations. One such example would be a situation wherein an agent must choose his best move provided the series of alternatives, each of which has a prescribed or hindsight-determined value that is not directly perceived to the person immediately. Some other applications include: -

1. Determine the best path to reach a certain place given the road situation and traffic status.
2. A model that assigns a given UBER driver to a nearby customer given the set of nearby UBER drivers in the locality.
3. An operating system to decide the best task to be executed at a deadlock situation.
4. A smart traffic lighting system that dynamically allows traffic movement based on traffic density.
5. Smart lighting system that ensures low power consumption based on ambient lighting.

INTRODUCTION

In chess, by evaluating the positions on the board before and after a move was made, it is possible to assess whether the move was good or bad. Chess programs make use of a recursive algorithm known as minimax, which performs a kind of tree-based search in order to find the best move. The best move has an assigned score that corresponds to the best position that can arise out of the current one, assuming that both players play best moves throughout.

Chess programs are becoming amazingly strong, to the point that there is little hope that a world-class player can outplay the best computer programs available today. After the much-publicized match between Kasparov and deep blue in 1997, which ended in the machine's favor, chess engines kept evolving to the point that they now seem to belong to an entirely different league.

Disappointing as it may seem, this also creates new opportunities for studying the game and improving the skill of human players. Here, we are interested in taking advantage of the strength of computer programs to suggest them the safe moves possible for each piece in the game.

LITERATURE REVIEW

Chess has had a statistically proven positive impact on the mathematical and cognitive skills of students and hence has always been a point of great interest. Chess engines powerful enough to defeat humans have been developed. Here, the aim has been to create a simple chess engine using data structures which can predict the moves of the chess pieces present on the board.

Chess has been classified as an "imperfect" problem, as the player cannot store all the possible moves of all the pieces on the board in the brain and hence the decisions made are based on information that is not complete. This results in imperfect decisions.

1. **Position Representation**

Position representations, which encapsulate the state of the game at a given time, have undergone few changes in the last 30 years. Back in the 1970s, when memory was expensive, using compact board representations was important. The most popular representation was the most obvious way, a 64-byte array, where each byte represents a single square on the board and contains an integer representing the piece located in that square with some extra bytes for castling opportunities and en passant captures. A clever way to present board was developed by Soviet Unions KAISSA team: they use bitboards, a data structure where each bit represents a game position or state. The state of a chess game can be completely represented by 12 bitboards: each for representing each type of pieces by each player. The main advantage of bitboard representation is that move generation calculations can be done in a more efficient way using these structures. As an example, if you need to generate the moves of the white knights currently on the board, just
find the attack bitboards associated with the positions occupied by the knights and fond their intersection with the logical complement of the bitboard representing "all squares occupied by white pieces" as the only limit on knights is that they can not capture their own pieces. An auxiliary type of representation in chess is transposition tables. The idea is there are often many ways to reach the same position. For example, it doesn't matter whether you play two moves in either order, the game ends up in the same state, this is called transposing. How do transposition tables help chess programming? A position resulting from several positions would not need to be re-evaluated if a transpose of this position was searched and evaluated before and recorded in a transposition table (first incorporated in the Richard Greenblatt's Mac Hack 6 in the late 1960s).

A transposition table is then a repository of past search results, a concept similar to memoization concept we have seen in the class. There are numerous advantages to this process, including speed, and free depth. A related concept is the use of history tables to store killer moves — moves have had interesting results in the past (i.e., which ones have cut-off search quickly along with a continuation) and should, therefore, be tried again at a later time.

2.Move Generation

During a game, a player has 30 or more legal moves to choose from in every situation, some good, some bad. For trained humans, it is easy to characterize the majority of these moves as bad, but coding that information into a computer has proven to be too difficult. Today the strongest programs largely rely on brute force: if you can analyze all possible moves fast enough and predict their consequences far enough down the search tree, it doesn't matter whether or not you start with a clear idea of what you are trying to accomplish, because you'll discover a good move eventually. Move generation can be categorized into the following three categories:

Selective generation: The board is examined and few likely moves are decided while everything else is discarded. This is usually what a human brain does while playing chess.

Incremental generation: A few moves are decided, with the aspiration that they will be very good or very bad, and hence terminate the searches in similar directions.

Complete generation: Generate all moves possible at that time, hoping that the transposition tables will contain relevant information on one of them and that there will be no need to search for anything at all. Selective generation (and its associated search technique, forward pruning) has not been used since the mid-1970s since it was found generating all moves and evaluating all of them were found to be less expensive than a selective generation. An exception is a null move forward pruning.

Sophisticated chess programs since at least CHESS 4.5 have adopted the incremental move generation strategy: generate a few moves at a time, search them, and if a cut-off can be caused, there will be no need to generate the rest of the moves. This approach has some advantages such as low memory use and reducing search space easily because often finding a move that will cause a cutoff is a capture (which are relatively cheap to enumerate).

3. Search Techniques for Chess

Tree search is a central component of any game-playing program. A typical tree search looks at all possible game positions as a tree whose root is the current position; legal game moves determine both the nodes in the tree and the connections between the tree nodes. The problem for any interesting game like chess is that the size of this tree is enormous: if M is the average number of moves per position and D is the depth of the search corresponding to the tree, the tree will contain roughly MD positions. Given that chess games often progress 3 beyond 20 moves and that there can be anywhere from 5-20 different moves at each stage, the size of the tree renders a full search impossible. All practical algorithms merely approximate this full search, with varying degrees of success. We read about the search functions typically described in the literature: Minimax, AlphaBeta, Aspiration Search, and techniques like NegaScout, memory enhanced test, quiescence search, mtd(f). Our final framework provides support for

implementing some but not all of these features; the level of generality required is fairly daunting, especially since many of these algorithms must carry incremental state (e.g. minimax requires access to the structure of the search tree when making its decisions). The further complication of separating the implementation from the algorithm, with respect to the MOVE, POSITION, and PLAYER types our best-move-finding-algorithm interface used, acted as an impediment to fully taking advantage of the full depth of information we could generate and might have occasion to use, depending upon the particular chess algorithm being implemented on the framework. In retrospect, we should have spent more time attempting to generalize from specific algorithms, for our final design fails to adequately address the needs of a sophisticated chess AI.

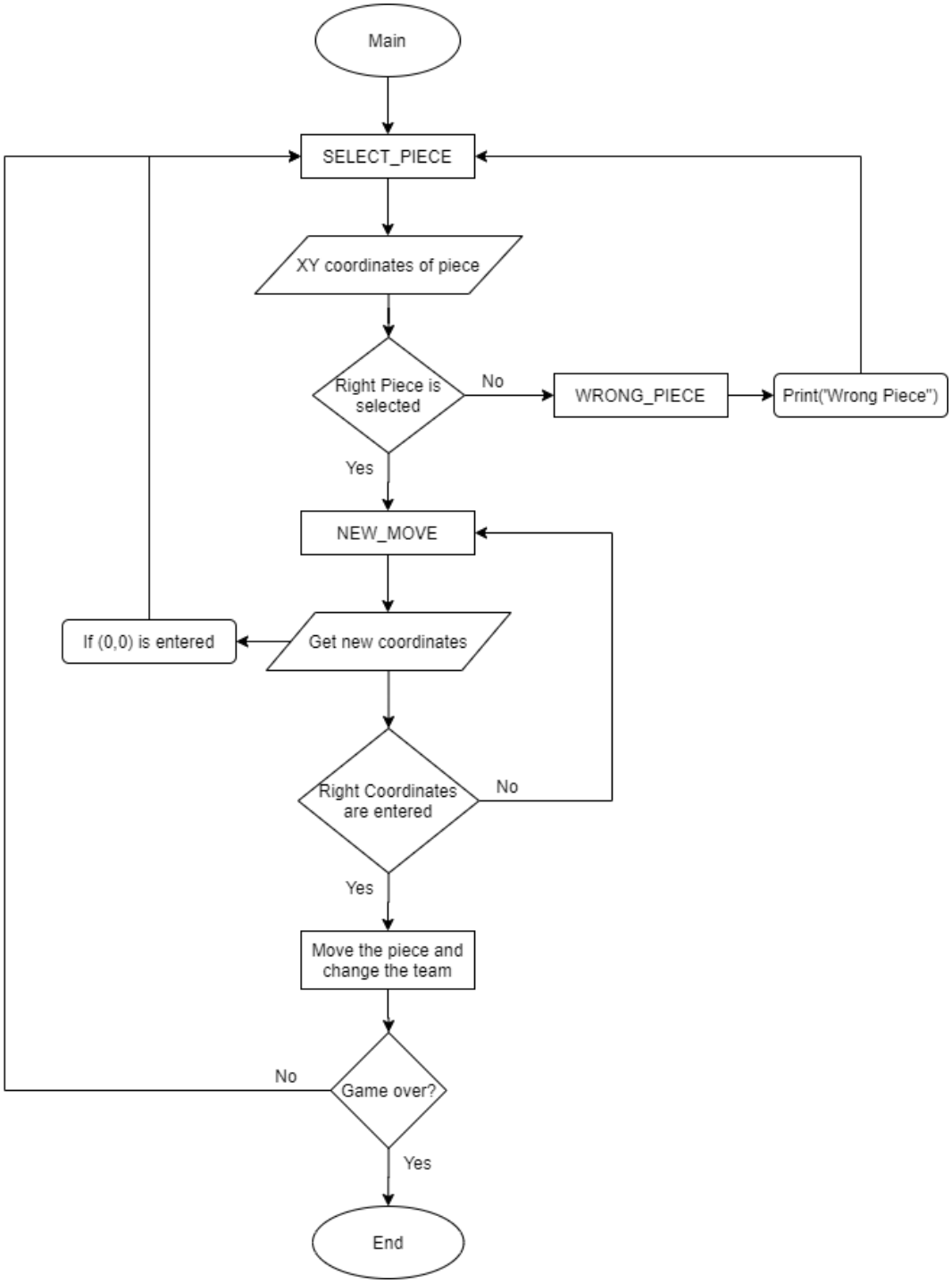
4. Evaluation Functions

Evaluation functions enable the chess engine to evaluate the relative strength of possible moves found through the search in the game tree. There are several features of the game that give advantage to one player over the other. The first and simplest heuristic is the material balance of players. Material balance is an account of which pieces are on the board for each side with each piece assigned a numeric value, with the king having an infinite utility value. Computing material balance is therefore straightforward: a side's material value is equal to the sum of weights of each piece times the number of each piece on aside. The second one is mobility and board control. As an example to understand the importance of mobility, we can take the example of a checkmate where the victim has no legal moves left. Assessing mobility is easy, by just counting the number of legal moves a player at a time can legally make. In practice, this statistic may be useless because many moves are pointless. Trying to limit the opponent's mobility at all costs might cause the program to search for "pointless checks" which results in the destruction of its defence. Still, some strategies such as "bad bishops" are useful. A close concept to mobility is board control. A controlled square is the one where more attacking pieces of the player can access than that of the opponent. A controlled square is considered safe to be moved to, and unsafe to move to one controlled by the opponent. Another important feature is the development. That is the strategy to sequence moves by different pieces. For example, a well-known heuristic is that bishops and knights should be brought into the battle as quickly as possible, the king should castle at the earliest and that rooks and queens should stay quiet until it is time for a crucial attack. Another heuristic that was prevalent in literature is pawn formations. As an example, two or more pawns hinder each other's movement, and hence should not be kept on the same file. King safety is something all sophisticated chess engines try to ensure, Especially, in the opening and middle game, protecting the king is important, one of the best ways to do so is castling.

Lastly, while we have all these different heuristics we need to find a way to give weight to each of these features. There is no absolute answer to find an optimal linear combination of features in evaluation functions. It is tough to refine an evaluation function enough to gain as much performance as one would from an extra ply of search. So it is better to keep the evaluator simple and leave as much processing power as possible to the search techniques.

IMPLEMENTATION

Flowchart:



Program:

```
/*Game Information or rules
Chess Game using OOP in C++
Y-Axis Label Coordinates (14,1)
X-Axis Label Coordinates (18,24)
Dimension of Each Square/Step = (5 characters) x (3 Line Break)
X-Distance to First Component = 17 Characters
Y-Distance to First Component = 1 Line Break
Coordinates of First Component = (18,1)
Horizontal Distance from One Component to Other = 5 Characters
Vertical Distance from One Component to Other = 3 Line Breaks
Pawn (Soldier) = 1
Knight (Horse) = 2
Bishop (Camel) = 3
Rook (Elephant/Castle) = 4
Queen = 5
King = 6
*/
#include<iostream>
#include<windows.h>
#include<conio.h>
using namespace std;
enum pieceName {BLANK = 0, PAWN = 1, KNIGHT = 2, BISHOP = 3, ROOK = 4, QUEEN = 5,
KING = 7};
void gotoxy(int x, int y);
void set_data_highlight (int xx, int yy, int pp, bool uh);
void check_and_unhighlight();
bool end_game_or_not ();
void tell_about_check(int,int);
//Highlighted Steps Data Structure
struct dataHighlight {
        int x_coordinate;
        int y_coordinate;
        int kept_piece;
        bool unhighlight_it;
};
//Game Board Class_____________________________________________________
class CChessBoard {
    public:
         bool change;       //for alternating rows
         //0-Orgument Constructor
```

```
CChessBoard(){
    change=0;
}
//Green Step
void greenStep(){
    HANDLE hConsole;
    hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
    SetConsoleTextAttribute(hConsole, 2);          //Green Colour Code = 2
    for (int i=0; i<5; i++) {
        cout<<"\xB0";
    }
}
//Gray Step
void grayStep(){
    HANDLE hConsole;
    hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
    SetConsoleTextAttribute(hConsole, 8);          //Gray Colour Code = 8
    for (int i=0; i<5; i++) {
        cout<<"\xDB";
    }
}
//Printing Row starting with Green Step
void printRowGreen() {
    for (int i=0; i<4; i++) {
        greenStep();
        grayStep();
    }
}
//Printing Row starting with Gray Step
void printRowGray() {
    for (int i=0; i<4; i++) {
        grayStep();
        greenStep();
    }
}
//Printing Whole Board of the Game
void printBoard () {
    change=0;
    cout<<"\t\t";
    for (int i=0; i<8; i++) {
        if(change==0) {
            printRowGreen(); cout<<"\n"; cout<<"\t\t";
            printRowGreen(); cout<<"\n"; cout<<"\t\t";   //Print 3 rows of green
            printRowGreen(); cout<<"\n"; cout<<"\t\t";
```

```
            change=1;
        }
        else {
            printRowGray(); cout<<"\n"; cout<<"\t\t";
            printRowGray(); cout<<"\n"; cout<<"\t\t";
            printRowGray(); cout<<"\n"; cout<<"\t\t";
            change=0;
        }
    }
}
//Printing X-Axis Label
void PrintX_Label(){
    HANDLE hConsole;
    hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
    SetConsoleTextAttribute(hConsole, 12);          //Red Colour Code = 12
    int cnt = 1;
    for(int i=18;i<58;i+=5){
        gotoxy(i,24);
        cout<<cnt;
        cnt++;
    }
}
//Printing Y-Axis Label
void PrintY_Label(){
    HANDLE hConsole;
    hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
    SetConsoleTextAttribute(hConsole, 12);          //Red Colour Code = 12
    int cnt = 1;
    for(int i=22;i>=1;i-=3){
        gotoxy(15,i);
        cout<<cnt;
        cnt++;
    }
}
//Move To Function
void MoveTo(int numx,int numy){
    HANDLE hConsole;
    hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
    SetConsoleTextAttribute(hConsole, 2);          //Green Colour Code = 2
    int X = (numx * 5)+16-3;
    int Y = 24-(numy * 3)+1;
    gotoxy(X,Y);
}
//Move Function
```

```
        void Move(char chx,char chy){
            HANDLE hConsole;
            hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
            SetConsoleTextAttribute(hConsole, 12);          //Red Colour Code = 12
            int numx = chx - '0'; int numy = chy - '0';
            int X = (numx * 5)+16-3;
            int Y = 24-(numy * 3)+1;
            gotoxy(X,Y);
            cout<<"\xDB";
        }
        //MoveInt Function
        void MoveInt(int numx,int numy, int color){
            HANDLE hConsole;
            hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
            SetConsoleTextAttribute(hConsole, color);          //Green Colour Code = 2
            int X = (numx * 5)+16-3;
            int Y = 24-(numy * 3)+1;
            gotoxy(X,Y);
            cout<<"\xDB";
        }
};
//Piece Class_____________________________________________________________
class CPiece {
    public:
        //Data Members
        bool is_empty;
        bool is_highlight;
        int which_piece;
        bool kill_him;
        int square_x, square_y;
                        //Constructor
        CPiece () {
            square_x = 0;
            square_y = 0;
            is_empty = 1;
            is_highlight = 0;
            which_piece = BLANK;
            kill_him = 0;
        }
        //Set This Square Function
        void setSquare (int sx, int sy, bool isEmp, bool isHigh, int which, bool kill) {
            square_x = sx;
            square_y = sy;
            is_empty = isEmp;
```

```
    is_highlight = isHigh;
    which_piece = which;
    kill_him = kill;
}
//Select Piece Function
bool selectPiece () {
    if (is_empty == 0) {
        return 1;
    }
    else return 0;
}
//Select New Position Function
bool selectNewPosition () {
    if (is_highlight == 1) {
        return 1;
    }
    else return 0;
}
//Print Blank
void printBlank (int cx, int cy) {
    if ( cx%2==0 && cy%2==0 ) {          //If x is even, y is evem
        HANDLE hConsole;                 //Print Gray Step
        hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
        SetConsoleTextAttribute(hConsole, 8);        //Gray Colour Code = 8
        int X = (cx * 5)+16-3;
        int Y = 24-(cy * 3)+1;
        gotoxy(X,Y);
        cout<<"\xDB\xDB";
    }
    else if ( cx%2!=0 && cy%2!=0 ) {     //If x is odd, y is odd
        HANDLE hConsole;                 //Print Gray Step
        hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
        SetConsoleTextAttribute(hConsole, 8);        //Gray Colour Code = 8
        int X = (cx * 5)+16-3;
        int Y = 24-(cy * 3)+1;
        gotoxy(X,Y);
        cout<<"\xDB\xDB";
    }
    else if ( cx%2!=0 && cy%2==0 ) {     //If x is odd, y is even
        HANDLE hConsole;                 //Print Green Step
        hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
        SetConsoleTextAttribute(hConsole, 2);        //Green Colour Code = 2
        int X = (cx * 5)+16-3;
        int Y = 24-(cy * 3)+1;
```

```
            gotoxy(X,Y);
            cout<<"\xB0\xB0";
        }
        else if ( cx%2==0 && cy%2!=0 ) {       //If x is even, y is odd
            HANDLE hConsole;                   //Print Green Step
            hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
            SetConsoleTextAttribute(hConsole, 2);          //Green Colour Code = 2
            int X = (cx * 5)+16-3;
            int Y = 24-(cy * 3)+1;
            gotoxy(X,Y);
            cout<<"\xB0\xB0";
        }
    }
    //Print Piece
    void printPiece (int cx, int cy, int color, int pieceNum) {
        HANDLE hConsole;
        hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
        SetConsoleTextAttribute(hConsole, color);
        int X = (cx * 5)+16-3;
        int Y = 24-(cy * 3)+1;
        gotoxy(X,Y);
        if (pieceNum == PAWN ) {
            cout<<"\xDB";
        }
        else if (pieceNum == ROOK ) {
            cout<<"[]";
        }
        else if (pieceNum == KNIGHT ) {
            cout<<"\x15";
        }
        else if (pieceNum == BISHOP ) {
            cout<<"\x0F";
        }
        else if (pieceNum == QUEEN ) {
            cout<<"\x03";
        }
        else if (pieceNum == KING ) {
            cout<<"\x05";
        }
    }
    //Highlighted Pink Step
                    void pinkHighlightStep(int cx, int cy, int pieceNum);
                    void highlightPiece (int cx, int cy, int thisPiece);
                    void unhighlightStep(int cx, int cy);
```

```
};
//Global Game Board
CPiece gameSquares[9][9];
//Global Highlight Steps Data Container
struct dataHighlight data[9][9];
//Game Play Class_____________________________________________________________
class CGamePlay {
    public:
        //Data Members
        CChessBoard board;
        //Setting Game Board Function
        void settingGameBoard () {
            board.printBoard();
            board.PrintX_Label();
            board.PrintY_Label();
            //Player Pieces_________
            //Pawns for Player
            for (int i=1; i<9; i++) {
               gameSquares[i][2].printPiece(i,2,10,PAWN);
               gameSquares[i][2].setSquare(i,2,0,0,PAWN,0);
            }
            //Rooks for Players
            gameSquares[1][1].printPiece(1,1,10,ROOK);
            gameSquares[1][1].setSquare(1,1,0,0,ROOK,0);
            gameSquares[8][1].printPiece(8,1,10,ROOK);
            gameSquares[8][1].setSquare(8,1,0,0,ROOK,0);
            //Knights for Players
            gameSquares[2][1].printPiece(2,1,10,KNIGHT);
            gameSquares[2][1].setSquare(2,1,0,0,KNIGHT,0);
            gameSquares[7][1].printPiece(7,1,10,KNIGHT);
            gameSquares[7][1].setSquare(7,1,0,0,KNIGHT,0);
            //Bishops for Players
            gameSquares[3][1].printPiece(3,1,10,BISHOP);
            gameSquares[3][1].setSquare(3,1,0,0,BISHOP,0);
            gameSquares[6][1].printPiece(6,1,10,BISHOP);
            gameSquares[6][1].setSquare(6,1,0,0,BISHOP,0);
            //Queen & King for Players
            gameSquares[5][1].printPiece(5,1,10,KING);
            gameSquares[5][1].setSquare(5,1,0,0,KING,0);
            gameSquares[4][1].printPiece(4,1,10,QUEEN);
            gameSquares[4][1].setSquare(4,1,0,0,QUEEN,0);
            //Computer Pieces_________
            //Pawns for Computer
            for (int i=1; i<9; i++) {
```

```
            gameSquares[i][7].printPiece(i,7,15,PAWN);
            gameSquares[i][7].setSquare(i,7,0,0,PAWN,1);
        }
        //Rooks for Computer
        gameSquares[1][8].printPiece(1,8,15,ROOK);
        gameSquares[1][8].setSquare(1,8,0,0,ROOK,1);
        gameSquares[8][8].printPiece(8,8,15,ROOK);
        gameSquares[8][8].setSquare(8,8,0,0,ROOK,1);
        //Knights for Computer
        gameSquares[2][8].printPiece(2,8,15,KNIGHT);
        gameSquares[2][8].setSquare(2,8,0,0,KNIGHT,1);
        gameSquares[7][8].printPiece(7,8,15,KNIGHT);
        gameSquares[7][8].setSquare(7,8,0,0,KNIGHT,1);
        //Bishops for Computer
        gameSquares[3][8].printPiece(3,8,15,BISHOP);
        gameSquares[3][8].setSquare(3,8,0,0,BISHOP,1);
        gameSquares[6][8].printPiece(6,8,15,BISHOP);
        gameSquares[6][8].setSquare(6,8,0,0,BISHOP,1);
        //Queen & King for Computer
        gameSquares[4][8].printPiece(4,8,15,KING);
        gameSquares[4][8].setSquare(4,8,0,0,KING,1);
        gameSquares[5][8].printPiece(5,8,15,QUEEN);
        gameSquares[5][8].setSquare(5,8,0,0,QUEEN,1);
    }
    //Remove Piece Function
    void removePiece (int cx, int cy) {
        gameSquares[cx][cy].setSquare(cx,cy,1,0,0,0);
        gameSquares[cx][cy].printBlank(cx,cy);
    }
};
//Main Function_____________________________________________________
int main () {
    HANDLE hConsole;
    hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
    CGamePlay game1;
    game1.settingGameBoard();
    SetConsoleTextAttribute(hConsole, 7);        //Light White Colour Code = 7
    char chx; char chy;
    int coord_x, coord_y, X, Y;
        int pieceColor;
        bool whose_turn = 0;
        gotoxy(0,10);
        cout<<"Green to Move!";
    while(true){
```

```
SELECT_PIECE:
                gotoxy(0,26);
        SetConsoleTextAttribute(hConsole, 7);          //Light White Colour Code = 7
        cout<<"Which Piece (in form of XY, for e.g. 62): ";
        gotoxy(42,26);
        chx = getche(); chy = getche();
                //cin>>chx; cin>>chy;
        coord_x = chx - '0'; coord_y = chy - '0';
                if (gameSquares[coord_x][coord_y].kill_him != whose_turn ) { goto
WRONG_PIECE; }
                else {
                        if ( gameSquares[coord_x][coord_y].selectPiece() ) {

gameSquares[coord_x][coord_y].highlightPiece(coord_x,coord_y,gameSquares[coord_x][coord
_y].which_piece);
NEW_MOVE:
                                if (gameSquares[coord_x][coord_y].kill_him == 1) {
pieceColor = 15; }
                                else { pieceColor = 10; }
                                gotoxy(0,25);
                        SetConsoleTextAttribute(hConsole, 14);          //Yellow Colour
Code = 14
                        cout<<"Where to Move (New Coordinates):  ";
                        gotoxy(33,25);
                        //cin>>chx; cin>>chy;
                                chx = getche(); chy = getche();
                        X = chx - '0';
                        Y = chy - '0';
                                if (X==0 && Y==0) {
                                        check_and_unhighlight();

        gameSquares[coord_x][coord_y].printPiece(coord_x,coord_y,pieceColor,gameSquares[c
oord_x][coord_y].which_piece);
                                        whose_turn =
gameSquares[coord_x][coord_y].kill_him;
                                        goto SELECT_PIECE;
                                }
                        if (gameSquares[X][Y].is_highlight==1) {
                                        check_and_unhighlight();
                                        game1.removePiece(X,Y);
                                        if (gameSquares[coord_x][coord_y].which_piece ==
PAWN && gameSquares[coord_x][coord_y].kill_him == 0 && Y==8) {
                                                if (X==1 || X==8 ) {
gameSquares[coord_x][coord_y].which_piece = ROOK; }
```

```
                                                else if (X==2 || X==7) {
gameSquares[coord_x][coord_y].which_piece = KNIGHT; }
                                                else if (X==3 || X==6) {
gameSquares[coord_x][coord_y].which_piece = BISHOP; }
                                                else if (X==4 || X==5) {
gameSquares[coord_x][coord_y].which_piece = QUEEN; }
                                        }
                                        if (gameSquares[coord_x][coord_y].which_piece ==
PAWN && gameSquares[coord_x][coord_y].kill_him == 1 && Y==1) {
                                                if (X==1 || X==8) {
gameSquares[coord_x][coord_y].which_piece = ROOK; }
                                                else if (X==2 || X==7) {
gameSquares[coord_x][coord_y].which_piece = KNIGHT; }
                                                else if (X==3 || X==6) {
gameSquares[coord_x][coord_y].which_piece = BISHOP; }
                                                else if (X==4 || X==5) {
gameSquares[coord_x][coord_y].which_piece = QUEEN; }
                                        }

        gameSquares[coord_x][coord_y].printPiece(X,Y,pieceColor,gameSquares[coord_x][coor
d_y].which_piece);

        gameSquares[X][Y].setSquare(X,Y,0,0,gameSquares[coord_x][coord_y].which_piece,ga
meSquares[coord_x][coord_y].kill_him);
                                        game1.removePiece(coord_x,coord_y);
                                        data[X][Y].unhighlight_it = 0;
                                        gotoxy(5,15);
                                        SetConsoleTextAttribute(hConsole, 0);          //RED
Colour Code = 207
                                        cout<<"      ";
                                        tell_about_check(X,Y);
                                }
                                else {
                                        gotoxy(0,27);
                                        SetConsoleTextAttribute(hConsole, 12);          //Red
Colour Code = 12
                                        cout<<"Invalid Move!";
                                        goto NEW_MOVE;
                                }
                                if (whose_turn == 0) {
                                        whose_turn = 1;
                                        gotoxy(0,10);
                                        cout<<"Green to Move!";
                                }
```

```
                                        else {
                                                whose_turn = 0;
                                                gotoxy(0,10);
                                                cout<<"White to Move!";
                                        }
                                        gotoxy(0,27);
                                        SetConsoleTextAttribute(hConsole, 0);         //Red Colour
Code = 12
                                        cout<<"                    ";
                                }
                                else {
WRONG_PIECE:
                                        gotoxy(0,27);
                                        SetConsoleTextAttribute(hConsole, 12);         //Red Colour
Code = 12
                                        cout<<"Wrong Piece Selected!";
                                }
                        }
                        if ( end_game_or_not() ) { break; }
                        if (whose_turn == 0) {
                                SetConsoleTextAttribute(hConsole, 10);         //Green Colour
Code = 10
                                gotoxy(0,10);
                                cout<<"Green to Move!";
                        }
                        else {
                                SetConsoleTextAttribute(hConsole, 15);         //White Colour
Code = 15
                                gotoxy(0,10);
                                cout<<"White to Move!";
                        }
    }
    gotoxy(0,25);
          cout<<"Game Over!";
    system("Pause");
}
//gotoxy Function_____________________________________________________
void gotoxy(int x, int y)
{
     HANDLE hConsoleOutput;
     COORD dwCursorPosition;
     cout.flush();
     dwCursorPosition.X = x;
     dwCursorPosition.Y = y;
```

```
    hConsoleOutput = GetStdHandle(STD_OUTPUT_HANDLE);
    SetConsoleCursorPosition(hConsoleOutput,dwCursorPosition);
}
//Set Data Function for dataHighlight stru//////cture
void set_data_highlight (int xx, int yy, int pp, bool uh) {
        data[xx][yy].x_coordinate = xx;
        data[xx][yy].y_coordinate = yy;
        data[xx][yy].kept_piece = pp;
        data[xx][yy].unhighlight_it = uh;
}
//Check And Unhighlight the Highlighted Steps
void check_and_unhighlight() {
        for (int i=1; i<9; i++) {
                for (int j=1; j<9; j++) {
                        if (data[i][j].unhighlight_it == 1) {
                                gameSquares[i][j].unhighlightStep(i,j);
                                data[i][j].unhighlight_it = 0;
                        }
                }
        }
}
//To Check whether Game is ended or not
bool end_game_or_not () {
        int num=0;
        for (int i=1; i<9; i++) {
                for (int j=1; j<9; j++) {
                        if (gameSquares[i][j].which_piece == KING) {
                                num++;
                        }
                }
        }
        if (num == 2) {
                return 0;
        }
        else return 1;
}
//Tell About the Check
/*void tell_about_check (int xx, int yy) {
        if ( gameSquares[xx][yy].which_piece == KING && gameSquares[xx][yy].kill_him == 1
&& gameSquares[xx][yy].is_highlight == 1) {
                gotoxy(5,15);
                HANDLE hConsole;
                hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
                SetConsoleTextAttribute(hConsole, 207);         //RED Colour Code = 207
```

```
                cout<<"Check";
        }
}*/
//Highlight Piece Function_____________________________________________
void CPiece::highlightPiece (int cx, int cy, int thisPiece) {
        //_____________________PAWN STEPS_____________________//
        if (thisPiece == PAWN) {
                if (gameSquares[cx][cy].kill_him == 0) {        //If it is Player's PAWN
                        if (cy == 2) {                  //If this is first action of pawn
                                printPiece(cx,cy,14,PAWN);
                           for (int i=cy, j=0; j<2; i++, j++) {    //j<2 for 2 steps
                                        if (gameSquares[cx][i+1].is_empty==0 &&
gameSquares[cx][i+1].kill_him==0) { break; }
                                        else if (gameSquares[cx][i+1].is_empty==0 &&
gameSquares[cx][i+1].kill_him==1) { break; }
                                        else {
                                                pinkHighlightStep(cx,i,PAWN);

        gameSquares[cx][i+1].setSquare(cx,i,1,1,BLANK,0);
                                                set_data_highlight(cx,i+1,BLANK,1);
                                        }
                                }
                                if (gameSquares[cx+1][cy+1].is_empty==0 &&
gameSquares[cx+1][cy+1].kill_him==1) {

        pinkHighlightStep(cx+1,cy,gameSquares[cx+1][cy+1].which_piece);

        gameSquares[cx+1][cy+1].setSquare(cx+1,cy+1,0,1,gameSquares[cx+1][cy+1].which_pi
ece,1);

        set_data_highlight(cx+1,cy+1,gameSquares[cx+1][cy+1].which_piece,1);

        gameSquares[cx+1][cy+1].printPiece(cx+1,cy+1,11,gameSquares[cx+1][cy+1].which_pie
ce);
                                        //tell_about_check(cx+1,cy+1);
                                }
                                if (gameSquares[cx-1][cy+1].is_empty==0 && gameSquares[cx-
1][cy+1].kill_him==1) {
                                        pinkHighlightStep(cx-1,cy,gameSquares[cx-
1][cy+1].which_piece);
                                        gameSquares[cx-1][cy+1].setSquare(cx-
1,cy+1,0,1,gameSquares[cx-1][cy+1].which_piece,1);
                                        set_data_highlight(cx-1,cy+1,gameSquares[cx-
1][cy+1].which_piece,1);
```

```
                                        gameSquares[cx-1][cy+1].printPiece(cx-
1,cy+1,11,gameSquares[cx-1][cy+1].which_piece);
                                        //tell_about_check(cx+1,cy+1);
                                }
                        }
                        else {                          //Not First Move
                                printPiece(cx,cy,14,PAWN);
                                if (gameSquares[cx][cy+1].is_empty==0 &&
gameSquares[cx][cy+1].kill_him==0) { }
                                else if (gameSquares[cx][cy+1].is_empty==0 &&
gameSquares[cx][cy+1].kill_him==1) { }
                                else {
                                        pinkHighlightStep(cx,cy,BLANK);
                                        gameSquares[cx][cy+1].setSquare(cx,cy+1,1,1,BLANK,0);
                                        set_data_highlight(cx,cy+1,BLANK,1);
                                }
                                if (gameSquares[cx+1][cy+1].is_empty==0 &&
gameSquares[cx+1][cy+1].kill_him==1) {

        pinkHighlightStep(cx+1,cy,gameSquares[cx+1][cy+1].which_piece);

        gameSquares[cx+1][cy+1].setSquare(cx+1,cy+1,0,1,gameSquares[cx+1][cy+1].which_pi
ece,1);

        set_data_highlight(cx+1,cy+1,gameSquares[cx+1][cy+1].which_piece,1);

        gameSquares[cx+1][cy+1].printPiece(cx+1,cy+1,11,gameSquares[cx+1][cy+1].which_pie
ce);
                                }
                                if (gameSquares[cx-1][cy+1].is_empty==0 && gameSquares[cx-
1][cy+1].kill_him==1) {
                                        pinkHighlightStep(cx-1,cy,gameSquares[cx-
1][cy+1].which_piece);
                                        gameSquares[cx-1][cy+1].setSquare(cx-
1,cy+1,0,1,gameSquares[cx-1][cy+1].which_piece,1);
                                        set_data_highlight(cx-1,cy+1,gameSquares[cx-
1][cy+1].which_piece,1);
                                        gameSquares[cx-1][cy+1].printPiece(cx-
1,cy+1,11,gameSquares[cx-1][cy+1].which_piece);
                                }
                        }
                }

                else {   //If it is Computer's PAWN
```

```
                        if (cy == 7) {                  //If this is first action of pawn
                                printPiece(cx,cy,14,PAWN);
                           for (int i=cy-1, j=0; j<2; i--, j++) {    //j<2 for 2 steps
                                if (gameSquares[cx][i].is_empty==0 &&
gameSquares[cx][i].kill_him==0) { break; }
                                else if (gameSquares[cx][i].is_empty==0 &&
gameSquares[cx][i].kill_him==1) { break; }
                                else {
                                        pinkHighlightStep(cx,i-1,PAWN);
                                        gameSquares[cx][i].setSquare(cx,i,1,1,BLANK,0);
                                        set_data_highlight(cx,i,BLANK,1);
                                }
                           }
                           if (gameSquares[cx+1][cy-1].is_empty==0 &&
gameSquares[cx+1][cy-1].kill_him==0 && cx!=8) {
                                pinkHighlightStep(cx+1,cy-2,gameSquares[cx+1][cy-
1].which_piece);
                                gameSquares[cx+1][cy-1].setSquare(cx+1,cy-
1,0,1,gameSquares[cx+1][cy-1].which_piece,0);
                                set_data_highlight(cx+1,cy-1,gameSquares[cx+1][cy-
1].which_piece,1);
                                gameSquares[cx+1][cy-1].printPiece(cx+1,cy-
1,11,gameSquares[cx+1][cy-1].which_piece);
                           }
                           if (gameSquares[cx-1][cy-1].is_empty==0 && gameSquares[cx-
1][cy-1].kill_him==0 && cx!=1) {
                                pinkHighlightStep(cx-1,cy-2,gameSquares[cx-1][cy-
1].which_piece);
                                gameSquares[cx-1][cy-1].setSquare(cx-1,cy-
1,0,1,gameSquares[cx-1][cy-1].which_piece,0);
                                set_data_highlight(cx-1,cy-1,gameSquares[cx-1][cy-
1].which_piece,1);
                                gameSquares[cx-1][cy-1].printPiece(cx-1,cy-
1,11,gameSquares[cx-1][cy-1].which_piece);
                           }
                        }
                        else {
                                printPiece(cx,cy,14,PAWN);
                                if (gameSquares[cx][cy-1].is_empty==0 && gameSquares[cx][cy-
1].kill_him==1) { }
                                if (gameSquares[cx][cy-1].is_empty==0 && gameSquares[cx][cy-
1].kill_him==0) { }
                                else {
                                        pinkHighlightStep(cx,cy-2,PAWN);
```

```
                                        gameSquares[cx][cy-1].setSquare(cx,cy-1,1,1,BLANK,0);
                                        set_data_highlight(cx,cy-1,BLANK,1);
                                }
                                if (gameSquares[cx+1][cy-1].is_empty==0 &&
gameSquares[cx+1][cy-1].kill_him==0 && cx!=8) {
                                        pinkHighlightStep(cx+1,cy-2,gameSquares[cx+1][cy-
1].which_piece);
                                        gameSquares[cx+1][cy-1].setSquare(cx+1,cy-
1,0,1,gameSquares[cx+1][cy-1].which_piece,gameSquares[cx+1][cy-1].kill_him);
                                        set_data_highlight(cx+1,cy-1,gameSquares[cx+1][cy-
1].which_piece,1);
                                        gameSquares[cx+1][cy-1].printPiece(cx+1,cy-
1,11,gameSquares[cx+1][cy-1].which_piece);
                                }
                                if (gameSquares[cx-1][cy-1].is_empty==0 && gameSquares[cx-
1][cy-1].kill_him==0 && cx!=1) {
                                        pinkHighlightStep(cx-1,cy-2,gameSquares[cx-1][cy-
1].which_piece);
                                        gameSquares[cx-1][cy-1].setSquare(cx-1,cy-
1,0,1,gameSquares[cx-1][cy-1].which_piece,gameSquares[cx+1][cy-1].kill_him);
                                        set_data_highlight(cx-1,cy-1,gameSquares[cx-1][cy-
1].which_piece,1);
                                        gameSquares[cx-1][cy-1].printPiece(cx-1,cy-
1,11,gameSquares[cx-1][cy-1].which_piece);
                                }
                        }
                }
        }
        //_____________________KNIGHT STEPS_____________________//
   else if (thisPiece == KNIGHT) {
                if (gameSquares[cx][cy].kill_him == 0) {
                   printPiece(cx,cy,14,KNIGHT);
                        int temp_x; int temp_y;
                        //First Step (X-1,Y+2)
                        temp_x = cx - 1;
                        temp_y = cy + 1;
                        if (temp_x < 1 || temp_x > 8 || temp_y < 0 || temp_y >= 8) { }
                        else {
                                if (gameSquares[temp_x][temp_y+1].is_empty==0 &&
gameSquares[temp_x][temp_y+1].kill_him==0) { }
                                else if (gameSquares[temp_x][temp_y+1].is_empty==0 &&
gameSquares[temp_x][temp_y+1].kill_him==1) {

        pinkHighlightStep(temp_x,temp_y,gameSquares[temp_x][temp_y+1].which_piece);
```

```
	gameSquares[temp_x][temp_y+1].setSquare(temp_x,temp_y+1,0,1,gameSquares[temp
_x][temp_y+1].which_piece,1);

	set_data_highlight(temp_x,temp_y+1,gameSquares[temp_x][temp_y+1].which_piece,1);

	gameSquares[temp_x][temp_y+1].printPiece(temp_x,temp_y+1,11,gameSquares[temp_
x][temp_y+1].which_piece);
				}
				else  {
					pinkHighlightStep(temp_x,temp_y,BLANK);

	gameSquares[temp_x][temp_y+1].setSquare(temp_x,temp_y+1,1,1,BLANK,0);
					set_data_highlight(temp_x,temp_y+1,BLANK,1);
				}
			}
			//Second Step (X-1,Y-2)
			temp_x = cx - 1;
			temp_y = cy - 3;
			if (temp_x < 1 || temp_x > 8 || temp_y < 0 || temp_y >= 8) { }
			else {
				if (gameSquares[temp_x][temp_y+1].is_empty==0 &&
gameSquares[temp_x][temp_y+1].kill_him==0 ) { }
				else if (gameSquares[temp_x][temp_y+1].is_empty==0 &&
gameSquares[temp_x][temp_y+1].kill_him==1) {

	pinkHighlightStep(temp_x,temp_y,gameSquares[temp_x][temp_y+1].which_piece);

	gameSquares[temp_x][temp_y+1].setSquare(temp_x,temp_y+1,0,1,gameSquares[temp
_x][temp_y+1].which_piece,1);

	set_data_highlight(temp_x,temp_y+1,gameSquares[temp_x][temp_y+1].which_piece,1);

	gameSquares[temp_x][temp_y+1].printPiece(temp_x,temp_y+1,11,gameSquares[temp_
x][temp_y+1].which_piece);
				}
				else  {
					pinkHighlightStep(temp_x,temp_y,BLANK);

	gameSquares[temp_x][temp_y+1].setSquare(temp_x,temp_y+1,1,1,BLANK,0);
					set_data_highlight(temp_x,temp_y+1,BLANK,1);
				}
			}
			//Third Step (X+1,Y+2)
```

```
                        temp_x = cx + 1;
                        temp_y = cy + 1;
                        if (temp_x < 1 || temp_x > 8 || temp_y < 0 || temp_y >= 8) { }
                        else {
                                if (gameSquares[temp_x][temp_y+1].is_empty==0 &&
gameSquares[temp_x][temp_y+1].kill_him==0) { }
                                else if (gameSquares[temp_x][temp_y+1].is_empty==0 &&
gameSquares[temp_x][temp_y+1].kill_him==1) {

        pinkHighlightStep(temp_x,temp_y,gameSquares[temp_x][temp_y+1].which_piece);

        gameSquares[temp_x][temp_y+1].setSquare(temp_x,temp_y+1,0,1,gameSquares[temp
_x][temp_y+1].which_piece,1);

        set_data_highlight(temp_x,temp_y+1,gameSquares[temp_x][temp_y+1].which_piece,1);

        gameSquares[temp_x][temp_y+1].printPiece(temp_x,temp_y+1,11,gameSquares[temp_
x][temp_y+1].which_piece);
                                }
                                else  {
                                        pinkHighlightStep(temp_x,temp_y,BLANK);

        gameSquares[temp_x][temp_y+1].setSquare(temp_x,temp_y+1,1,1,BLANK,0);
                                        set_data_highlight(temp_x,temp_y+1,BLANK,1);
                                }
                        }
                        //Fourth Step (X+1,Y-2)
                        temp_x = cx + 1;
                        temp_y = cy - 3;
                        if (temp_x < 1 || temp_x > 8 || temp_y < 0 || temp_y >= 8) { }
                        else {
                                if (gameSquares[temp_x][temp_y+1].is_empty==0 &&
gameSquares[temp_x][temp_y+1].kill_him==0) { }
                                else if (gameSquares[temp_x][temp_y+1].is_empty==0 &&
gameSquares[temp_x][temp_y+1].kill_him==1) {

        pinkHighlightStep(temp_x,temp_y,gameSquares[temp_x][temp_y+1].which_piece);

        gameSquares[temp_x][temp_y+1].setSquare(temp_x,temp_y+1,0,1,gameSquares[temp
_x][temp_y+1].which_piece,1);

        set_data_highlight(temp_x,temp_y+1,gameSquares[temp_x][temp_y+1].which_piece,1);
```

```
        gameSquares[temp_x][temp_y+1].printPiece(temp_x,temp_y+1,11,gameSquares[temp_
x][temp_y+1].which_piece);
                                }
                                else  {
                                        pinkHighlightStep(temp_x,temp_y,BLANK);

        gameSquares[temp_x][temp_y+1].setSquare(temp_x,temp_y+1,1,1,BLANK,0);
                                        set_data_highlight(temp_x,temp_y+1,BLANK,1);
                                }
                        }
                        //Fifth Step (X-2,Y-1)
                        temp_x = cx - 2;
                        temp_y = cy - 2;
                        if (temp_x < 1 || temp_x > 8 || temp_y < 0 || temp_y >= 8) { }
                        else {
                                if (gameSquares[temp_x][temp_y+1].is_empty==0 &&
gameSquares[temp_x][temp_y+1].kill_him==0) { }
                                else if (gameSquares[temp_x][temp_y+1].is_empty==0 &&
gameSquares[temp_x][temp_y+1].kill_him==1) {

        pinkHighlightStep(temp_x,temp_y,gameSquares[temp_x][temp_y+1].which_piece);

        gameSquares[temp_x][temp_y+1].setSquare(temp_x,temp_y+1,0,1,gameSquares[temp
_x][temp_y+1].which_piece,1);

        set_data_highlight(temp_x,temp_y+1,gameSquares[temp_x][temp_y+1].which_piece,1);

        gameSquares[temp_x][temp_y+1].printPiece(temp_x,temp_y+1,11,gameSquares[temp_
x][temp_y+1].which_piece);
                                }
                                else  {
                                        pinkHighlightStep(temp_x,temp_y,BLANK);

        gameSquares[temp_x][temp_y+1].setSquare(temp_x,temp_y+1,1,1,BLANK,0);
                                        set_data_highlight(temp_x,temp_y+1,BLANK,1);
                                }
                        }
                        //Sixth Step (X-2,Y+1)
                        temp_x = cx - 2;
                        temp_y = cy + 0;
                        if (temp_x < 1 || temp_x > 8 || temp_y < 0 || temp_y >= 8) { }
                        else {
```

```
                        if (gameSquares[temp_x][temp_y+1].is_empty==0 &&
gameSquares[temp_x][temp_y+1].kill_him==0) { }
                        else if (gameSquares[temp_x][temp_y+1].is_empty==0 &&
gameSquares[temp_x][temp_y+1].kill_him==1) {

        pinkHighlightStep(temp_x,temp_y,gameSquares[temp_x][temp_y+1].which_piece);

        gameSquares[temp_x][temp_y+1].setSquare(temp_x,temp_y+1,0,1,gameSquares[temp
_x][temp_y+1].which_piece,1);

        set_data_highlight(temp_x,temp_y+1,gameSquares[temp_x][temp_y+1].which_piece,1);

        gameSquares[temp_x][temp_y+1].printPiece(temp_x,temp_y+1,11,gameSquares[temp_
x][temp_y+1].which_piece);
                        }
                        else  {
                                pinkHighlightStep(temp_x,temp_y,BLANK);

        gameSquares[temp_x][temp_y+1].setSquare(temp_x,temp_y+1,1,1,BLANK,0);
                                set_data_highlight(temp_x,temp_y+1,BLANK,1);
                        }
                }
                //Seventh Step (X+2,Y-1)
                temp_x = cx + 2;
                temp_y = cy - 2;
                if (temp_x < 1 || temp_x > 8 || temp_y < 0 || temp_y >= 8) { }
                else {
                        if (gameSquares[temp_x][temp_y+1].is_empty==0 &&
gameSquares[temp_x][temp_y+1].kill_him==0) { }
                        else if (gameSquares[temp_x][temp_y+1].is_empty==0 &&
gameSquares[temp_x][temp_y+1].kill_him==1) {

        pinkHighlightStep(temp_x,temp_y,gameSquares[temp_x][temp_y+1].which_piece);

        gameSquares[temp_x][temp_y+1].setSquare(temp_x,temp_y+1,0,1,gameSquares[temp
_x][temp_y+1].which_piece,1);

        set_data_highlight(temp_x,temp_y+1,gameSquares[temp_x][temp_y+1].which_piece,1);

        gameSquares[temp_x][temp_y+1].printPiece(temp_x,temp_y+1,11,gameSquares[temp_
x][temp_y+1].which_piece);
                        }
                        else  {
                                pinkHighlightStep(temp_x,temp_y,BLANK);
```

```
        gameSquares[temp_x][temp_y+1].setSquare(temp_x,temp_y+1,1,1,BLANK,0);
                                        set_data_highlight(temp_x,temp_y+1,BLANK,1);
                                }
                        }
                        //Eighth Step (X+2,Y+1)
                        temp_x = cx + 2;
                        temp_y = cy + 0;
                        if (temp_x < 1 || temp_x > 8 || temp_y < 0 || temp_y >= 8) { }
                        else {
                                if (gameSquares[temp_x][temp_y+1].is_empty==0 &&
gameSquares[temp_x][temp_y+1].kill_him==0) { }
                                else if (gameSquares[temp_x][temp_y+1].is_empty==0 &&
gameSquares[temp_x][temp_y+1].kill_him==1) {

        pinkHighlightStep(temp_x,temp_y,gameSquares[temp_x][temp_y+1].which_piece);

        gameSquares[temp_x][temp_y+1].setSquare(temp_x,temp_y+1,0,1,gameSquares[temp
_x][temp_y+1].which_piece,1);

        set_data_highlight(temp_x,temp_y+1,gameSquares[temp_x][temp_y+1].which_piece,1);

        gameSquares[temp_x][temp_y+1].printPiece(temp_x,temp_y+1,11,gameSquares[temp_
x][temp_y+1].which_piece);
                                }
                                else  {
                                        pinkHighlightStep(temp_x,temp_y,BLANK);

        gameSquares[temp_x][temp_y+1].setSquare(temp_x,temp_y+1,1,1,BLANK,0);
                                        set_data_highlight(temp_x,temp_y+1,BLANK,1);
                                }
                        }
                }
                else  {
                   printPiece(cx,cy,14,KNIGHT);
                        int temp_x; int temp_y;
                        //First Step (X-1,Y+2)
                        temp_x = cx - 1;
                        temp_y = cy + 1;
                        if (temp_x < 1 || temp_x > 8 || temp_y < 0 || temp_y >= 8) { }
                        else {
                                if (gameSquares[temp_x][temp_y+1].is_empty==0 &&
gameSquares[temp_x][temp_y+1].kill_him==1) { }
```

```
                        else if (gameSquares[temp_x][temp_y+1].is_empty==0 &&
gameSquares[temp_x][temp_y+1].kill_him==0) {

        pinkHighlightStep(temp_x,temp_y,gameSquares[temp_x][temp_y+1].which_piece);

        gameSquares[temp_x][temp_y+1].setSquare(temp_x,temp_y+1,0,1,gameSquares[temp
_x][temp_y+1].which_piece,gameSquares[temp_x][temp_y+1].kill_him);

        set_data_highlight(temp_x,temp_y+1,gameSquares[temp_x][temp_y+1].which_piece,1);

        gameSquares[temp_x][temp_y+1].printPiece(temp_x,temp_y+1,11,gameSquares[temp_
x][temp_y+1].which_piece);
                        }
                        else  {
                                pinkHighlightStep(temp_x,temp_y,BLANK);

        gameSquares[temp_x][temp_y+1].setSquare(temp_x,temp_y+1,1,1,BLANK,0);
                                set_data_highlight(temp_x,temp_y+1,BLANK,1);
                        }
                }
                //Second Step (X-1,Y-2)
                temp_x = cx - 1;
                temp_y = cy - 3;
                if (temp_x < 1 || temp_x > 8 || temp_y < 0 || temp_y >= 8) { }
                else {
                        if (gameSquares[temp_x][temp_y+1].is_empty==0 &&
gameSquares[temp_x][temp_y+1].kill_him==1 ) { }
                        else if (gameSquares[temp_x][temp_y+1].is_empty==0 &&
gameSquares[temp_x][temp_y+1].kill_him==0) {

        pinkHighlightStep(temp_x,temp_y,gameSquares[temp_x][temp_y+1].which_piece);

        gameSquares[temp_x][temp_y+1].setSquare(temp_x,temp_y+1,0,1,gameSquares[temp
_x][temp_y+1].which_piece,gameSquares[temp_x][temp_y+1].kill_him);

        set_data_highlight(temp_x,temp_y+1,gameSquares[temp_x][temp_y+1].which_piece,1);

        gameSquares[temp_x][temp_y+1].printPiece(temp_x,temp_y+1,11,gameSquares[temp_
x][temp_y+1].which_piece);
                        }
                        else  {
                                pinkHighlightStep(temp_x,temp_y,BLANK);

        gameSquares[temp_x][temp_y+1].setSquare(temp_x,temp_y+1,1,1,BLANK,0);
```

```
                                        set_data_highlight(temp_x,temp_y+1,BLANK,1);
                                }
                        }
                        //Third Step (X+1,Y+2)
                        temp_x = cx + 1;
                        temp_y = cy + 1;
                        if (temp_x < 1 || temp_x > 8 || temp_y < 0 || temp_y >= 8) { }
                        else {
                                if (gameSquares[temp_x][temp_y+1].is_empty==0 &&
gameSquares[temp_x][temp_y+1].kill_him==1) { }
                                else if (gameSquares[temp_x][temp_y+1].is_empty==0 &&
gameSquares[temp_x][temp_y+1].kill_him==0) {

        pinkHighlightStep(temp_x,temp_y,gameSquares[temp_x][temp_y+1].which_piece);

        gameSquares[temp_x][temp_y+1].setSquare(temp_x,temp_y+1,0,1,gameSquares[temp
_x][temp_y+1].which_piece,gameSquares[temp_x][temp_y+1].kill_him);

        set_data_highlight(temp_x,temp_y+1,gameSquares[temp_x][temp_y+1].which_piece,1);

        gameSquares[temp_x][temp_y+1].printPiece(temp_x,temp_y+1,11,gameSquares[temp_
x][temp_y+1].which_piece);
                                }
                                else  {
                                        pinkHighlightStep(temp_x,temp_y,BLANK);

        gameSquares[temp_x][temp_y+1].setSquare(temp_x,temp_y+1,1,1,BLANK,0);
                                        set_data_highlight(temp_x,temp_y+1,BLANK,1);
                                }
                        }
                        //Fourth Step (X+1,Y-2)
                        temp_x = cx + 1;
                        temp_y = cy - 3;
                        if (temp_x < 1 || temp_x > 8 || temp_y < 0 || temp_y >= 8) { }
                        else {
                                if (gameSquares[temp_x][temp_y+1].is_empty==0 &&
gameSquares[temp_x][temp_y+1].kill_him==1) { }
                                else if (gameSquares[temp_x][temp_y+1].is_empty==0 &&
gameSquares[temp_x][temp_y+1].kill_him==0) {

        pinkHighlightStep(temp_x,temp_y,gameSquares[temp_x][temp_y+1].which_piece);

        gameSquares[temp_x][temp_y+1].setSquare(temp_x,temp_y+1,0,1,gameSquares[temp
_x][temp_y+1].which_piece,gameSquares[temp_x][temp_y+1].kill_him);
```

```
	set_data_highlight(temp_x,temp_y+1,gameSquares[temp_x][temp_y+1].which_piece,1);

	gameSquares[temp_x][temp_y+1].printPiece(temp_x,temp_y+1,11,gameSquares[temp_
x][temp_y+1].which_piece);
				}
				else  {
					pinkHighlightStep(temp_x,temp_y,BLANK);

	gameSquares[temp_x][temp_y+1].setSquare(temp_x,temp_y+1,1,1,BLANK,0);
					set_data_highlight(temp_x,temp_y+1,BLANK,1);
				}
			}
			//Fifth Step (X-2,Y-1)
			temp_x = cx - 2;
			temp_y = cy - 2;
			if (temp_x < 1 || temp_x > 8 || temp_y < 0 || temp_y >= 8) { }
			else {
				if (gameSquares[temp_x][temp_y+1].is_empty==0 &&
gameSquares[temp_x][temp_y+1].kill_him==1) { }
				else if (gameSquares[temp_x][temp_y+1].is_empty==0 &&
gameSquares[temp_x][temp_y+1].kill_him==0) {

	pinkHighlightStep(temp_x,temp_y,gameSquares[temp_x][temp_y+1].which_piece);

	gameSquares[temp_x][temp_y+1].setSquare(temp_x,temp_y+1,0,1,gameSquares[temp
_x][temp_y+1].which_piece,gameSquares[temp_x][temp_y+1].kill_him);

	set_data_highlight(temp_x,temp_y+1,gameSquares[temp_x][temp_y+1].which_piece,1);

	gameSquares[temp_x][temp_y+1].printPiece(temp_x,temp_y+1,11,gameSquares[temp_
x][temp_y+1].which_piece);
				}
				else  {
					pinkHighlightStep(temp_x,temp_y,BLANK);

	gameSquares[temp_x][temp_y+1].setSquare(temp_x,temp_y+1,1,1,BLANK,0);
					set_data_highlight(temp_x,temp_y+1,BLANK,1);
				}
			}
			//Sixth Step (X-2,Y+1)
			temp_x = cx - 2;
			temp_y = cy + 0;
			if (temp_x < 1 || temp_x > 8 || temp_y < 0 || temp_y >= 8) { }
```

```
                        else {
                                if (gameSquares[temp_x][temp_y+1].is_empty==0 &&
gameSquares[temp_x][temp_y+1].kill_him==1) { }
                                else if (gameSquares[temp_x][temp_y+1].is_empty==0 &&
gameSquares[temp_x][temp_y+1].kill_him==0) {

        pinkHighlightStep(temp_x,temp_y,gameSquares[temp_x][temp_y+1].which_piece);

        gameSquares[temp_x][temp_y+1].setSquare(temp_x,temp_y+1,0,1,gameSquares[temp
_x][temp_y+1].which_piece,gameSquares[temp_x][temp_y+1].kill_him);

        set_data_highlight(temp_x,temp_y+1,gameSquares[temp_x][temp_y+1].which_piece,1);

        gameSquares[temp_x][temp_y+1].printPiece(temp_x,temp_y+1,11,gameSquares[temp_
x][temp_y+1].which_piece);
                                }
                                else  {
                                        pinkHighlightStep(temp_x,temp_y,BLANK);

        gameSquares[temp_x][temp_y+1].setSquare(temp_x,temp_y+1,1,1,BLANK,0);
                                        set_data_highlight(temp_x,temp_y+1,BLANK,1);
                                }
                        }
                        //Seventh Step (X+2,Y-1)
                        temp_x = cx + 2;
                        temp_y = cy - 2;
                        if (temp_x < 1 || temp_x > 8 || temp_y < 0 || temp_y >= 8) { }
                        else {
                                if (gameSquares[temp_x][temp_y+1].is_empty==0 &&
gameSquares[temp_x][temp_y+1].kill_him==1) { }
                                else if (gameSquares[temp_x][temp_y+1].is_empty==0 &&
gameSquares[temp_x][temp_y+1].kill_him==0) {

        pinkHighlightStep(temp_x,temp_y,gameSquares[temp_x][temp_y+1].which_piece);

        gameSquares[temp_x][temp_y+1].setSquare(temp_x,temp_y+1,0,1,gameSquares[temp
_x][temp_y+1].which_piece,gameSquares[temp_x][temp_y+1].kill_him);

        set_data_highlight(temp_x,temp_y+1,gameSquares[temp_x][temp_y+1].which_piece,1);

        gameSquares[temp_x][temp_y+1].printPiece(temp_x,temp_y+1,11,gameSquares[temp_
x][temp_y+1].which_piece);
                                }
                                else  {
```

```
                                        pinkHighlightStep(temp_x,temp_y,BLANK);

        gameSquares[temp_x][temp_y+1].setSquare(temp_x,temp_y+1,1,1,BLANK,0);
                                        set_data_highlight(temp_x,temp_y+1,BLANK,1);
                                }
                        }
                        //Eighth Step (X+2,Y+1)
                        temp_x = cx + 2;
                        temp_y = cy + 0;
                        if (temp_x < 1 || temp_x > 8 || temp_y < 0 || temp_y >= 8) { }
                        else {
                                if (gameSquares[temp_x][temp_y+1].is_empty==0 &&
gameSquares[temp_x][temp_y+1].kill_him==1) { }
                                else if (gameSquares[temp_x][temp_y+1].is_empty==0 &&
gameSquares[temp_x][temp_y+1].kill_him==0) {

        pinkHighlightStep(temp_x,temp_y,gameSquares[temp_x][temp_y+1].which_piece);

        gameSquares[temp_x][temp_y+1].setSquare(temp_x,temp_y+1,0,1,gameSquares[temp
_x][temp_y+1].which_piece,gameSquares[temp_x][temp_y+1].kill_him);

        set_data_highlight(temp_x,temp_y+1,gameSquares[temp_x][temp_y+1].which_piece,1);

        gameSquares[temp_x][temp_y+1].printPiece(temp_x,temp_y+1,11,gameSquares[temp_
x][temp_y+1].which_piece);
                                }
                                else  {
                                        pinkHighlightStep(temp_x,temp_y,BLANK);

        gameSquares[temp_x][temp_y+1].setSquare(temp_x,temp_y+1,1,1,BLANK,0);
                                        set_data_highlight(temp_x,temp_y+1,BLANK,1);
                                }
                        }
                }
        }
        //_____________________BISHOP STEPS_____________________//
   else if (thisPiece == BISHOP) {
                if (gameSquares[cx][cy].kill_him == 0) {
                   printPiece(cx,cy,14,BISHOP);
                        int temp_x, temp_y;
                        //1st Quadrant (+ve x , +ve y)
                        temp_x = cx; temp_y = cy;
                        if (temp_x >= 8 || temp_y >= 8) { }
                        else {
```

```
                        for (int i=cx+1, j=cy; /*<No Condition>*/ ; i++, j++) {
                                if (i > 8 || j >= 8) { break; }
                                if (gameSquares[i][j+1].is_empty==0 &&
gameSquares[i][j+1].kill_him==0) { break; }
                                if (gameSquares[i][j+1].is_empty==0 &&
gameSquares[i][j+1].kill_him==1) {

        pinkHighlightStep(i,j,gameSquares[i][j+1].which_piece);

        gameSquares[i][j+1].setSquare(i,j+1,0,1,gameSquares[i][j+1].which_piece,gameSquares
[i][j+1].kill_him);

        set_data_highlight(i,j+1,gameSquares[i][j+1].which_piece,1);

        gameSquares[i][j+1].printPiece(i,j+1,11,gameSquares[i][j+1].which_piece);
                                        break;
                                }
                                pinkHighlightStep(i,j,BLANK);
                                gameSquares[i][j+1].setSquare(i,j+1,1,1,BLANK,0);
                                set_data_highlight(i,j+1,BLANK,1);
                        }
                }
                //2nd Quadrant (x -ve , y +ve)
                if (temp_x < 1 || temp_y >= 8) { }
                else {
                        for (int i=cx-1, j=cy; /*<No Condition>*/ ; i--, j++) {
                                if (i < 1 || j > 7) { break; }
                                if (gameSquares[i][j+1].is_empty==0 &&
gameSquares[i][j+1].kill_him==0) { break; }
                                if (gameSquares[i][j+1].is_empty==0 &&
gameSquares[i][j+1].kill_him==1) {

        pinkHighlightStep(i,j,gameSquares[i][j+1].which_piece);

        gameSquares[i][j+1].setSquare(i,j+1,0,1,gameSquares[i][j+1].which_piece,gameSquares
[i][j+1].kill_him);

        set_data_highlight(i,j+1,gameSquares[i][j+1].which_piece,1);

        gameSquares[i][j+1].printPiece(i,j+1,11,gameSquares[i][j+1].which_piece);
                                        break;
                                }
                                pinkHighlightStep(i,j,BLANK);
                                gameSquares[i][j+1].setSquare(i,j+1,1,1,BLANK,0);
```

```
                                        set_data_highlight(i,j+1,BLANK,1);
                                }
                        }
                        //3rd Quadrant (x -ve , y -ve)
                        if (temp_x < 1 || temp_y < 1) { }
                        else {
                                for (int i=cx-1, j=cy-2; /*<No Condition>*/ ; i--, j--) {
                                        if (i <= 0 || j < 0) { break; }
                                        if (gameSquares[i][j+1].is_empty==0 &&
gameSquares[i][j+1].kill_him==0) { break; }
                                        if (gameSquares[i][j+1].is_empty==0 &&
gameSquares[i][j+1].kill_him==1) {

        pinkHighlightStep(i,j,gameSquares[i][j+1].which_piece);

        gameSquares[i][j+1].setSquare(i,j+1,0,1,gameSquares[i][j+1].which_piece,gameSquares
[i][j+1].kill_him);

        set_data_highlight(i,j+1,gameSquares[i][j+1].which_piece,1);

        gameSquares[i][j+1].printPiece(i,j+1,11,gameSquares[i][j+1].which_piece);
                                                break;
                                        }
                                        pinkHighlightStep(i,j,BLANK);
                                        gameSquares[i][j+1].setSquare(i,j+1,1,1,BLANK,0);
                                        set_data_highlight(i,j+1,BLANK,1);
                                }
                        }
                        //4th Quadrant (x +ve , y -ve)
                        if (temp_x >= 8 || temp_y < 1) { }
                        else {
                                for (int i=cx+1, j=cy-2; /*<No Condition>*/ ; i++, j--) {
                                        if (i > 8 || j < 0) { break; }
                                        if (gameSquares[i][j+1].is_empty==0 &&
gameSquares[i][j+1].kill_him==0) { break; }
                                        if (gameSquares[i][j+1].is_empty==0 &&
gameSquares[i][j+1].kill_him==1) {

        pinkHighlightStep(i,j,gameSquares[i][j+1].which_piece);

        gameSquares[i][j+1].setSquare(i,j+1,0,1,gameSquares[i][j+1].which_piece,gameSquares
[i][j+1].kill_him);

        set_data_highlight(i,j+1,gameSquares[i][j+1].which_piece,1);
```

```
gameSquares[i][j+1].printPiece(i,j+1,11,gameSquares[i][j+1].which_piece);
                                        break;
                                }
                                pinkHighlightStep(i,j,BLANK);
                                gameSquares[i][j+1].setSquare(i,j+1,1,1,BLANK,0);
                                set_data_highlight(i,j+1,BLANK,1);
                        }
                }
        }
        else  {                                 //COMPUTER STEPS
           printPiece(cx,cy,14,BISHOP);
                int temp_x, temp_y;
                //1st Quadrant (+ve x , +ve y)
                temp_x = cx; temp_y = cy;
                if (temp_x >= 8 || temp_y >= 8) { }
                else {
                        for (int i=cx+1, j=cy; /*<No Condition>*/ ; i++, j++) {
                                if (i > 8 || j >= 8) { break; }
                                if (gameSquares[i][j+1].is_empty==0 &&
gameSquares[i][j+1].kill_him==1) { break; }
                                if (gameSquares[i][j+1].is_empty==0 &&
gameSquares[i][j+1].kill_him==0) {

        pinkHighlightStep(i,j,gameSquares[i][j+1].which_piece);

        gameSquares[i][j+1].setSquare(i,j+1,0,1,gameSquares[i][j+1].which_piece,gameSquares
[i][j+1].kill_him);

        set_data_highlight(i,j+1,gameSquares[i][j+1].which_piece,1);

        gameSquares[i][j+1].printPiece(i,j+1,11,gameSquares[i][j+1].which_piece);
                                        break;
                                }
                                pinkHighlightStep(i,j,BLANK);
                                gameSquares[i][j+1].setSquare(i,j+1,1,1,BLANK,0);
                                set_data_highlight(i,j+1,BLANK,1);
                        }
                }
                //2nd Quadrant (x -ve , y +ve)
                if (temp_x < 1 || temp_y >= 8) { }
                else {
                        for (int i=cx-1, j=cy; /*<No Condition>*/ ; i--, j++) {
                                if (i < 1 || j > 7) { break; }
```

```
                                        if (gameSquares[i][j+1].is_empty==0 &&
gameSquares[i][j+1].kill_him==1) { break; }
                                        if (gameSquares[i][j+1].is_empty==0 &&
gameSquares[i][j+1].kill_him==0) {

        pinkHighlightStep(i,j,gameSquares[i][j+1].which_piece);

        gameSquares[i][j+1].setSquare(i,j+1,0,1,gameSquares[i][j+1].which_piece,gameSquares
[i][j+1].kill_him);

        set_data_highlight(i,j+1,gameSquares[i][j+1].which_piece,1);

        gameSquares[i][j+1].printPiece(i,j+1,11,gameSquares[i][j+1].which_piece);
                                                break;
                                        }
                                        pinkHighlightStep(i,j,BLANK);
                                        gameSquares[i][j+1].setSquare(i,j+1,1,1,BLANK,0);
                                        set_data_highlight(i,j+1,BLANK,1);
                                }
                        }
                        //3rd Quadrant (x -ve , y -ve)
                        if (temp_x < 1 || temp_y < 1) { }
                        else {
                                for (int i=cx-1, j=cy-2; /*<No Condition>*/ ; i--, j--) {
                                        if (i <= 0 || j < 0) { break; }
                                        if (gameSquares[i][j+1].is_empty==0 &&
gameSquares[i][j+1].kill_him==1) { break; }
                                        if (gameSquares[i][j+1].is_empty==0 &&
gameSquares[i][j+1].kill_him==0) {

        pinkHighlightStep(i,j,gameSquares[i][j+1].which_piece);

        gameSquares[i][j+1].setSquare(i,j+1,0,1,gameSquares[i][j+1].which_piece,gameSquares
[i][j+1].kill_him);

        set_data_highlight(i,j+1,gameSquares[i][j+1].which_piece,1);

        gameSquares[i][j+1].printPiece(i,j+1,11,gameSquares[i][j+1].which_piece);
                                                break;
                                        }
                                        pinkHighlightStep(i,j,BLANK);
                                        gameSquares[i][j+1].setSquare(i,j+1,1,1,BLANK,0);
                                        set_data_highlight(i,j+1,BLANK,1);
                                }
```

```
                        }
                        //4th Quadrant (x +ve , y -ve)
                        if (temp_x >= 8 || temp_y < 1) { }
                        else {
                                for (int i=cx+1, j=cy-2; /*<No Condition>*/ ; i++, j--) {
                                        if (i > 8 || j < 0) { break; }
                                        if (gameSquares[i][j+1].is_empty==0 &&
gameSquares[i][j+1].kill_him==1) { break; }
                                        if (gameSquares[i][j+1].is_empty==0 &&
gameSquares[i][j+1].kill_him==0) {

        pinkHighlightStep(i,j,gameSquares[i][j+1].which_piece);

        gameSquares[i][j+1].setSquare(i,j+1,0,1,gameSquares[i][j+1].which_piece,gameSquares
[i][j+1].kill_him);

        set_data_highlight(i,j+1,gameSquares[i][j+1].which_piece,1);

        gameSquares[i][j+1].printPiece(i,j+1,11,gameSquares[i][j+1].which_piece);
                                                break;
                                        }
                                        pinkHighlightStep(i,j,BLANK);
                                        gameSquares[i][j+1].setSquare(i,j+1,1,1,BLANK,0);
                                        set_data_highlight(i,j+1,BLANK,1);
                                }
                        }
                }
        }
        //_____________________ROOK STEPS_____________________//
   else if (thisPiece == ROOK)
        {
                if (gameSquares[cx][cy].kill_him == 0) {
                        printPiece(cx,cy,14,ROOK);
                        int temp_x, temp_y;
                        //1st loop Increasing x, Constanat y
                        temp_x = cx; temp_y = cy-1;
                        if ( temp_x >= 8 ) { }
                        else {
                                for (int i=cx+1; /*<No Condition>*/ ; i++) {
                                        if (i > 8) { break; }
                                        if (gameSquares[i][temp_y+1].is_empty==0 &&
gameSquares[i][temp_y+1].kill_him==0) { break; }
                                        if (gameSquares[i][temp_y+1].is_empty==0 &&
gameSquares[i][temp_y+1].kill_him==1) {
```

```
	pinkHighlightStep(i,temp_y,gameSquares[i][temp_y+1].which_piece);

	gameSquares[i][temp_y+1].setSquare(i,temp_y+1,0,1,gameSquares[i][temp_y+1].which
_piece,gameSquares[i][temp_y+1].kill_him);

	set_data_highlight(i,temp_y+1,gameSquares[i][temp_y+1].which_piece,1);

	gameSquares[i][temp_y+1].printPiece(i,temp_y+1,11,gameSquares[i][temp_y+1].which_
piece);
                                        break;
                                    }
                                    pinkHighlightStep(i,temp_y,BLANK);

	gameSquares[i][temp_y+1].setSquare(i,temp_y+1,1,1,BLANK,0);
                                    set_data_highlight(i,temp_y+1,BLANK,1);
                                }
                            }
                            //2nd loop Decreasing x, Constanat y
                            temp_x = cx; temp_y = cy-1;
                            if ( temp_x < 1 ) { }
                            else {
                                for (int i=cx-1; /*<No Condition>*/ ; i--) {
                                    if (i < 1) { break; }
                                    if (gameSquares[i][temp_y+1].is_empty==0 &&
gameSquares[i][temp_y+1].kill_him==0) { break; }
                                    if (gameSquares[i][temp_y+1].is_empty==0 &&
gameSquares[i][temp_y+1].kill_him==1) {

	pinkHighlightStep(i,temp_y,gameSquares[i][temp_y+1].which_piece);

	gameSquares[i][temp_y+1].setSquare(i,temp_y+1,0,1,gameSquares[i][temp_y+1].which
_piece,gameSquares[i][temp_y+1].kill_him);

	set_data_highlight(i,temp_y+1,gameSquares[i][temp_y+1].which_piece,1);

	gameSquares[i][temp_y+1].printPiece(i,temp_y+1,11,gameSquares[i][temp_y+1].which_
piece);
                                        break;
                                    }
                                    pinkHighlightStep(i,temp_y,BLANK);

	gameSquares[i][temp_y+1].setSquare(i,temp_y+1,1,1,BLANK,0);
                                    set_data_highlight(i,temp_y+1,BLANK,1);
```

```
                        }
                    }
                    //3rd loop Constant x, Increasing y
                    temp_x = cx; temp_y = cy;
                    if ( temp_y >= 8 ) { }
                    else {
                        for (int i=cy; /*<No Condition>*/ ; i++) {
                            if (i >= 8) { break; }
                            if (gameSquares[temp_x][i+1].is_empty==0 &&
gameSquares[temp_x][i+1].kill_him==0) { break; }
                            if (gameSquares[temp_x][i+1].is_empty==0 &&
gameSquares[temp_x][i+1].kill_him==1) {

        pinkHighlightStep(temp_x,i,gameSquares[temp_x][i+1].which_piece);

        gameSquares[temp_x][i+1].setSquare(temp_x,i+1,0,1,gameSquares[temp_x][i+1].which
_piece,gameSquares[temp_x][i+1].kill_him);

        set_data_highlight(temp_x,i+1,gameSquares[temp_x][i+1].which_piece,1);

        gameSquares[temp_x][i+1].printPiece(temp_x,i+1,11,gameSquares[temp_x][i+1].which_
piece);
                                break;
                            }
                            pinkHighlightStep(temp_x,i,BLANK);

        gameSquares[temp_x][i+1].setSquare(temp_x,i+1,1,1,BLANK,0);
                            set_data_highlight(temp_x,i+1,BLANK,1);
                        }
                    }
                    //4th loop Constant x, Decreasing y
                    temp_x = cx; temp_y = cy;
                    if ( temp_y < 0 ) { }
                    else {
                        for (int i=cy-2; /*<No Condition>*/ ; i--) {
                            if (i < 0) { break; }
                            if (gameSquares[temp_x][i+1].is_empty==0 &&
gameSquares[temp_x][i+1].kill_him==0) { break; }
                            if (gameSquares[temp_x][i+1].is_empty==0 &&
gameSquares[temp_x][i+1].kill_him==1) {

        pinkHighlightStep(temp_x,i,gameSquares[temp_x][i+1].which_piece);
```

```
	gameSquares[temp_x][i+1].setSquare(temp_x,i+1,0,1,gameSquares[temp_x][i+1].which
_piece,gameSquares[temp_x][i+1].kill_him);

	set_data_highlight(temp_x,i+1,gameSquares[temp_x][i+1].which_piece,1);

	gameSquares[temp_x][i+1].printPiece(temp_x,i+1,11,gameSquares[temp_x][i+1].which_
piece);
                                        break;
                                    }
                                    pinkHighlightStep(temp_x,i,BLANK);

	gameSquares[temp_x][i+1].setSquare(temp_x,i+1,1,1,BLANK,0);
                                    set_data_highlight(temp_x,i+1,BLANK,1);
                                }
                        }
                }
	else  {                                 //COMPUTER STEPS
                        printPiece(cx,cy,14,ROOK);
                        int temp_x, temp_y;
                        //1st loop Increasing x, Constanat y
                        temp_x = cx; temp_y = cy-1;
                        if ( temp_x >= 8 ) { }
                        else {
                                for (int i=cx+1; /*<No Condition>*/ ; i++) {
                                    if (i > 8) { break; }
                                    if (gameSquares[i][temp_y+1].is_empty==0 &&
gameSquares[i][temp_y+1].kill_him==1) { break; }
                                    if (gameSquares[i][temp_y+1].is_empty==0 &&
gameSquares[i][temp_y+1].kill_him==0) {

	pinkHighlightStep(i,temp_y,gameSquares[i][temp_y+1].which_piece);

	gameSquares[i][temp_y+1].setSquare(i,temp_y+1,0,1,gameSquares[i][temp_y+1].which
_piece,gameSquares[i][temp_y+1].kill_him);

	set_data_highlight(i,temp_y+1,gameSquares[i][temp_y+1].which_piece,1);

	gameSquares[i][temp_y+1].printPiece(i,temp_y+1,11,gameSquares[i][temp_y+1].which_
piece);
                                        break;
                                    }
                                    pinkHighlightStep(i,temp_y,BLANK);
```

```
	gameSquares[i][temp_y+1].setSquare(i,temp_y+1,1,1,BLANK,0);
					set_data_highlight(i,temp_y+1,BLANK,1);
				}
			}
			//2nd loop Decreasing x, Constanat y
			temp_x = cx; temp_y = cy-1;
			if ( temp_x < 1 ) { }
			else {
				for (int i=cx-1; /*<No Condition>*/ ; i--) {
					if (i < 1) { break; }
					if (gameSquares[i][temp_y+1].is_empty==0 &&
gameSquares[i][temp_y+1].kill_him==1) { break; }
					if (gameSquares[i][temp_y+1].is_empty==0 &&
gameSquares[i][temp_y+1].kill_him==0) {

	pinkHighlightStep(i,temp_y,gameSquares[i][temp_y+1].which_piece);

	gameSquares[i][temp_y+1].setSquare(i,temp_y+1,0,1,gameSquares[i][temp_y+1].which
_piece,gameSquares[i][temp_y+1].kill_him);

	set_data_highlight(i,temp_y+1,gameSquares[i][temp_y+1].which_piece,1);

	gameSquares[i][temp_y+1].printPiece(i,temp_y+1,11,gameSquares[i][temp_y+1].which_
piece);
						break;
					}
					pinkHighlightStep(i,temp_y,BLANK);

	gameSquares[i][temp_y+1].setSquare(i,temp_y+1,1,1,BLANK,0);
					set_data_highlight(i,temp_y+1,BLANK,1);
				}
			}
			//3rd loop Constant x, Increasing y
			temp_x = cx; temp_y = cy;
			if ( temp_y >= 8 ) { }
			else {
				for (int i=cy; /*<No Condition>*/ ; i++) {
					if (i >= 8) { break; }
					if (gameSquares[temp_x][i+1].is_empty==0 &&
gameSquares[temp_x][i+1].kill_him==1) { break; }
					if (gameSquares[temp_x][i+1].is_empty==0 &&
gameSquares[temp_x][i+1].kill_him==0) {
```

```
	pinkHighlightStep(temp_x,i,gameSquares[temp_x][i+1].which_piece);

	gameSquares[temp_x][i+1].setSquare(temp_x,i+1,0,1,gameSquares[temp_x][i+1].which
_piece,gameSquares[temp_x][i+1].kill_him);

	set_data_highlight(temp_x,i+1,gameSquares[temp_x][i+1].which_piece,1);

	gameSquares[temp_x][i+1].printPiece(temp_x,i+1,11,gameSquares[temp_x][i+1].which_
piece);
					break;
				}
				pinkHighlightStep(temp_x,i,BLANK);

	gameSquares[temp_x][i+1].setSquare(temp_x,i+1,1,1,BLANK,0);
				set_data_highlight(temp_x,i+1,BLANK,1);
			}
		}
		//4th loop Constant x, Decreasing y
		temp_x = cx; temp_y = cy;
		if ( temp_y < 0 ) { }
		else {
			for (int i=cy-2; /*<No Condition>*/ ; i--) {
				if (i < 0) { break; }
				if (gameSquares[temp_x][i+1].is_empty==0 &&
gameSquares[temp_x][i+1].kill_him==1) { break; }
				if (gameSquares[temp_x][i+1].is_empty==0 &&
gameSquares[temp_x][i+1].kill_him==0) {

	pinkHighlightStep(temp_x,i,gameSquares[temp_x][i+1].which_piece);

	gameSquares[temp_x][i+1].setSquare(temp_x,i+1,0,1,gameSquares[temp_x][i+1].which
_piece,gameSquares[temp_x][i+1].kill_him);

	set_data_highlight(temp_x,i+1,gameSquares[temp_x][i+1].which_piece,1);

	gameSquares[temp_x][i+1].printPiece(temp_x,i+1,11,gameSquares[temp_x][i+1].which_
piece);
					break;
				}
				pinkHighlightStep(temp_x,i,BLANK);

	gameSquares[temp_x][i+1].setSquare(temp_x,i+1,1,1,BLANK,0);
				set_data_highlight(temp_x,i+1,BLANK,1);
```

```
                                        }
                                }
                        }
                }
                //_____________________QUEEN STEPS_____________________//
        else if (thisPiece == QUEEN) {
                        if (gameSquares[cx][cy].kill_him == 0) {
                                printPiece(cx,cy,14,QUEEN);
                                int temp_x, temp_y;
                                //1st Quadrant (+ve x , +ve y)
                                temp_x = cx; temp_y = cy;
                                if (temp_x >= 8 || temp_y >= 8) { }
                                else {
                                        for (int i=cx+1, j=cy; /*<No Condition>*/ ; i++, j++) {
                                                if (i > 8 || j >= 8) { break; }
                                                if (gameSquares[i][j+1].is_empty==0 &&
gameSquares[i][j+1].kill_him==0) { break; }
                                                if (gameSquares[i][j+1].is_empty==0 &&
gameSquares[i][j+1].kill_him==1) {

                pinkHighlightStep(i,j,gameSquares[i][j+1].which_piece);

                gameSquares[i][j+1].setSquare(i,j+1,0,1,gameSquares[i][j+1].which_piece,gameSquares
[i][j+1].kill_him);

                set_data_highlight(i,j+1,gameSquares[i][j+1].which_piece,1);

                gameSquares[i][j+1].printPiece(i,j+1,11,gameSquares[i][j+1].which_piece);
                                                        break;
                                                }
                                                pinkHighlightStep(i,j,BLANK);
                                                gameSquares[i][j+1].setSquare(i,j+1,1,1,BLANK,0);
                                                set_data_highlight(i,j+1,BLANK,1);
                                        }
                                }
                                //2nd Quadrant (x -ve , y +ve)
                                if (temp_x < 1 || temp_y >= 8) { }
                                else {
                                        for (int i=cx-1, j=cy; /*<No Condition>*/ ; i--, j++) {
                                                if (i < 1 || j > 7) { break; }
                                                if (gameSquares[i][j+1].is_empty==0 &&
gameSquares[i][j+1].kill_him==0) { break; }
                                                if (gameSquares[i][j+1].is_empty==0 &&
gameSquares[i][j+1].kill_him==1) {
```

```
		pinkHighlightStep(i,j,gameSquares[i][j+1].which_piece);

		gameSquares[i][j+1].setSquare(i,j+1,0,1,gameSquares[i][j+1].which_piece,gameSquares
[i][j+1].kill_him);

		set_data_highlight(i,j+1,gameSquares[i][j+1].which_piece,1);

		gameSquares[i][j+1].printPiece(i,j+1,11,gameSquares[i][j+1].which_piece);
                                                break;
                                        }
                                        pinkHighlightStep(i,j,BLANK);
                                        gameSquares[i][j+1].setSquare(i,j+1,1,1,BLANK,0);
                                        set_data_highlight(i,j+1,BLANK,1);
                                }
                        }
                        //3rd Quadrant (x -ve , y -ve)
                        if (temp_x < 1 || temp_y < 1) { }
                        else {
                                for (int i=cx-1, j=cy-2; /*<No Condition>*/ ; i--, j--) {
                                        if (i <= 0 || j < 0) { break; }
                                        if (gameSquares[i][j+1].is_empty==0 &&
gameSquares[i][j+1].kill_him==0) { break; }
                                        if (gameSquares[i][j+1].is_empty==0 &&
gameSquares[i][j+1].kill_him==1) {

		pinkHighlightStep(i,j,gameSquares[i][j+1].which_piece);

		gameSquares[i][j+1].setSquare(i,j+1,0,1,gameSquares[i][j+1].which_piece,gameSquares
[i][j+1].kill_him);

		set_data_highlight(i,j+1,gameSquares[i][j+1].which_piece,1);

		gameSquares[i][j+1].printPiece(i,j+1,11,gameSquares[i][j+1].which_piece);
                                                break;
                                        }
                                        pinkHighlightStep(i,j,BLANK);
                                        gameSquares[i][j+1].setSquare(i,j+1,1,1,BLANK,0);
                                        set_data_highlight(i,j+1,BLANK,1);
                                }
                        }
                        //4th Quadrant (x +ve , y -ve)
                        if (temp_x >= 8 || temp_y < 1) { }
                        else {
```

```
                        for (int i=cx+1, j=cy-2; /*<No Condition>*/ ; i++, j--) {
                                if (i > 8 || j < 0) { break; }
                                if (gameSquares[i][j+1].is_empty==0 &&
gameSquares[i][j+1].kill_him==0) { break; }
                                if (gameSquares[i][j+1].is_empty==0 &&
gameSquares[i][j+1].kill_him==1) {

        pinkHighlightStep(i,j,gameSquares[i][j+1].which_piece);

        gameSquares[i][j+1].setSquare(i,j+1,0,1,gameSquares[i][j+1].which_piece,gameSquares
[i][j+1].kill_him);

        set_data_highlight(i,j+1,gameSquares[i][j+1].which_piece,1);

        gameSquares[i][j+1].printPiece(i,j+1,11,gameSquares[i][j+1].which_piece);
                                        break;
                                }
                                pinkHighlightStep(i,j,BLANK);
                                gameSquares[i][j+1].setSquare(i,j+1,1,1,BLANK,0);
                                set_data_highlight(i,j+1,BLANK,1);
                        }
                }
                ////////////////////////////////////////
                //1st loop Increasing x, Constanat y
                temp_x = cx; temp_y = cy-1;
                if ( temp_x >= 8 ) { }
                else {
                        for (int i=cx+1; /*<No Condition>*/ ; i++) {
                                if (i > 8) { break; }
                                if (gameSquares[i][temp_y+1].is_empty==0 &&
gameSquares[i][temp_y+1].kill_him==0) { break; }
                                if (gameSquares[i][temp_y+1].is_empty==0 &&
gameSquares[i][temp_y+1].kill_him==1) {

        pinkHighlightStep(i,temp_y,gameSquares[i][temp_y+1].which_piece);

        gameSquares[i][temp_y+1].setSquare(i,temp_y+1,0,1,gameSquares[i][temp_y+1].which
_piece,gameSquares[i][temp_y+1].kill_him);

        set_data_highlight(i,temp_y+1,gameSquares[i][temp_y+1].which_piece,1);

        gameSquares[i][temp_y+1].printPiece(i,temp_y+1,11,gameSquares[i][temp_y+1].which_
piece);
                                        break;
```

```
                                        }
                                        pinkHighlightStep(i,temp_y,BLANK);

        gameSquares[i][temp_y+1].setSquare(i,temp_y+1,1,1,BLANK,0);
                                        set_data_highlight(i,temp_y+1,BLANK,1);
                                }
                        }
                        //2nd loop Decreasing x, Constanat y
                        temp_x = cx; temp_y = cy-1;
                        if ( temp_x < 1 ) { }
                        else {
                                for (int i=cx-1; /*<No Condition>*/ ; i--) {
                                        if (i < 1) { break; }
                                        if (gameSquares[i][temp_y+1].is_empty==0 &&
gameSquares[i][temp_y+1].kill_him==0) { break; }
                                        if (gameSquares[i][temp_y+1].is_empty==0 &&
gameSquares[i][temp_y+1].kill_him==1) {

        pinkHighlightStep(i,temp_y,gameSquares[i][temp_y+1].which_piece);

        gameSquares[i][temp_y+1].setSquare(i,temp_y+1,0,1,gameSquares[i][temp_y+1].which
_piece,gameSquares[i][temp_y+1].kill_him);

        set_data_highlight(i,temp_y+1,gameSquares[i][temp_y+1].which_piece,1);

        gameSquares[i][temp_y+1].printPiece(i,temp_y+1,11,gameSquares[i][temp_y+1].which_
piece);
                                                break;
                                        }
                                        pinkHighlightStep(i,temp_y,BLANK);

        gameSquares[i][temp_y+1].setSquare(i,temp_y+1,1,1,BLANK,0);
                                        set_data_highlight(i,temp_y+1,BLANK,1);
                                }
                        }
                        //3rd loop Constant x, Increasing y
                        temp_x = cx; temp_y = cy;
                        if ( temp_y >= 8 ) { }
                        else {
                                for (int i=cy; /*<No Condition>*/ ; i++) {
                                        if (i >= 8) { break; }
                                        if (gameSquares[temp_x][i+1].is_empty==0 &&
gameSquares[temp_x][i+1].kill_him==0) { break; }
```

```
                                        if (gameSquares[temp_x][i+1].is_empty==0 &&
gameSquares[temp_x][i+1].kill_him==1) {

        pinkHighlightStep(temp_x,i,gameSquares[temp_x][i+1].which_piece);

        gameSquares[temp_x][i+1].setSquare(temp_x,i+1,0,1,gameSquares[temp_x][i+1].which
_piece,gameSquares[temp_x][i+1].kill_him);

        set_data_highlight(temp_x,i+1,gameSquares[temp_x][i+1].which_piece,1);

        gameSquares[temp_x][i+1].printPiece(temp_x,i+1,11,gameSquares[temp_x][i+1].which_
piece);
                                                break;
                                        }
                                        pinkHighlightStep(temp_x,i,BLANK);

        gameSquares[temp_x][i+1].setSquare(temp_x,i+1,1,1,BLANK,0);
                                        set_data_highlight(temp_x,i+1,BLANK,1);
                                }
                        }
                        //4th loop Constant x, Decreasing y
                        temp_x = cx; temp_y = cy;
                        if ( temp_y < 0 ) { }
                        else {
                                for (int i=cy-2; /*<No Condition>*/ ; i--) {
                                        if (i < 0) { break; }
                                        if (gameSquares[temp_x][i+1].is_empty==0 &&
gameSquares[temp_x][i+1].kill_him==0) { break; }
                                        if (gameSquares[temp_x][i+1].is_empty==0 &&
gameSquares[temp_x][i+1].kill_him==1) {

        pinkHighlightStep(temp_x,i,gameSquares[temp_x][i+1].which_piece);

        gameSquares[temp_x][i+1].setSquare(temp_x,i+1,0,1,gameSquares[temp_x][i+1].which
_piece,gameSquares[temp_x][i+1].kill_him);

        set_data_highlight(temp_x,i+1,gameSquares[temp_x][i+1].which_piece,1);

        gameSquares[temp_x][i+1].printPiece(temp_x,i+1,11,gameSquares[temp_x][i+1].which_
piece);
                                                break;
                                        }
                                        pinkHighlightStep(temp_x,i,BLANK);
```

```
        gameSquares[temp_x][i+1].setSquare(temp_x,i+1,1,1,BLANK,0);
                                        set_data_highlight(temp_x,i+1,BLANK,1);
                                }
                        }
                }
                        else  {                         //COMPUTER STEPS
                                printPiece(cx,cy,14,QUEEN);
                                int temp_x, temp_y;
                                //1st Quadrant (+ve x , +ve y)
                                temp_x = cx; temp_y = cy;
                                if (temp_x >= 8 || temp_y >= 8) { }
                                else {
                                        for (int i=cx+1, j=cy; /*<No Condition>*/ ; i++, j++) {
                                                if (i > 8 || j >= 8) { break; }
                                                if (gameSquares[i][j+1].is_empty==0 &&
gameSquares[i][j+1].kill_him==1) { break; }
                                                if (gameSquares[i][j+1].is_empty==0 &&
gameSquares[i][j+1].kill_him==0) {

        pinkHighlightStep(i,j,gameSquares[i][j+1].which_piece);

        gameSquares[i][j+1].setSquare(i,j+1,0,1,gameSquares[i][j+1].which_piece,gameSquares
[i][j+1].kill_him);

        set_data_highlight(i,j+1,gameSquares[i][j+1].which_piece,1);

        gameSquares[i][j+1].printPiece(i,j+1,11,gameSquares[i][j+1].which_piece);
                                                        break;
                                                }
                                                pinkHighlightStep(i,j,BLANK);
                                                gameSquares[i][j+1].setSquare(i,j+1,1,1,BLANK,0);
                                                set_data_highlight(i,j+1,BLANK,1);
                                        }
                                }
                                //2nd Quadrant (x -ve , y +ve)
                                if (temp_x < 1 || temp_y >= 8) { }
                                else {
                                        for (int i=cx-1, j=cy; /*<No Condition>*/ ; i--, j++) {
                                                if (i < 1 || j > 7) { break; }
                                                if (gameSquares[i][j+1].is_empty==0 &&
gameSquares[i][j+1].kill_him==1) { break; }
                                                if (gameSquares[i][j+1].is_empty==0 &&
gameSquares[i][j+1].kill_him==0) {
```

```
		pinkHighlightStep(i,j,gameSquares[i][j+1].which_piece);

		gameSquares[i][j+1].setSquare(i,j+1,0,1,gameSquares[i][j+1].which_piece,gameSquares
[i][j+1].kill_him);

		set_data_highlight(i,j+1,gameSquares[i][j+1].which_piece,1);

		gameSquares[i][j+1].printPiece(i,j+1,11,gameSquares[i][j+1].which_piece);
                                                break;
                                        }
                                        pinkHighlightStep(i,j,BLANK);
                                        gameSquares[i][j+1].setSquare(i,j+1,1,1,BLANK,0);
                                        set_data_highlight(i,j+1,BLANK,1);
                                }
                        }
                        //3rd Quadrant (x -ve , y -ve)
                        if (temp_x < 1 || temp_y < 1) { }
                        else {
                                for (int i=cx-1, j=cy-2; /*<No Condition>*/ ; i--, j--) {
                                        if (i <= 0 || j < 0) { break; }
                                        if (gameSquares[i][j+1].is_empty==0 &&
gameSquares[i][j+1].kill_him==1) { break; }
                                        if (gameSquares[i][j+1].is_empty==0 &&
gameSquares[i][j+1].kill_him==0) {

		pinkHighlightStep(i,j,gameSquares[i][j+1].which_piece);

		gameSquares[i][j+1].setSquare(i,j+1,0,1,gameSquares[i][j+1].which_piece,gameSquares
[i][j+1].kill_him);

		set_data_highlight(i,j+1,gameSquares[i][j+1].which_piece,1);

		gameSquares[i][j+1].printPiece(i,j+1,11,gameSquares[i][j+1].which_piece);
                                                break;
                                        }
                                        pinkHighlightStep(i,j,BLANK);
                                        gameSquares[i][j+1].setSquare(i,j+1,1,1,BLANK,0);
                                        set_data_highlight(i,j+1,BLANK,1);
                                }
                        }
                        //4th Quadrant (x +ve , y -ve)
                        if (temp_x >= 8 || temp_y < 1) { }
                        else {
```

```
                        for (int i=cx+1, j=cy-2; /*<No Condition>*/ ; i++, j--) {
                                if (i > 8 || j < 0) { break; }
                                if (gameSquares[i][j+1].is_empty==0 &&
gameSquares[i][j+1].kill_him==1) { break; }
                                if (gameSquares[i][j+1].is_empty==0 &&
gameSquares[i][j+1].kill_him==0) {

        pinkHighlightStep(i,j,gameSquares[i][j+1].which_piece);

        gameSquares[i][j+1].setSquare(i,j+1,0,1,gameSquares[i][j+1].which_piece,gameSquares
[i][j+1].kill_him);

        set_data_highlight(i,j+1,gameSquares[i][j+1].which_piece,1);

        gameSquares[i][j+1].printPiece(i,j+1,11,gameSquares[i][j+1].which_piece);
                                        break;
                                }
                                pinkHighlightStep(i,j,BLANK);
                                gameSquares[i][j+1].setSquare(i,j+1,1,1,BLANK,0);
                                set_data_highlight(i,j+1,BLANK,1);
                        }
                }
                ////////////////////////////////////////
                //1st loop Increasing x, Constanat y
                temp_x = cx; temp_y = cy-1;
                if ( temp_x >= 8 ) { }
                else {
                        for (int i=cx+1; /*<No Condition>*/ ; i++) {
                                if (i > 8) { break; }
                                if (gameSquares[i][temp_y+1].is_empty==0 &&
gameSquares[i][temp_y+1].kill_him==1) { break; }
                                if (gameSquares[i][temp_y+1].is_empty==0 &&
gameSquares[i][temp_y+1].kill_him==0) {

        pinkHighlightStep(i,temp_y,gameSquares[i][temp_y+1].which_piece);

        gameSquares[i][temp_y+1].setSquare(i,temp_y+1,0,1,gameSquares[i][temp_y+1].which
_piece,gameSquares[i][temp_y+1].kill_him);

        set_data_highlight(i,temp_y+1,gameSquares[i][temp_y+1].which_piece,1);

        gameSquares[i][temp_y+1].printPiece(i,temp_y+1,11,gameSquares[i][temp_y+1].which_
piece);
                                        break;
```

```
                                        }
                                        pinkHighlightStep(i,temp_y,BLANK);

        gameSquares[i][temp_y+1].setSquare(i,temp_y+1,1,1,BLANK,0);
                                        set_data_highlight(i,temp_y+1,BLANK,1);
                                }
                        }
                        //2nd loop Decreasing x, Constanat y
                        temp_x = cx; temp_y = cy-1;
                        if ( temp_x < 1 ) { }
                        else {
                                for (int i=cx-1; /*<No Condition>*/ ; i--) {
                                        if (i < 1) { break; }
                                        if (gameSquares[i][temp_y+1].is_empty==0 &&
gameSquares[i][temp_y+1].kill_him==1) { break; }
                                        if (gameSquares[i][temp_y+1].is_empty==0 &&
gameSquares[i][temp_y+1].kill_him==0) {

        pinkHighlightStep(i,temp_y,gameSquares[i][temp_y+1].which_piece);

        gameSquares[i][temp_y+1].setSquare(i,temp_y+1,0,1,gameSquares[i][temp_y+1].which
_piece,gameSquares[i][temp_y+1].kill_him);

        set_data_highlight(i,temp_y+1,gameSquares[i][temp_y+1].which_piece,1);

        gameSquares[i][temp_y+1].printPiece(i,temp_y+1,11,gameSquares[i][temp_y+1].which_
piece);
                                                break;
                                        }
                                        pinkHighlightStep(i,temp_y,BLANK);

        gameSquares[i][temp_y+1].setSquare(i,temp_y+1,1,1,BLANK,0);
                                        set_data_highlight(i,temp_y+1,BLANK,1);
                                }
                        }
                        //3rd loop Constant x, Increasing y
                        temp_x = cx; temp_y = cy;
                        if ( temp_y >= 8 ) { }
                        else {
                                for (int i=cy; /*<No Condition>*/ ; i++) {
                                        if (i >= 8) { break; }
                                        if (gameSquares[temp_x][i+1].is_empty==0 &&
gameSquares[temp_x][i+1].kill_him==1) { break; }
```

```
                                        if (gameSquares[temp_x][i+1].is_empty==0 &&
gameSquares[temp_x][i+1].kill_him==0) {

        pinkHighlightStep(temp_x,i,gameSquares[temp_x][i+1].which_piece);

        gameSquares[temp_x][i+1].setSquare(temp_x,i+1,0,1,gameSquares[temp_x][i+1].which
_piece,gameSquares[temp_x][i+1].kill_him);

        set_data_highlight(temp_x,i+1,gameSquares[temp_x][i+1].which_piece,1);

        gameSquares[temp_x][i+1].printPiece(temp_x,i+1,11,gameSquares[temp_x][i+1].which_
piece);
                                                break;
                                        }
                                        pinkHighlightStep(temp_x,i,BLANK);

        gameSquares[temp_x][i+1].setSquare(temp_x,i+1,1,1,BLANK,0);
                                        set_data_highlight(temp_x,i+1,BLANK,1);
                                }
                        }
                        //4th loop Constant x, Decreasing y
                        temp_x = cx; temp_y = cy;
                        if ( temp_y < 0 ) { }
                        else {
                                for (int i=cy-2; /*<No Condition>*/ ; i--) {
                                        if (i < 0) { break; }
                                        if (gameSquares[temp_x][i+1].is_empty==0 &&
gameSquares[temp_x][i+1].kill_him==1) { break; }
                                        if (gameSquares[temp_x][i+1].is_empty==0 &&
gameSquares[temp_x][i+1].kill_him==0) {

        pinkHighlightStep(temp_x,i,gameSquares[temp_x][i+1].which_piece);

        gameSquares[temp_x][i+1].setSquare(temp_x,i+1,0,1,gameSquares[temp_x][i+1].which
_piece,gameSquares[temp_x][i+1].kill_him);

        set_data_highlight(temp_x,i+1,gameSquares[temp_x][i+1].which_piece,1);

        gameSquares[temp_x][i+1].printPiece(temp_x,i+1,11,gameSquares[temp_x][i+1].which_
piece);
                                                break;
                                        }
                                        pinkHighlightStep(temp_x,i,BLANK);
```

```
        gameSquares[temp_x][i+1].setSquare(temp_x,i+1,1,1,BLANK,0);
                                        set_data_highlight(temp_x,i+1,BLANK,1);
                                }
                        }
                   }
        }
        //_____________________KING STEPS_____________________//
   else if (thisPiece == KING) {
                if (gameSquares[cx][cy].kill_him == 0) {
                        printPiece(cx,cy,14,KING);
                        int temp_x, temp_y;
                        //Right-Upper Step
                        temp_x = cx; temp_y = cy;
                        if (temp_x >= 8 || temp_y >= 8) { }
                        else {
                                temp_x = cx+1; temp_y = cy;
                                if (gameSquares[temp_x][temp_y+1].is_empty==0 &&
gameSquares[temp_x][temp_y+1].kill_him==0) { }
                                else if (gameSquares[temp_x][temp_y+1].is_empty==0 &&
gameSquares[temp_x][temp_y+1].kill_him==1) {

        pinkHighlightStep(temp_x,temp_y,gameSquares[temp_x][temp_y+1].which_piece);

        gameSquares[temp_x][temp_y+1].setSquare(temp_x,temp_y+1,0,1,gameSquares[temp
_x][temp_y+1].which_piece,1);

        set_data_highlight(temp_x,temp_y+1,gameSquares[temp_x][temp_y+1].which_piece,1);

        gameSquares[temp_x][temp_y+1].printPiece(temp_x,temp_y+1,11,gameSquares[temp_
x][temp_y+1].which_piece);
                                }
                                else {
                                        pinkHighlightStep(temp_x,temp_y,BLANK);

        gameSquares[temp_x][temp_y+1].setSquare(temp_x,temp_y+1,1,1,BLANK,0);
                                        set_data_highlight(temp_x,temp_y+1,BLANK,1);
                                }
                        }
                        //Left-Upper Step
                        temp_x = cx; temp_y = cy;
                        if (temp_x <= 1 || temp_y >= 8) { }
                        else {
                                temp_x = cx-1; temp_y = cy;
```

```
                        if (gameSquares[temp_x][temp_y+1].is_empty==0 &&
gameSquares[temp_x][temp_y+1].kill_him==0) { }
                        else if (gameSquares[temp_x][temp_y+1].is_empty==0 &&
gameSquares[temp_x][temp_y+1].kill_him==1) {

        pinkHighlightStep(temp_x,temp_y,gameSquares[temp_x][temp_y+1].which_piece);

        gameSquares[temp_x][temp_y+1].setSquare(temp_x,temp_y+1,0,1,gameSquares[temp
_x][temp_y+1].which_piece,1);

        set_data_highlight(temp_x,temp_y+1,gameSquares[temp_x][temp_y+1].which_piece,1);

        gameSquares[temp_x][temp_y+1].printPiece(temp_x,temp_y+1,11,gameSquares[temp_
x][temp_y+1].which_piece);
                        }
                        else {
                                pinkHighlightStep(temp_x,temp_y,BLANK);

        gameSquares[temp_x][temp_y+1].setSquare(temp_x,temp_y+1,1,1,BLANK,0);
                                set_data_highlight(temp_x,temp_y+1,BLANK,1);
                        }
                }
                //Right-Lower Step
                temp_x = cx; temp_y = cy;
                if (temp_x >= 8 || temp_y <= 1) { }
                else {
                        temp_x = cx+1; temp_y = cy-2;
                        if (gameSquares[temp_x][temp_y+1].is_empty==0 &&
gameSquares[temp_x][temp_y+1].kill_him==0) { }
                        else if (gameSquares[temp_x][temp_y+1].is_empty==0 &&
gameSquares[temp_x][temp_y+1].kill_him==1) {

        pinkHighlightStep(temp_x,temp_y,gameSquares[temp_x][temp_y+1].which_piece);

        gameSquares[temp_x][temp_y+1].setSquare(temp_x,temp_y+1,0,1,gameSquares[temp
_x][temp_y+1].which_piece,1);

        set_data_highlight(temp_x,temp_y+1,gameSquares[temp_x][temp_y+1].which_piece,1);

        gameSquares[temp_x][temp_y+1].printPiece(temp_x,temp_y+1,11,gameSquares[temp_
x][temp_y+1].which_piece);
                        }
                        else {
                                pinkHighlightStep(temp_x,temp_y,BLANK);
```

```
gameSquares[temp_x][temp_y+1].setSquare(temp_x,temp_y+1,1,1,BLANK,0);
                                        set_data_highlight(temp_x,temp_y+1,BLANK,1);
                                }
                        }
                        //Left-Lower Step
                        temp_x = cx; temp_y = cy;
                        if (temp_x <= 1 || temp_y <= 1) { }
                        else {
                                temp_x = cx-1; temp_y = cy-2;
                                if (gameSquares[temp_x][temp_y+1].is_empty==0 &&
gameSquares[temp_x][temp_y+1].kill_him==0) { }
                                else if (gameSquares[temp_x][temp_y+1].is_empty==0 &&
gameSquares[temp_x][temp_y+1].kill_him==1) {

        pinkHighlightStep(temp_x,temp_y,gameSquares[temp_x][temp_y+1].which_piece);

        gameSquares[temp_x][temp_y+1].setSquare(temp_x,temp_y+1,0,1,gameSquares[temp
_x][temp_y+1].which_piece,1);

        set_data_highlight(temp_x,temp_y+1,gameSquares[temp_x][temp_y+1].which_piece,1);

        gameSquares[temp_x][temp_y+1].printPiece(temp_x,temp_y+1,11,gameSquares[temp_
x][temp_y+1].which_piece);
                                }
                                else {
                                        pinkHighlightStep(temp_x,temp_y,BLANK);

        gameSquares[temp_x][temp_y+1].setSquare(temp_x,temp_y+1,1,1,BLANK,0);
                                        set_data_highlight(temp_x,temp_y+1,BLANK,1);
                                }
                        }
                        //Right Step
                        temp_x = cx; temp_y = cy;
                        if ( temp_x >= 8 ) { }
                        else {
                                temp_x = cx+1; temp_y = cy-1;
                                if (gameSquares[temp_x][temp_y+1].is_empty==0 &&
gameSquares[temp_x][temp_y+1].kill_him==0) { }
                                else if (gameSquares[temp_x][temp_y+1].is_empty==0 &&
gameSquares[temp_x][temp_y+1].kill_him==1) {

        pinkHighlightStep(temp_x,temp_y,gameSquares[temp_x][temp_y+1].which_piece);
```

```
	gameSquares[temp_x][temp_y+1].setSquare(temp_x,temp_y+1,0,1,gameSquares[temp
_x][temp_y+1].which_piece,1);

	set_data_highlight(temp_x,temp_y+1,gameSquares[temp_x][temp_y+1].which_piece,1);

	gameSquares[temp_x][temp_y+1].printPiece(temp_x,temp_y+1,11,gameSquares[temp_
x][temp_y+1].which_piece);
				}
				else {
					pinkHighlightStep(temp_x,temp_y,BLANK);

	gameSquares[temp_x][temp_y+1].setSquare(temp_x,temp_y+1,1,1,BLANK,0);
					set_data_highlight(temp_x,temp_y+1,BLANK,1);
				}
			}
			//Left Step
			temp_x = cx; temp_y = cy;
			if ( temp_x <= 1 ) { }
			else {
				temp_x = cx-1; temp_y = cy-1;
				if (gameSquares[temp_x][temp_y+1].is_empty==0 &&
gameSquares[temp_x][temp_y+1].kill_him==0) { }
				else if (gameSquares[temp_x][temp_y+1].is_empty==0 &&
gameSquares[temp_x][temp_y+1].kill_him==1) {

	pinkHighlightStep(temp_x,temp_y,gameSquares[temp_x][temp_y+1].which_piece);

	gameSquares[temp_x][temp_y+1].setSquare(temp_x,temp_y+1,0,1,gameSquares[temp
_x][temp_y+1].which_piece,1);

	set_data_highlight(temp_x,temp_y+1,gameSquares[temp_x][temp_y+1].which_piece,1);

	gameSquares[temp_x][temp_y+1].printPiece(temp_x,temp_y+1,11,gameSquares[temp_
x][temp_y+1].which_piece);
				}
				else {
					pinkHighlightStep(temp_x,temp_y,BLANK);

	gameSquares[temp_x][temp_y+1].setSquare(temp_x,temp_y+1,1,1,BLANK,0);
					set_data_highlight(temp_x,temp_y+1,BLANK,1);
				}
			}
			//Up Step
```

```
                        temp_x = cx; temp_y = cy;
                        if ( temp_y >= 8 ) { }
                        else {
                                temp_x = cx; temp_y = cy;
                                if (gameSquares[temp_x][temp_y+1].is_empty==0 &&
gameSquares[temp_x][temp_y+1].kill_him==0) { }
                                else if (gameSquares[temp_x][temp_y+1].is_empty==0 &&
gameSquares[temp_x][temp_y+1].kill_him==1) {

        pinkHighlightStep(temp_x,temp_y,gameSquares[temp_x][temp_y+1].which_piece);

        gameSquares[temp_x][temp_y+1].setSquare(temp_x,temp_y+1,0,1,gameSquares[temp
_x][temp_y+1].which_piece,1);

        set_data_highlight(temp_x,temp_y+1,gameSquares[temp_x][temp_y+1].which_piece,1);

        gameSquares[temp_x][temp_y+1].printPiece(temp_x,temp_y+1,11,gameSquares[temp_
x][temp_y+1].which_piece);
                                }
                                else {
                                        pinkHighlightStep(temp_x,temp_y,BLANK);

        gameSquares[temp_x][temp_y+1].setSquare(temp_x,temp_y+1,1,1,BLANK,0);
                                        set_data_highlight(temp_x,temp_y+1,BLANK,1);
                                }
                        }
                        //Down Step
                        temp_x = cx; temp_y = cy;
                        if ( temp_y <= 1 ) { }
                        else {
                                temp_x = cx; temp_y = cy-2;
                                if (gameSquares[temp_x][temp_y+1].is_empty==0 &&
gameSquares[temp_x][temp_y+1].kill_him==0) { }
                                else if (gameSquares[temp_x][temp_y+1].is_empty==0 &&
gameSquares[temp_x][temp_y+1].kill_him==1) {

        pinkHighlightStep(temp_x,temp_y,gameSquares[temp_x][temp_y+1].which_piece);

        gameSquares[temp_x][temp_y+1].setSquare(temp_x,temp_y+1,0,1,gameSquares[temp
_x][temp_y+1].which_piece,1);

        set_data_highlight(temp_x,temp_y+1,gameSquares[temp_x][temp_y+1].which_piece,1);
```

```
	gameSquares[temp_x][temp_y+1].printPiece(temp_x,temp_y+1,11,gameSquares[temp_
x][temp_y+1].which_piece);
                                }
                                else {
                                        pinkHighlightStep(temp_x,temp_y,BLANK);

	gameSquares[temp_x][temp_y+1].setSquare(temp_x,temp_y+1,1,1,BLANK,0);
                                        set_data_highlight(temp_x,temp_y+1,BLANK,1);
                                }
                        }
                }
                else if (gameSquares[cx][cy].kill_him == 1) {
                        printPiece(cx,cy,14,KING);
                        int temp_x, temp_y;
                        //Right-Upper Step
                        temp_x = cx; temp_y = cy;
                        if (temp_x >= 8 || temp_y >= 8) { }
                        else {
                                temp_x = cx+1; temp_y = cy;
                                if (gameSquares[temp_x][temp_y+1].is_empty==0 &&
gameSquares[temp_x][temp_y+1].kill_him==1) { }
                                else if (gameSquares[temp_x][temp_y+1].is_empty==0 &&
gameSquares[temp_x][temp_y+1].kill_him==0) {

	pinkHighlightStep(temp_x,temp_y,gameSquares[temp_x][temp_y+1].which_piece);

	gameSquares[temp_x][temp_y+1].setSquare(temp_x,temp_y+1,0,1,gameSquares[temp
_x][temp_y+1].which_piece,gameSquares[temp_x][temp_y+1].kill_him);

	set_data_highlight(temp_x,temp_y+1,gameSquares[temp_x][temp_y+1].which_piece,1);

	gameSquares[temp_x][temp_y+1].printPiece(temp_x,temp_y+1,11,gameSquares[temp_
x][temp_y+1].which_piece);
                                }
                                else {
                                        pinkHighlightStep(temp_x,temp_y,BLANK);

	gameSquares[temp_x][temp_y+1].setSquare(temp_x,temp_y+1,1,1,BLANK,0);
                                        set_data_highlight(temp_x,temp_y+1,BLANK,1);
                                }
                        }
                        //Left-Upper Step
                        temp_x = cx; temp_y = cy;
```

```
                        if (temp_x <= 1 || temp_y >= 8) { }
                        else {
                                temp_x = cx-1; temp_y = cy;
                                if (gameSquares[temp_x][temp_y+1].is_empty==0 &&
gameSquares[temp_x][temp_y+1].kill_him==1) { }
                                else if (gameSquares[temp_x][temp_y+1].is_empty==0 &&
gameSquares[temp_x][temp_y+1].kill_him==0) {

        pinkHighlightStep(temp_x,temp_y,gameSquares[temp_x][temp_y+1].which_piece);

        gameSquares[temp_x][temp_y+1].setSquare(temp_x,temp_y+1,0,1,gameSquares[temp
_x][temp_y+1].which_piece,gameSquares[temp_x][temp_y+1].kill_him);

        set_data_highlight(temp_x,temp_y+1,gameSquares[temp_x][temp_y+1].which_piece,1);

        gameSquares[temp_x][temp_y+1].printPiece(temp_x,temp_y+1,11,gameSquares[temp_
x][temp_y+1].which_piece);
                                }
                                else {
                                        pinkHighlightStep(temp_x,temp_y,BLANK);

        gameSquares[temp_x][temp_y+1].setSquare(temp_x,temp_y+1,1,1,BLANK,0);
                                        set_data_highlight(temp_x,temp_y+1,BLANK,1);
                                }
                        }
                        //Right-Lower Step
                        temp_x = cx; temp_y = cy;
                        if (temp_x >= 8 || temp_y <= 1) { }
                        else {
                                temp_x = cx+1; temp_y = cy-2;
                                if (gameSquares[temp_x][temp_y+1].is_empty==0 &&
gameSquares[temp_x][temp_y+1].kill_him==1) { }
                                else if (gameSquares[temp_x][temp_y+1].is_empty==0 &&
gameSquares[temp_x][temp_y+1].kill_him==0) {

        pinkHighlightStep(temp_x,temp_y,gameSquares[temp_x][temp_y+1].which_piece);

        gameSquares[temp_x][temp_y+1].setSquare(temp_x,temp_y+1,0,1,gameSquares[temp
_x][temp_y+1].which_piece,gameSquares[temp_x][temp_y+1].kill_him);

        set_data_highlight(temp_x,temp_y+1,gameSquares[temp_x][temp_y+1].which_piece,1);

        gameSquares[temp_x][temp_y+1].printPiece(temp_x,temp_y+1,11,gameSquares[temp_
x][temp_y+1].which_piece);
```

```
                        }
                        else {
                                pinkHighlightStep(temp_x,temp_y,BLANK);

        gameSquares[temp_x][temp_y+1].setSquare(temp_x,temp_y+1,1,1,BLANK,0);
                                set_data_highlight(temp_x,temp_y+1,BLANK,1);
                        }
                }
                //Left-Lower Step
                temp_x = cx; temp_y = cy;
                if (temp_x <= 1 || temp_y <= 1) { }
                else {
                        temp_x = cx-1; temp_y = cy-2;
                        if (gameSquares[temp_x][temp_y+1].is_empty==0 &&
gameSquares[temp_x][temp_y+1].kill_him==1) { }
                        else if (gameSquares[temp_x][temp_y+1].is_empty==0 &&
gameSquares[temp_x][temp_y+1].kill_him==0) {

        pinkHighlightStep(temp_x,temp_y,gameSquares[temp_x][temp_y+1].which_piece);

        gameSquares[temp_x][temp_y+1].setSquare(temp_x,temp_y+1,0,1,gameSquares[temp
_x][temp_y+1].which_piece,gameSquares[temp_x][temp_y+1].kill_him);

        set_data_highlight(temp_x,temp_y+1,gameSquares[temp_x][temp_y+1].which_piece,1);

        gameSquares[temp_x][temp_y+1].printPiece(temp_x,temp_y+1,11,gameSquares[temp_
x][temp_y+1].which_piece);
                        }
                        else {
                                pinkHighlightStep(temp_x,temp_y,BLANK);

        gameSquares[temp_x][temp_y+1].setSquare(temp_x,temp_y+1,1,1,BLANK,0);
                                set_data_highlight(temp_x,temp_y+1,BLANK,1);
                        }
                }
                //Right Step
                temp_x = cx; temp_y = cy;
                if ( temp_x >= 8 ) { }
                else {
                        temp_x = cx+1; temp_y = cy-1;
                        if (gameSquares[temp_x][temp_y+1].is_empty==0 &&
gameSquares[temp_x][temp_y+1].kill_him==1) { }
                        else if (gameSquares[temp_x][temp_y+1].is_empty==0 &&
gameSquares[temp_x][temp_y+1].kill_him==0) {
```

```
	pinkHighlightStep(temp_x,temp_y,gameSquares[temp_x][temp_y+1].which_piece);

	gameSquares[temp_x][temp_y+1].setSquare(temp_x,temp_y+1,0,1,gameSquares[temp
_x][temp_y+1].which_piece,gameSquares[temp_x][temp_y+1].kill_him);

	set_data_highlight(temp_x,temp_y+1,gameSquares[temp_x][temp_y+1].which_piece,1);

	gameSquares[temp_x][temp_y+1].printPiece(temp_x,temp_y+1,11,gameSquares[temp_
x][temp_y+1].which_piece);
				}
				else {
					pinkHighlightStep(temp_x,temp_y,BLANK);

	gameSquares[temp_x][temp_y+1].setSquare(temp_x,temp_y+1,1,1,BLANK,0);
					set_data_highlight(temp_x,temp_y+1,BLANK,1);
				}
			}
			//Left Step
			temp_x = cx; temp_y = cy;
			if ( temp_x <= 1 ) { }
			else {
				temp_x = cx-1; temp_y = cy-1;
				if (gameSquares[temp_x][temp_y+1].is_empty==0 &&
gameSquares[temp_x][temp_y+1].kill_him==1) { }
				else if (gameSquares[temp_x][temp_y+1].is_empty==0 &&
gameSquares[temp_x][temp_y+1].kill_him==0) {

	pinkHighlightStep(temp_x,temp_y,gameSquares[temp_x][temp_y+1].which_piece);

	gameSquares[temp_x][temp_y+1].setSquare(temp_x,temp_y+1,0,1,gameSquares[temp
_x][temp_y+1].which_piece,gameSquares[temp_x][temp_y+1].kill_him);

	set_data_highlight(temp_x,temp_y+1,gameSquares[temp_x][temp_y+1].which_piece,1);

	gameSquares[temp_x][temp_y+1].printPiece(temp_x,temp_y+1,11,gameSquares[temp_
x][temp_y+1].which_piece);
				}
				else {
					pinkHighlightStep(temp_x,temp_y,BLANK);

	gameSquares[temp_x][temp_y+1].setSquare(temp_x,temp_y+1,1,1,BLANK,0);
					set_data_highlight(temp_x,temp_y+1,BLANK,1);
				}
```

```
                        }
                        //Up Step
                        temp_x = cx; temp_y = cy;
                        if ( temp_y >= 8 ) { }
                        else {
                                temp_x = cx; temp_y = cy;
                                if (gameSquares[temp_x][temp_y+1].is_empty==0 &&
gameSquares[temp_x][temp_y+1].kill_him==1) { }
                                else if (gameSquares[temp_x][temp_y+1].is_empty==0 &&
gameSquares[temp_x][temp_y+1].kill_him==0) {

        pinkHighlightStep(temp_x,temp_y,gameSquares[temp_x][temp_y+1].which_piece);

        gameSquares[temp_x][temp_y+1].setSquare(temp_x,temp_y+1,0,1,gameSquares[temp
_x][temp_y+1].which_piece,gameSquares[temp_x][temp_y+1].kill_him);

        set_data_highlight(temp_x,temp_y+1,gameSquares[temp_x][temp_y+1].which_piece,1);

        gameSquares[temp_x][temp_y+1].printPiece(temp_x,temp_y+1,11,gameSquares[temp_
x][temp_y+1].which_piece);
                                }
                                else {
                                        pinkHighlightStep(temp_x,temp_y,BLANK);

        gameSquares[temp_x][temp_y+1].setSquare(temp_x,temp_y+1,1,1,BLANK,0);
                                        set_data_highlight(temp_x,temp_y+1,BLANK,1);
                                }
                        }
                        //Down Step
                        temp_x = cx; temp_y = cy;
                        if ( temp_y <= 1 ) { }
                        else {
                                temp_x = cx; temp_y = cy-2;
                                if (gameSquares[temp_x][temp_y+1].is_empty==0 &&
gameSquares[temp_x][temp_y+1].kill_him==1) { }
                                else if (gameSquares[temp_x][temp_y+1].is_empty==0 &&
gameSquares[temp_x][temp_y+1].kill_him==0) {

        pinkHighlightStep(temp_x,temp_y,gameSquares[temp_x][temp_y+1].which_piece);

        gameSquares[temp_x][temp_y+1].setSquare(temp_x,temp_y+1,0,1,gameSquares[temp
_x][temp_y+1].which_piece,gameSquares[temp_x][temp_y+1].kill_him);

        set_data_highlight(temp_x,temp_y+1,gameSquares[temp_x][temp_y+1].which_piece,1);
```

```
		gameSquares[temp_x][temp_y+1].printPiece(temp_x,temp_y+1,11,gameSquares[temp_
x][temp_y+1].which_piece);
					}
					else {
						pinkHighlightStep(temp_x,temp_y,BLANK);

		gameSquares[temp_x][temp_y+1].setSquare(temp_x,temp_y+1,1,1,BLANK,0);
						set_data_highlight(temp_x,temp_y+1,BLANK,1);
					}
				}
			}
		}
}
//Tell About the Check of King
void tell_about_check(int cx, int cy) {
		//For Pawn
		if (gameSquares[cx][cy].which_piece == PAWN) {
			if (gameSquares[cx][cy].kill_him == 0) {	//If it is Player's PAWN
				if (gameSquares[cx+1][cy+1].kill_him == 1 &&
gameSquares[cx+1][cy+1].which_piece == KING) {
					gotoxy(5,15);
					HANDLE hConsole;
					hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
					SetConsoleTextAttribute(hConsole, 207);		//RED Colour
Code = 207
					cout<<"Check";
				}
				if (gameSquares[cx-1][cy+1].kill_him == 1 && gameSquares[cx-
1][cy+1].which_piece == KING) {
					gotoxy(5,15);
					HANDLE hConsole;
					hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
					SetConsoleTextAttribute(hConsole, 207);		//RED Colour
Code = 207
					cout<<"Check";
				}
			}
			else {
				if (gameSquares[cx+1][cy-1].kill_him == 0 && gameSquares[cx+1][cy-
1].which_piece == KING) {
					gotoxy(5,15);
					HANDLE hConsole;
					hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
```

```
                                SetConsoleTextAttribute(hConsole, 207);         //RED Colour
Code = 207
                                cout<<"Check";
                        }
                        if (gameSquares[cx-1][cy-1].kill_him == 0 && gameSquares[cx+1][cy-
1].which_piece == KING) {
                                gotoxy(5,15);
                                HANDLE hConsole;
                                hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
                                SetConsoleTextAttribute(hConsole, 207);         //RED Colour
Code = 207
                                cout<<"Check";
                        }
                }
        }
        //For Knight
        if (gameSquares[cx][cy].which_piece == KNIGHT) {
                if (gameSquares[cx][cy].kill_him == 0) {
                        //(x+1,y+2)
                        if (gameSquares[cx+1][cy+2].kill_him == 1 &&
gameSquares[cx+1][cy+2].which_piece == KING) {
                                gotoxy(5,15);
                                HANDLE hConsole;
                                hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
                                SetConsoleTextAttribute(hConsole, 207);         //RED Colour
Code = 207
                                cout<<"Check";
                        }
                        //(x+1,y-2)
                        else if (gameSquares[cx+1][cy-2].kill_him == 1 &&
gameSquares[cx+1][cy-2].which_piece == KING) {
                                gotoxy(5,15);
                                HANDLE hConsole;
                                hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
                                SetConsoleTextAttribute(hConsole, 207);         //RED Colour
Code = 207
                                cout<<"Check";
                        }
                        //(x-1,y+2)
                        else if (gameSquares[cx-1][cy+2].kill_him == 1 && gameSquares[cx-
1][cy+2].which_piece == KING) {
                                gotoxy(5,15);
                                HANDLE hConsole;
                                hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
```

```
                                SetConsoleTextAttribute(hConsole, 207);         //RED Colour
Code = 207
                                cout<<"Check";
                        }
                        //(x-1,y-2)
                        else if (gameSquares[cx-1][cy-2].kill_him == 1 && gameSquares[cx-1][cy-
2].which_piece == KING) {
                                gotoxy(5,15);
                                HANDLE hConsole;
                                hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
                                SetConsoleTextAttribute(hConsole, 207);         //RED Colour
Code = 207
                                cout<<"Check";
                        }
                        //(x+2,y+1)
                        else if (gameSquares[cx+2][cy+1].kill_him == 1 &&
gameSquares[cx+2][cy+1].which_piece == KING) {
                                gotoxy(5,15);
                                HANDLE hConsole;
                                hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
                                SetConsoleTextAttribute(hConsole, 207);         //RED Colour
Code = 207
                                cout<<"Check";
                        }
                        //(x+2,y-1)
                        else if (gameSquares[cx+2][cy-1].kill_him == 1 &&
gameSquares[cx+2][cy-1].which_piece == KING) {
                                gotoxy(5,15);
                                HANDLE hConsole;
                                hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
                                SetConsoleTextAttribute(hConsole, 207);         //RED Colour
Code = 207
                                cout<<"Check";
                        }
                        //(x-2,y+1)
                        else if (gameSquares[cx-2][cy+1].kill_him == 1 && gameSquares[cx-
2][cy+1].which_piece == KING) {
                                gotoxy(5,15);
                                HANDLE hConsole;
                                hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
                                SetConsoleTextAttribute(hConsole, 207);         //RED Colour
Code = 207
                                cout<<"Check";
                        }
```

```
                        //(x-2,y-1)
                        else if (gameSquares[cx-2][cy-1].kill_him == 1 && gameSquares[cx-2][cy-
1].which_piece == KING) {
                                gotoxy(5,15);
                                HANDLE hConsole;
                                hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
                                SetConsoleTextAttribute(hConsole, 207);         //RED Colour
Code = 207
                                cout<<"Check";
                        }
                }
                else {
                        //(x+1,y+2)
                        if (gameSquares[cx+1][cy+2].kill_him == 0 &&
gameSquares[cx+1][cy+2].which_piece == KING) {
                                gotoxy(5,15);
                                HANDLE hConsole;
                                hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
                                SetConsoleTextAttribute(hConsole, 207);         //RED Colour
Code = 207
                                cout<<"Check";
                        }
                        //(x+1,y-2)
                        else if (gameSquares[cx+1][cy-2].kill_him == 0 &&
gameSquares[cx+1][cy-2].which_piece == KING) {
                                gotoxy(5,15);
                                HANDLE hConsole;
                                hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
                                SetConsoleTextAttribute(hConsole, 207);         //RED Colour
Code = 207
                                cout<<"Check";
                        }
                        //(x-1,y+2)
                        else if (gameSquares[cx-1][cy+2].kill_him == 0 && gameSquares[cx-
1][cy+2].which_piece == KING) {
                                gotoxy(5,15);
                                HANDLE hConsole;
                                hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
                                SetConsoleTextAttribute(hConsole, 207);         //RED Colour
Code = 207
                                cout<<"Check";
                        }
                        //(x-1,y-2)
```

```
                        else if (gameSquares[cx-1][cy-2].kill_him == 0 && gameSquares[cx-1][cy-
2].which_piece == KING) {
                                gotoxy(5,15);
                                HANDLE hConsole;
                                hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
                                SetConsoleTextAttribute(hConsole, 207);         //RED Colour
Code = 207
                                cout<<"Check";
                        }
                        //(x+2,y+1)
                        else if (gameSquares[cx+2][cy+1].kill_him == 0 &&
gameSquares[cx+2][cy+1].which_piece == KING) {
                                gotoxy(5,15);
                                HANDLE hConsole;
                                hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
                                SetConsoleTextAttribute(hConsole, 207);         //RED Colour
Code = 207
                                cout<<"Check";
                        }
                        //(x+2,y-1)
                        else if (gameSquares[cx+2][cy-1].kill_him == 0 &&
gameSquares[cx+2][cy-1].which_piece == KING) {
                                gotoxy(5,15);
                                HANDLE hConsole;
                                hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
                                SetConsoleTextAttribute(hConsole, 207);         //RED Colour
Code = 207
                                cout<<"Check";
                        }
                        //(x-2,y+1)
                        else if (gameSquares[cx-2][cy+1].kill_him == 0 && gameSquares[cx-
2][cy+1].which_piece == KING) {
                                gotoxy(5,15);
                                HANDLE hConsole;
                                hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
                                SetConsoleTextAttribute(hConsole, 207);         //RED Colour
Code = 207
                                cout<<"Check";
                        }
                        //(x-2,y-1)
                        else if (gameSquares[cx-2][cy-1].kill_him == 0 && gameSquares[cx-2][cy-
1].which_piece == KING) {
                                gotoxy(5,15);
                                HANDLE hConsole;
```

```
                                hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
                                SetConsoleTextAttribute(hConsole, 207);         //RED Colour
Code = 207
                                cout<<"Check";
                        }
                }
        }
        //For Bishop
        if (gameSquares[cx][cy].which_piece == BISHOP) {
                if (gameSquares[cx][cy].kill_him == 0) {
                        int temp_x, temp_y;
                        //1st Quadrant (+ve x , +ve y)
                        temp_x = cx; temp_y = cy;
                        if (temp_x >= 8 || temp_y >= 8) { }
                        else {
                                for (int i=cx+1, j=cy; /*<No Condition>*/ ; i++, j++) {
                                        if (i > 8 || j >= 8) { break; }
                                        if (gameSquares[i][j+1].is_empty==0 &&
gameSquares[i][j+1].kill_him==0) { break; }
                                        if (gameSquares[i][j+1].is_empty==0 &&
gameSquares[i][j+1].kill_him==1) { break; }
                                        if (gameSquares[i][j+1].is_empty==KING &&
gameSquares[i][j+1].kill_him==1) {
                                                gotoxy(5,15);
                                                HANDLE hConsole;
                                                hConsole =
GetStdHandle(STD_OUTPUT_HANDLE);
                                                SetConsoleTextAttribute(hConsole, 207);
//RED Colour Code = 207
                                                cout<<"Check";
                                                break;
                                        }
                                }
                        }
                        //2nd Quadrant (x -ve , y +ve)
                        if (temp_x < 1 || temp_y >= 8) { }
                        else {
                                for (int i=cx-1, j=cy; /*<No Condition>*/ ; i--, j++) {
                                        if (i < 1 || j > 7) { break; }
                                        if (gameSquares[i][j+1].is_empty==0 &&
gameSquares[i][j+1].kill_him==0) { break; }
                                        if (gameSquares[i][j+1].is_empty==0 &&
gameSquares[i][j+1].kill_him==1) { break; }
```

```
                                        if (gameSquares[i][j+1].is_empty==KING &&
gameSquares[i][j+1].kill_him==1) {
                                                gotoxy(5,15);
                                                HANDLE hConsole;
                                                hConsole =
GetStdHandle(STD_OUTPUT_HANDLE);
                                                SetConsoleTextAttribute(hConsole, 207);
//RED Colour Code = 207
                                                cout<<"Check";
                                                break;
                                        }
                                }
                        }
                        //3rd Quadrant (x -ve , y -ve)
                        if (temp_x < 1 || temp_y < 1) { }
                        else {
                                for (int i=cx-1, j=cy-2; /*<No Condition>*/ ; i--, j--) {
                                        if (i <= 0 || j < 0) { break; }
                                        if (gameSquares[i][j+1].is_empty==0 &&
gameSquares[i][j+1].kill_him==0) { break; }
                                        if (gameSquares[i][j+1].is_empty==0 &&
gameSquares[i][j+1].kill_him==1) { break; }
                                        if (gameSquares[i][j+1].is_empty==KING &&
gameSquares[i][j+1].kill_him==1) {
                                                gotoxy(5,15);
                                                HANDLE hConsole;
                                                hConsole =
GetStdHandle(STD_OUTPUT_HANDLE);
                                                SetConsoleTextAttribute(hConsole, 207);
//RED Colour Code = 207
                                                cout<<"Check";
                                                break;
                                        }
                                }
                        }
                        //4th Quadrant (x +ve , y -ve)
                        if (temp_x >= 8 || temp_y < 1) { }
                        else {
                                for (int i=cx+1, j=cy-2; /*<No Condition>*/ ; i++, j--) {
                                        if (i > 8 || j < 0) { break; }
                                        if (gameSquares[i][j+1].is_empty==0 &&
gameSquares[i][j+1].kill_him==0) { break; }
                                        if (gameSquares[i][j+1].is_empty==0 &&
gameSquares[i][j+1].kill_him==1) { break; }
```

```
                                        if (gameSquares[i][j+1].is_empty==KING &&
gameSquares[i][j+1].kill_him==1) {
                                                gotoxy(5,15);
                                                HANDLE hConsole;
                                                hConsole =
GetStdHandle(STD_OUTPUT_HANDLE);
                                                SetConsoleTextAttribute(hConsole, 207);
//RED Colour Code = 207
                                                cout<<"Check";
                                                break;
                                        }
                                }
                        }
                }
        }
}

//Unhighlighted Pink Step_____________________
void CPiece::unhighlightStep(int cx, int cy){
   HANDLE hConsole;
   hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
   if ( cx%2==0 && cy%2==0 ) {          //If x is even, y is evem
      HANDLE hConsole;                  //Print Gray Step
      hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
      SetConsoleTextAttribute(hConsole, 8);         //Gray Colour Code = 8
      int X = (cx * 5)+16-3;
      int Y = 24-(cy * 3)+1;
      X = X-2;
      Y = Y-1;
      gotoxy(X,Y);
      for (int i=0; i<3; i++) {
         for (int i=0; i<5; i++) {
            cout<<"\xDB";
         }
         Y++;
         gotoxy(X,Y);
      }
   }
   else if ( cx%2!=0 && cy%2!=0 ) {      //If x is odd, y is odd
      HANDLE hConsole;                  //Print Gray Step
      hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
      SetConsoleTextAttribute(hConsole, 8);         //Gray Colour Code = 8
      int X = (cx * 5)+16-3;
      int Y = 24-(cy * 3)+1;
```

```
    X = X-2;
    Y = Y-1;
    gotoxy(X,Y);
    for (int i=0; i<3; i++) {
        for (int i=0; i<5; i++) {
            cout<<"\xDB";
        }
        Y++;
        gotoxy(X,Y);
    }
}
else if ( cx%2!=0 && cy%2==0 ) {       //If x is odd, y is even
    HANDLE hConsole;                   //Print Green Step
    hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
    SetConsoleTextAttribute(hConsole, 2);          //Greem Colour Code = 2
    int X = (cx * 5)+16-3;
    int Y = 24-(cy * 3)+1;
    X = X-2;
    Y = Y-1;
    gotoxy(X,Y);
    for (int i=0; i<3; i++) {
        for (int i=0; i<5; i++) {
            cout<<"\xB0";
        }
        Y++;
        gotoxy(X,Y);
    }
}
else if ( cx%2==0 && cy%2!=0 ) {       //If x is even, y is odd
    HANDLE hConsole;                   //Print Green Step
    hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
    SetConsoleTextAttribute(hConsole, 2);          //Green Colour Code = 2
    int X = (cx * 5)+16-3;
    int Y = 24-(cy * 3)+1;
    X = X-2;
    Y = Y-1;
    gotoxy(X,Y);
    for (int i=0; i<3; i++) {
        for (int i=0; i<5; i++) {
            cout<<"\xB0";
        }
        Y++;
        gotoxy(X,Y);
    }
```

```
    }
        ////////////////////////////

        if (gameSquares[cx][cy].is_empty == 1) {
            gameSquares[cx][cy].setSquare(cx,cy,1,0,0,0);
        }
        else {
            if (gameSquares[cx][cy].kill_him == 1){

        gameSquares[cx][cy].setSquare(cx,cy,0,0,gameSquares[cx][cy].which_piece,gameSqua
res[cx][cy].kill_him);

        gameSquares[cx][cy].printPiece(cx,cy,15,gameSquares[cx][cy].which_piece);
            }
            else {

        gameSquares[cx][cy].setSquare(cx,cy,0,0,gameSquares[cx][cy].which_piece,gameSqua
res[cx][cy].kill_him);

        gameSquares[cx][cy].printPiece(cx,cy,10,gameSquares[cx][cy].which_piece);
            }
        }
}
//Pink Highlight Step_____________________________
void CPiece::pinkHighlightStep(int cx, int cy, int pieceNum){
    HANDLE hConsole;
    hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
    if ( cx%2==0 && cy%2==0 ) {          //If x is even, y is evem
        HANDLE hConsole;                 //Print Gray Step
        hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
        SetConsoleTextAttribute(hConsole, 5);         //Dark Pink Colour Code = 5
        int X = (cx * 5)+16-3;
        int Y = 24-(cy * 3)+1;
        int tempX = X - 2;
        int tempY = Y - 4;
        gotoxy(tempX,tempY);
        for (int i=0; i<3; i++) {
            for (int i=0; i<5; i++) {
                cout<<"\xDB";
            }
            tempY++;
            gotoxy(tempX,tempY);
        }
    }
```

```
    else if ( cx%2!=0 && cy%2!=0 ) {       //If x is odd, y is odd
        HANDLE hConsole;                   //Print Gray Step
        hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
        SetConsoleTextAttribute(hConsole, 5);          //Dark Pink Colour Code = 5
        int X = (cx * 5)+16-3;
        int Y = 24-(cy * 3)+1;
        int tempX = X - 2;
        int tempY = Y - 4;
        gotoxy(tempX,tempY);
        for (int i=0; i<3; i++) {
            for (int i=0; i<5; i++) {
                cout<<"\xDB";
            }
            tempY++;
            gotoxy(tempX,tempY);
        }
    }
    else if ( cx%2!=0 && cy%2==0 ) {       //If x is odd, y is even
        HANDLE hConsole;                   //Print Gray Step
        hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
        SetConsoleTextAttribute(hConsole, 13);         //Light Pink Colour Code = 13
        int X = (cx * 5)+16-3;
        int Y = 24-(cy * 3)+1;
        int tempX = X - 2;
        int tempY = Y - 4;
        gotoxy(tempX,tempY);
        for (int i=0; i<3; i++) {
            for (int i=0; i<5; i++) {
                cout<<"\xDB";
            }
            tempY++;
            gotoxy(tempX,tempY);
        }
    }
    else if ( cx%2==0 && cy%2!=0 ) {       //If x is even, y is odd
        HANDLE hConsole;                   //Print Gray Step
        hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
        SetConsoleTextAttribute(hConsole, 13);         //Light Pink Colour Code = 13
        int X = (cx * 5)+16-3;
        int Y = 24-(cy * 3)+1;
        int tempX = X - 2;
        int tempY = Y - 4;
        gotoxy(tempX,tempY);
        for (int i=0; i<3; i++) {
```

```
            for (int i=0; i<5; i++) {
                cout<<"\xDB";
            }
            tempY++;
            gotoxy(tempX,tempY);
        }
    }
}
```

Time Complexity Analysis:

Function Name	No. of lines of code	Complexity
CChessBoard::CChessBoard	3	1
CChessBoard::greenStep	8	2
CChessBoard::grayStep	8	2
CChessBoard::printRowGreen	6	2
CChessBoard::printRowGray	6	2
CChessBoard::printBoard	18	3
CChessBoard::PrintX_Label	11	2
CChessBoard::PrintY_Label	11	2
CChessBoard::MoveTo	8	1
CChessBoard::Move	10	1
CChessBoard::MoveInt	9	1
CPiece::CPiece	8	1
CPiece::setSquare	8	1
CPiece::selectPiece	6	2
CPiece::selectNewPosition	6	2
CPiece::printBlank	38	9
CPiece::printPiece	26	7

CGamePlay::settingGameBoard	45	3
CGamePlay::removePiece	4	1
main	107	33
gotoxy	10	1
set_data_highlight	6	1
check_and_unhighlight	10	4
end_game_or_not	14	5
CPiece::highlightPiece	1259	526
tell_about_check	224	91
CPiece::unhighlightStep	85	19
CPiece::pinkHighlightStep	72	17

MODULE DESCRIPTION

- **Basic C++ inbuilt libraries.**

  Basic C++ modules likeconio, iostream etc. was imported for construction of the backbone of this program.

- **Rook module (User defined)**

  A separate rook module will be constructed to define movement of the piece along with the way it captures the opponent piece. The Rook's movement is divided into 4 parts constructed using a loop: First loop is increasing X coordinate with constant Y coordinate,Second loop is decreasing X coordinate with constant Y coordinate, Third loop is constant X coordinate with increasing Y coordinate and the Fourth loop is constant X coordinate with decreasing Y coordinate.

- **Knight module (User defined)**

  A separate Knight module will be constructed to define movement of the piece along with the way it captures the opponent piece. The Knight's movement is divided into 8

portions: where the knight moves (X-1, Y+2), (X-1, Y-2), (X+1, Y+2), (X+1, Y-2), (X-2, Y+1), (X-2,Y-1), (X+2, Y+1), (X+2, Y-1) on the X axis and the Y axis.

- **King module (User defined)**

  A separate King module will be constructed to define movement of the piece along with the way it captures the opponent piece. The King's movements are defined as 8 different steps: the Right-Upper step, the Left-Upper step, the Right-Lower step, the Left-Lower step, the Right step, the Left step, the Front step and the Back step.

- **Queen module (User defined)**

  A separate Queen module will be constructed to define movement of the piece along with the way it captures the opponent piece. The Queen's movements are a combination of the movements of the Rook and the Bishop and are therefore divided into 8 portions these include: diagonal movement in each of the four quadrants i.e. (+ve X, +ve Y), (-ve X, +ve Y), (-ve X, -ve Y), (+ve X, -ve Y) and linear movement using loops as First loop is increasing X coordinate with constant Ycoordinate, Second loop is decreasing X coordinate with constant Y coordinate, Thirdloop is constant X coordinate withincreasing Y coordinate and the Fourth loop isconstant X coordinate with decreasing Y coordinate.

- **Bishop module (User defined)**

  A separate Bishop module will be constructed to define movement of the piece along with the way it captures the opponent piece. The Bishop's movement in divided into 4 portions: diagonal movement in each of the four quadrants i.e. (+ve X, +ve Y), (-ve X, +ve Y), (-ve X, -ve Y), (+ve X, -ve Y).

- **Pawn module (User defined)**

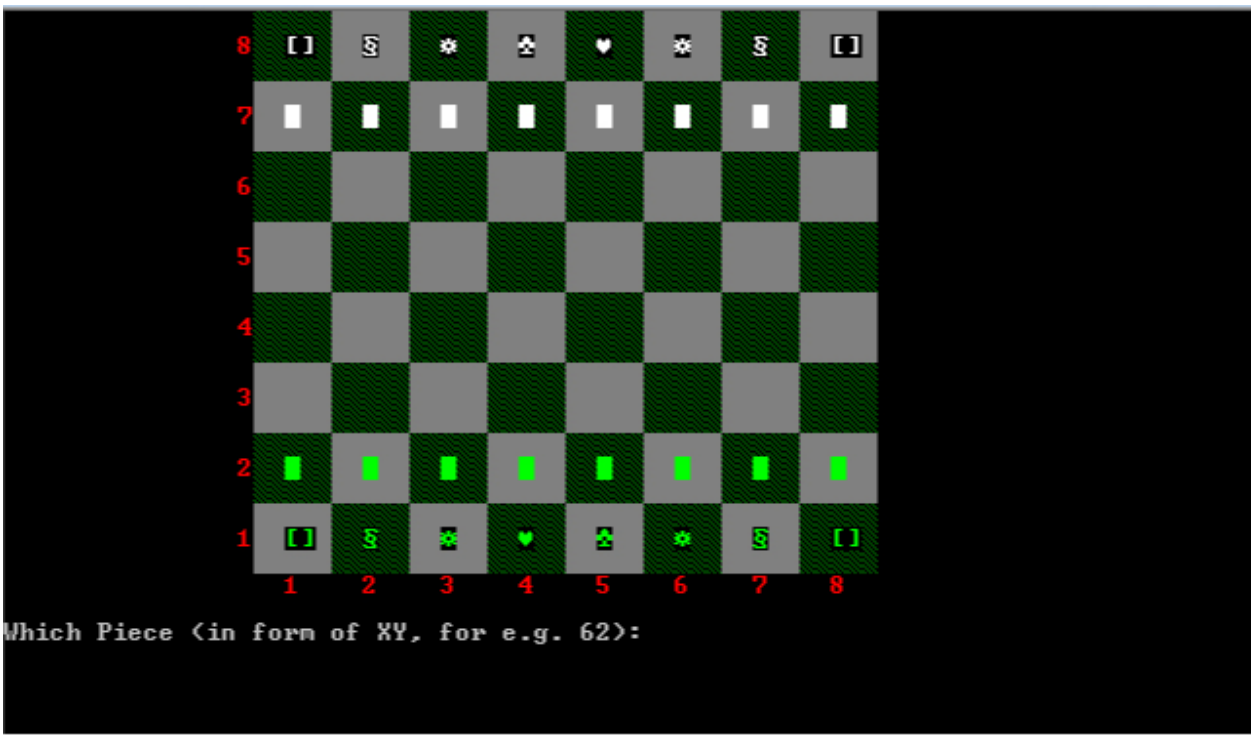
  A separate Pawn module will be constructed to define movement of the piece along with the way it captures the opponent piece. The Pawn's movement is divided into 3 main portions: one defines the movement of two steps forward in the first play only, the other defines the movement of a single step forward for all following moves, and the third defines the adjacent movement while capturing a chess piece only when an opponent chess piece is adjacent to it.

- **Graphics module**

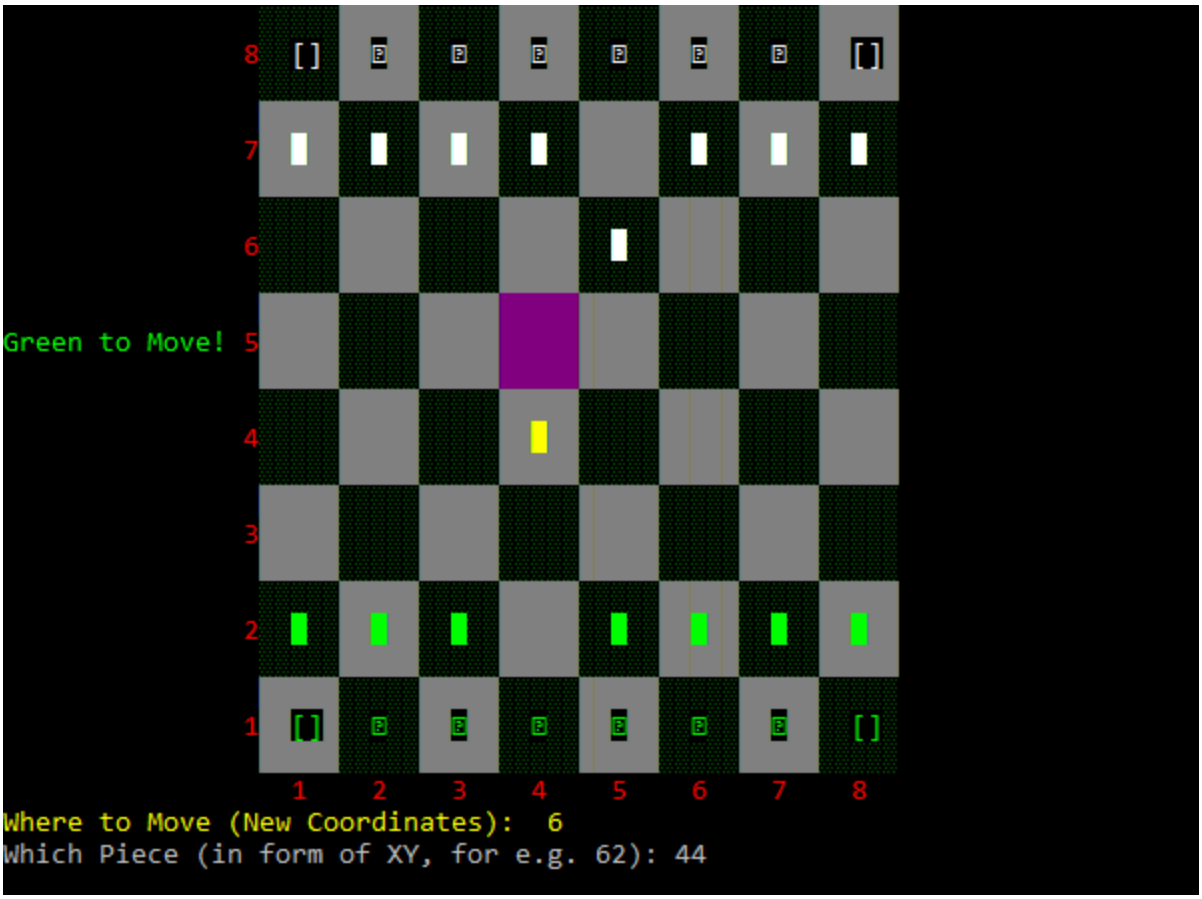
The graphics module will be imported to create a user-friendly interface. This is to help the player visualize the piece position and make accurate moves.

Results and Discussion

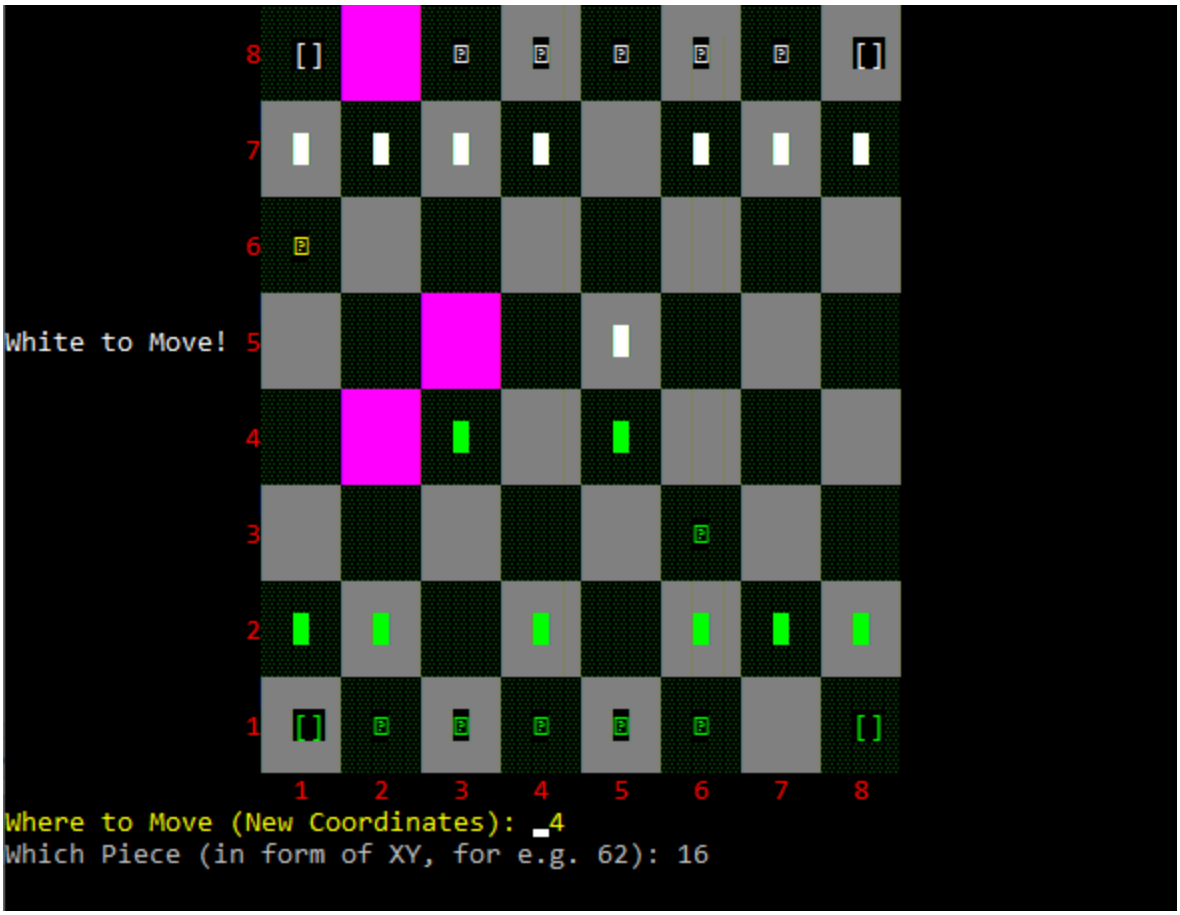
The Program simulates a game of chess between 2 players. During the game, the program is able to successfully predict all possible moves for a particular chess piece. It is able to do this using a 2-dimensional array data structure that represents the positions in a chess board. The settingGameBoard method of the CGameplay class sets the board to it's initial position.



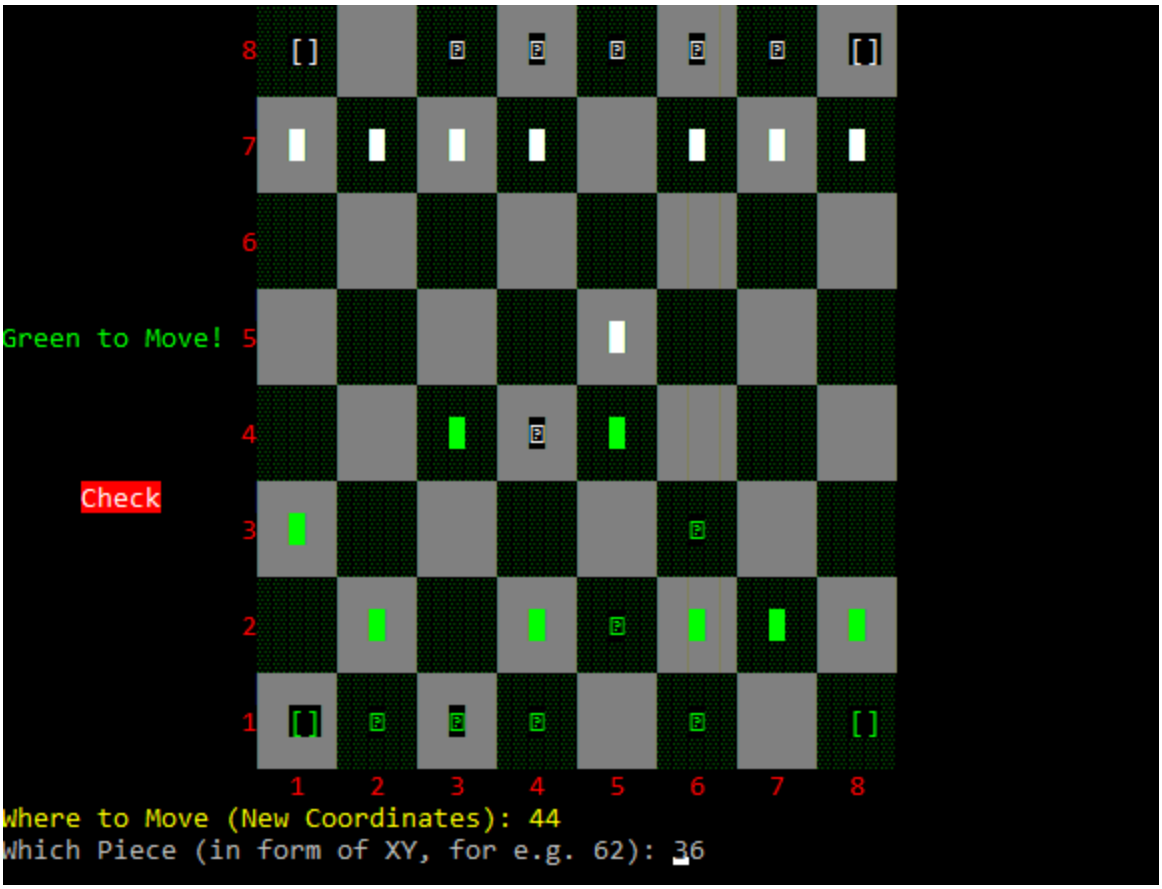
Each piece is highlighted on selection and so are the possible steps that the particular piece can take. It does this by setting the Boolean value is_highlighted of each of the eligible squares to True. The chess piece is then allowed to move only to one of the squares that has the is_highlighted set to True.



Depending upon the type of piece selected a different movement is set to it. This is set using the highlightPiece class which inherits from CPiece class. The CPiece class contains details about each piece such as it's current coordinates, whether or not it is highlighted, what type of piece it is and which team it is of. The CPiece class is also responsible for printing the corresponding icon for each piece type.



The function is also capable of informing about a check using the tell_about_check function. It does so by checking if the king piece of the opponent player lies on the path of any piece. In the situation of a check-mate the game is ended.



REFRENCES:

[1] Erik Bernhardsson.(2014) Deep learning for... chess.

[2] Christopher Clark and Amos Storkey.(2014) Teaching deep convolutional neural networks to play go. arXiv preprint arXiv:1412.3409

[4] H´ectorApolo Rosales Pulido(2016). Predicting the Outcome of a Chess Game by Statistical and Machine Learning techniques.

[5]BarakOshri and Nishith Khandwala for CS231N at Stanford University.(2016)Predicting Moves in Chess using Convolutional Neural Networks.

[6]Harsh Jhamtani, Varun Gangal, Eduard Hovy, Graham Neubig, Taylor Berg-Kirkpatrick.(2018)Learning to Generate Move-by-Move Commentary for Chess Games from

Large-Scale Social Forum DataP18-1154Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers) P18-1154
[7] Sam Wiseman, Stuart M Shieber, and Alexander M Rush. Challenges in Data-to-Document Generation.(2018)arXiv preprint arXiv:1707.08052 , 2018
[8] Kenneth W. Regan, University at Buffalo CSE. Human Decision Making (At Chess) – White Paper
[9]K. Regan.(2018) ``Rating Computer Science Via Chess," invited submission accepted in final form to the Springer LNCS 10,000 anniversary volume.
[10]IsaacKamlish, Isaac BentataChocron, Nicholas McCarthy.(2019)SentiMATE: Learning to play Chess through Natural Language Processing.
[11]CezaryDendek and JaeckMandzuik(2018). A neural network classifier of chess moves.
[12]MatthiaSabatelli,FrancescoBidoia,Valeriu Bogdan Codreanu,Marco A. Wiering(2018) Learning to evaluate chess positions using deep neural netwroks and limited lookahead.
[13]Diogo R. Ferreira(2012).Detremining the strength of chess players based on actual play
[14]Vale (2001), pp. 170–99
[15] "Fundamentals". *gap-system.org. Archived from* the original *on 7 June 2011.*