



CHESSICA: The Modern Game of Chess



A Computer Science Project

By Vineet Vinayak Pasupulety

CERTIFICATE

Name: Vineet Vinayak Pasupulety

Roll no. :

Class: XII A

Exam no. :

**This is certified to be the bonafide work of the student in the
Computer Laboratory during the academic year 2012-2013.**

Teacher In-charge

Examiners's Signature

Principal's Signature

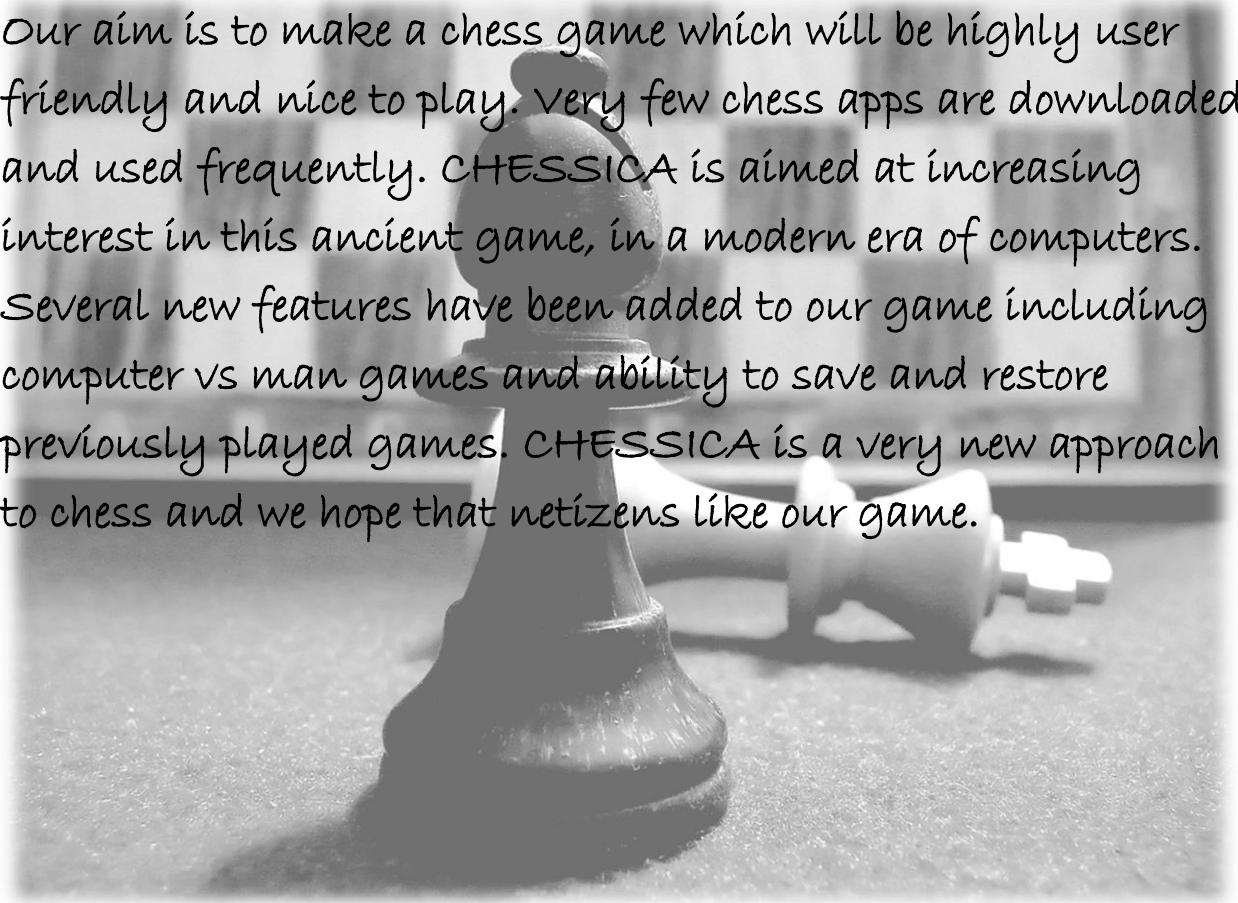
Date:

Institution's Rubber Stamp

INDEX

I am an avid chess player and frequently use chess apps to play chess. I got interested in how these chess apps work and how their logic works. So I started working on a code with my partner, Ajay, to make such a game. And our product, CHESSICA, was created.

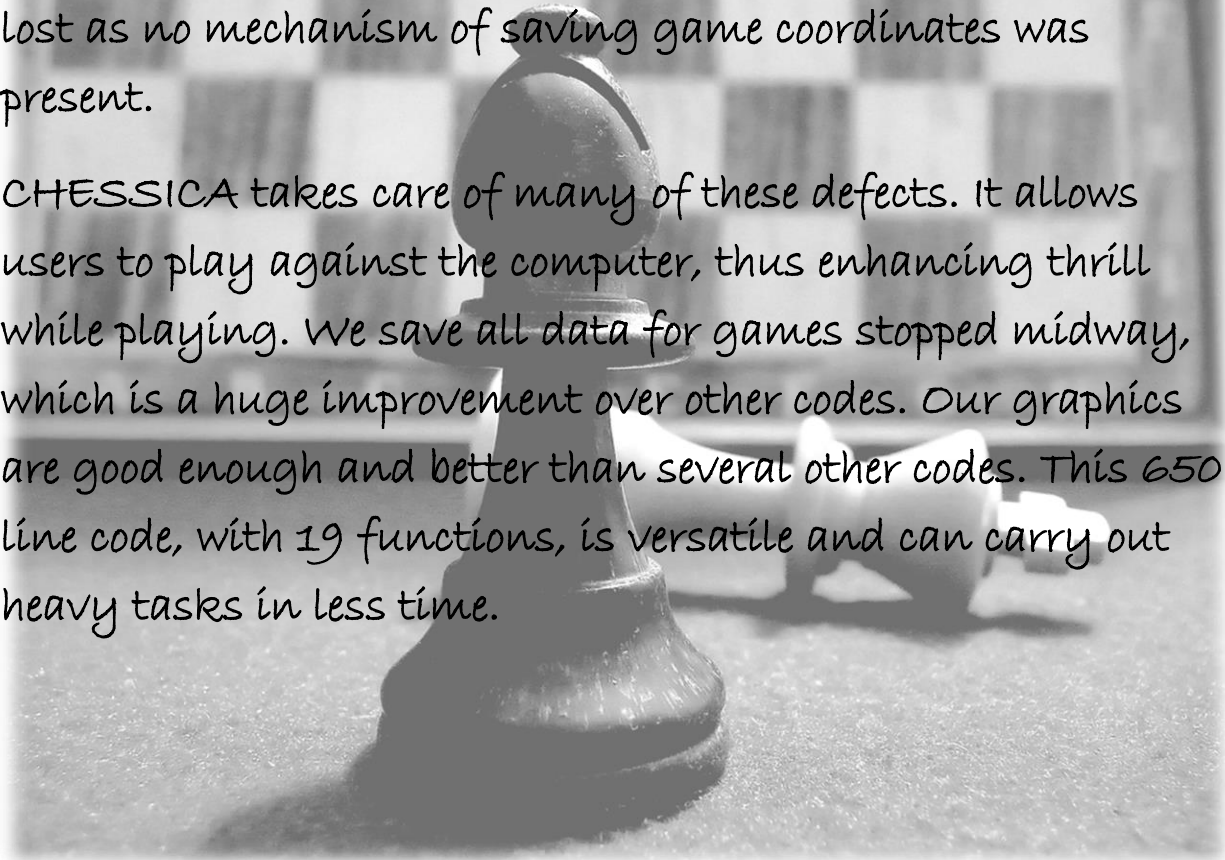
Our aim is to make a chess game which will be highly user friendly and nice to play. Very few chess apps are downloaded and used frequently. CHESSICA is aimed at increasing interest in this ancient game, in a modern era of computers. Several new features have been added to our game including computer vs man games and ability to save and restore previously played games. CHESSICA is a very new approach to chess and we hope that netizens like our game.



PROBLEM DEFINITION

When we tried to play chess on a couple of internet apps, we found that several apps didn't follow the basic rules of chess. Many of them had very bad graphics and showed major flaws in their coding. A lot more apps required payment and had restricted access. Many other apps allowed only man vs man games, which made it very uncomfortable as one computer was to be shared between 2 players, what was bad was that, in most apps, exciting games which were stopped midway, were lost as no mechanism of saving game coordinates was present.

CHESSICA takes care of many of these defects. It allows users to play against the computer, thus enhancing thrill while playing. We save all data for games stopped midway, which is a huge improvement over other codes. Our graphics are good enough and better than several other codes. This 650 line code, with 19 functions, is versatile and can carry out heavy tasks in less time.



ANALYSIS

1. INPUTS

The most important inputs that have to be given to the program by users are as follows:

- + The type of game he/she would like to play (Computer vs man/man vs man)

During each move in any particular game, the necessary inputs would be:

- + Initial coordinates of the coin he wishes to move
- + Final coordinates of the same coin

2. OUTPUTS

The following outputs are given by the compiler/program:

- + The required type of chess game (computer vs man/man vs man)
- + A chess board with all coins in positions allotted by the user. With every move, the chess board updates locations of all coins and displays the board
- + The program allows you to save games and quit them at any time
- + In case of errors by the user, the program informs the user of the mistake and allows him to rectify it.

PROJECT DESIGN

+ Data abstraction

The user is simply allowed to play the game; he/she isn't interrupted with messages and commands being sent in the background. The complex coding doesn't affect his/her speed or concentration.

+ Data encapsulation

The code has been made to ensure that memory and time is not wasted. Functions and objects are kept together for easy execution. But hiding of all unnecessary information and procedures is always ensured.

+ Modularity

The entire code has been broken down into cohesive modules which provide for code efficiency and redundancy at all levels.

+ Classes and Objects

Class COIN is the only class used which gives the definition of every coin on the board. 32 objects of class COIN are made to replicate the 32 coins on a chess board. Initialization of coin details takes place soon

after and these coins access only specific functions based on these parameters.

Function Overloading/Polymorphism

Function overloading is crucial in our game as all coins have similar base definitions and memory wastage doesn't take place as the compiler doesn't have to repeatedly call different functions. The logic functions for each coin are all overloaded.

Data File Handling

A datafile 'FILE' is used to store the location, colour, name, move and life status for all 32 coins. The file is regularly updated with every move.



CODE

```
#include<iostream.h>    // cout / cin
#include<conio.h>        // getch
#include <cstdlib>        // text color
#include<fstream.h>      //datafile handling
#include<math.h>         // absolute value
#include<process.h>      // exit
#include<iomanip.h>      //screen editing
#include<windows.h>      //system functions

void color(int k);           // Provides colour
void initializer();          //  Initializes the coins on the board
int retrieve();              //  Retrieve data from file
int winner();                //  find winner
int welcome();               //  the user interface
int objno(int a,int b);      //  returns object number
int self(int a, int b, char col); //check if object killing its own members
int checker(int xi,int yi, char col); //  checks if object exists
int input(int a);            //  receive input
void graph();                //  output chessboard
void save(int x);            //  save data
int retrand(); // returns a random number within the specification
void process();              //  acts on the user input
int logicq(int xi,int yi,int xf,int yf,char col); // logic of queen
```



```

int logicr(int xi,int yi,int xf,int yf,char col); // logic of rook
int logic_(int xi,int yi,int xf,int yf,char col); // logic of king
int logick(int xi, int yi, int xf, int yf,char col); // logic of knight
int logicb(int xi, int yi, int xf, int yf,char col); // logic of bishop
int logicp(int xi, int yi, int xf, int yf,char col); // logicof pawn
int input(intk,charai); // comp vs man - function overloading
using namespace std;
int ch,xf,yf,xi,yi,yrn,z,a;
char wit,col; // global variables

int retrand() // returns random numbers within the prescribed limit
{
    for(;;)
    {
        a=rand()%8;
        if(a>0)
            return a;
    }
return 5;
}

class coin{
    char name,lrd,color;
    int x,y,move; // data members

```

public:

int retx() // return x coordinate

{return x; }

int rety() // return y coordinate

{return y;}

char retl() // return the life / death condition

{return lrd;}

char retn() // return name

{return name;}

char retc() // return color

{return color;}

int retm() // return number of moves

{return move;}

**void ret(char q, char w,char e,int r,int t,int m) // receive /
//assign values to**

{

name=q;

lrd=w;

color=e;

x=r;

y=t;

move=m;

}

```
void in(char p) // edit name
```

```
{name=p;}
```

```
void il(char p) // edit living or death condition
```

```
{lrd=p;}
```

```
void ix(int a) //edit x coordinate
```

```
{x=a;}
```

```
void iy(int b) // edit y coordinate
```

```
{y=b;}
```

```
void ic(char p) // edit color
```

```
{color=p;}
```

```
void im(int y) // edit move
```

```
{move=y;}
```

```
}c[32];
```

```
void color(int k) // change color of the text
```

```
{
```

```
    HANDLE hConsole;
```

```
hConsole = GetStdHandle(STD_OUTPUT_HANDLE);
```

```
SetConsoleTextAttribute(hConsole, k);
```

```
}
```

```
void time()
```

```
{for(long double i=0;i<1200000000;++i)
```

```
    { }          // empty code for delays
}
```

```
void initializer() // Initialize all coins on the board
```

```
{
for(int i=0;i<8;++i)
{
    c[i].in('P');
    c[i].iy(2);
    c[i].ic('W');
    c[i].ix(i+1);
    c[i].il('L');
    c[i].im(0);
} //Initialize 8 white soldiers
for(int j=16;j<24;++j)
{
    c[j].in('P');
    c[j].iy(7);
    c[j].ix(j-15);
    c[j].ic('B');
    c[j].il('L');
    c[j].im(0);
} // Initialize 8 black soldiers
{ c[8].in('R');
```

```
c[8].im(0);
c[8].iy(1);
c[8].ic('W');
c[8].ix(1);
c[8].il('L');
c[15]=c[8];
c[15].ix(8);
c[24]=c[31]=c[8];
c[24].ic('B');
c[31].ic('B');
c[24].iy(8);
c[31].iy(8);
c[24].ix(1);
c[31].ix(8);
} // Initialize all the ROOKs
```

```
{ c[9].in('K');
c[9].im(0);
c[9].iy(1);
c[9].ic('W');
c[9].ix(2);
c[9].il('L');
c[14]=c[9];
```

```
c[14].ix(7);  
c[25]=c[30]=c[9];  
c[25].ic('B');  
c[30].ic('B');  
c[30].ix(7);  
c[30].iy(8);  
c[25].iy(8);  
} // Initialize all the Knights
```

```
{ c[10].in('B');  
c[10].im(0);  
c[10].iy(1);  
c[10].ic('W');  
c[10].ix(3);  
c[10].il('L');  
c[13]=c[10];  
c[13].ix(6);  
c[26]=c[29]=c[10];  
c[26].ic('B');  
c[29].ic('B');  
c[29].ix(6);  
c[29].iy(8);  
c[26].iy(8);
```

```
} // Initialize all the Bishops
```

```
{c[11].in('Q');
```

```
c[11].im(0);
```

```
c[11].ic('W');
```

```
c[11].ix(4);
```

```
c[11].iy(1);
```

```
c[11].il('L');
```

```
c[27]=c[11];
```

```
c[27].iy(8);
```

```
c[27].ic('B');
```

```
} // initialize both the queens
```

```
{c[12].in('*');
```

```
c[12].im(0);
```

```
c[12].ic('W');
```

```
c[12].ix(5);
```

```
c[12].iy(1);
```

```
c[12].il('L');
```

```
c[28]=c[12];
```

```
c[28].ic('B');
```

```
c[28].iy(8);
```

```
} // initialize both the kings
```

```
}//close the function
```

```
int retrieve() // retrieve data from text file and copy to objects
```

```
{
```

```
ifstream coin; //creation of object
```

```
coin.open("file.txt"); //opening of file.txt
```

```
char q,w,e;
```

```
int r,t,y; //declaration of variables
```

```
for(int j=0;j<32;++j)
```

```
{
```

```
coin>>q>>w>>e>>r>>t>>y; //load data into variables
```

```
c[j].ret(q,w,e,r,t,y); //initialize all the objects
```

```
}
```

```
coin>>r;
```

```
return r;
```

```
}
```

```
void process() // deal with user input
```

```
{
```

```
int k=welcome(); //calling welcome()
```

```
system("cls");
```

```
system("color 1A");
```



```
cout<<"\n\n\tPress \n\n\t(1) Start new game \n\n\t(2) Restore  
Previous game\n\n\t\t: ";
```

```
cin>>ch;
```

```
if(ch==2)
```

```
{
```

```
    ch=retrieve(); // starts from previous user
```

```
    system("cls");
```

```
    cout<<"\n\n\n\n\n\t\t\tRetrieving ...";
```

```
    time(); // create time lag
```

```
} // retrieving if user wants to open saved game
```

```
else
```

```
{
```

```
    initializer();
```

```
    ch=0;
```

```
} // create identity to the coins
```

```
if(k==1) // man on man
```

```
for(;winner()!=1;++ch)
```

```
{
```

```
    graph(); // display chessboard
```

```
    input(ch);
```

```
    save(ch); // save
```

```
}
```

```
else //comp on man
```

```
for(;winner()!=1;++ch) // play until somebody wins
```

```
{
```

```
graph();
```

```
input(ch,'a');
```

```
save(ch);
```

```
}
```

```
}//closing process()
```

```
int winner() // check if somebody has won
```

```
{ system("cls");
```

```
if(c[12].retl()=='D') // check if kings are living
```

```
{
```

```
system("color 30");
```

```
cout<<"\n\n\n\n\n\n\n\n\n\t\t\tBlack is the winner ";
```

```
getch();
```

```
return 1;
```

```
}
```

```
else if(c[28].retl()=='D')
```

```
{
```

```
system("color 30");
```

```
cout<<"\n\n\n\n\n\n\n\n\n\t\t\tWhite is the winner ";// display winner
```

```
getch();
```

```
return 1;    }
```

```
else return 0;
```

```
}
```

```
int welcome()    //user interface
```

```
{
```

```
system("color 1A"); // background blue text green
```

```
cout<<"\n\n\n\n\n\t\t\tWelcome to CHESSICA \n\n\n\n\n\n\t\t\t";
```

```
time(); //delay
```

```
system("cls");
```

```
opt: system("color A1"); //makes background green and color blue
```

```
cout<<"\n\n\t\t\tPlease press \n\n\n\n \t(1) Start game \n \n\t(2)
```

```
Instructions and Tips \n\n\n\t(3) Credits \n\n\n\t(4) Quit \n\n\n\t Choice : ";
```

```
cin>>ch;
```

```
switch(ch)
```

```
{
```

```
case 1: system("cls");
```

```
        system("color 2c"); // background blue text green
```

```
        cout<<"\n\n\n\t\t\tPlease press \n\n\n\n \t(1) Man-Man \n\n\n \t(2)  
        Computer-Man \n\n\n\t Choice : ";
```

```
        cin>>ch; // input choice
```

```
        return ch; // return choice
```

```
case 2: system("color E0");
```

```
        system("cls");
```

```
cout<<"\n(A) Move object by entering initial and final  
coordinates\n\t((column <space> row) format)\n\tex. 5 3\n\n";
```

```
cout<<"(B) Press 9 9 to save and exit "<<endl<<"\n(C) Press 0 0  
WHILE entering final coordinates to re-enter\n\t\tinitial  
coordinates";
```

```
cout<<"\n\n(D) To forfeit turn press 11 11 \n\n(E) The game is  
saved automatically after every turn\n\n(F) When a game is  
being restored it starts from the user who had entered 9 9";
```

```
getch();
```

```
system("cls");
```

```
goto opt;
```

```
case 3: ShellExecute(NULL, "open",  
"http://chesscredits.blogspot.com",NULL, NULL, SW_SHOWNORMAL);
```

```
system("cls");
```

```
goto opt;
```

```
default : exit(0);
```

```
}
```

```
}
```

```
int objno(int a,int b) //return object number given the coordinates
```

```
{
```

```
for(int i=0;i<32;++i)
```

```
{
```

```
if(c[i].retx() == a && c[i].rety()==b && c[i].retl()=='L')
```

```
return i;
```

```

    }
    return -1; // if object doesnt exist return -1
}

int logicq(int xi, int yi, int xf, int yf, char col) // logic queen
{
    int k = logicb(xi, yi, xf, yf, col); // logic of bishop
    int l = logicr(xi, yi, xf, yf, col); // logic of rook
    int m = logicp(xi, yi, xf, yf, col); // logic of pawn
    if(l==1 || m==1 || k==1) // queens logic is either bishop /rook /pawn
        return 1;
    else return 0;
}

```

```

int logicb(int xi, int yi, int xf, int yf, char col) // logic bishop
{
    int r=objno(xf,yf); // checking if object exists in final destination
    if(fabs(xf-xi)== fabs(yf-yi) ) // checking if move is diagonal
    {
        if(r!=-1)
        { c[r].il('D'); } //killing if object exists in final position
        return 1;
    }
}

```

```
else return 0;
```

```
}
```

```
int logicr(int xi, int yi, int xf, int yf, char col) // logic rook LC
```

```
{
```

```
int r=objno(xf,yf),f;// checking if object exists in final position
```

```
if(fabs(xf-xi)!=0 &&fabs(yf-yi)==0) // moving longitudinally
```

```
{
```

```
for(intnik=1;nik<(fabs(xf-xi));++nik)
```

```
{
```

```
if(xf>xi)
```

```
{ f=objno(xi+nik,yi);
```

```
if(f!=-1)
```

```
return 0; }
```

```
else { f=objno(xf+nik,yi);
```

```
if(f!=-1)
```

```
return 0; }
```

```
}
```

```
if(r!=-1)
```

```
{c[r].il('D');} // kill
```

```
return 1;}
```

```
else if(fabs(xf-xi)==0 &&fabs(yf-yi)!=0) // moving horizontally
```

```
{
```

```

        for(int nik=1;nik<(fabs(yf-yi));++nik)
        {
            if(yf>yi)
            { f=objno(xi,yi+nik);
              if(f!=-1)
                return 0; }
            else { f=objno(xi,yf+nik);
                  if(f!=-1)
                    return 0; }
        }
    if(r!=-1)
        {c[r].il('D');}          // kill
    return 1;
}

else return 0;
} // logically certified

int logick(int xi, int yi, int xf, int yf,char col) // logic knight LC
{
    int r=objno(xf,yf);
    if(fabs(xf-xi)>=1 &&fabs (yf-yi)>=1) // check if move is logical
        if(fabs(xf-xi)+fabs(yf-yi)==3)
            {

```

```

if(r!=-1)
    {c[r].il('D');} //kill
    return 1; }
return 0;
}

```

```

int logic_(int xi, int yi, int xf, int yf, char col) // logic king Lc
{
    int r = objno(xf,yf); // check if another object exists in final position
    int k = fabs(xf-xi);
    int j = fabs(yf-yi);
    if(k==1 && j==1)
    {
        if(r!=-1)
            { c[r].il('D');} // killing opponent
            return 1;    }
    else if( k==1 && j==0)
        { if(r!=-1) // checking if object exists
            { c[r].il('D');} // killing opponent
            return 1; }
    else if( k==0 && j==1)
        { if(r!=-1) //checking if object exists
            { c[r].il('D');} //killing opponent

```



```

        return 1; }

return 0;

}

int logicp(int xi, int yi, int xf, int yf, char col) //logic pawn LT
{
int r=objno(xf,yf),e=objno(xi,yi),f;
if(col=='W')
    { f = objno(xi,yi+1);
if(c[e].retm() == 0 ) // no need to worry about returning -1 as input
takes care of that
{ if ((xf-xi == 0 && yf-yi == 2) && (f==-1))
    { c[e].im(1); return 1;}
}
if(xf-xi == 0 && yf-yi == 1)
    return 1;
else if(((xi-xf == 0) || (fabs(xf-xi) == 1)) && (yf-yi==1) ) // if an object is
not moving straight it is going to kill
    { if(r!=-1) // check if object exists
        { c[r].il('D'); // killing the opponent
        return 1; }
    }
else

```

```

    return 0;

}

} // close the 1st if

//The else part is for the movement of the black candidates
else

    { f = objno(xi,yi-1);
if(c[e].retm() == 0 )
    { if ((xf-xi == 0 && yi-yf == 2) && ( f== -1))
        { c[e].im(1); return 1;}
    }

if(yi-yf == 1 && xi-xf == 0)

    return 1;

else { if((fabs(xi-xf)==1)&& (yi-yf==1) ) // if an object is not moving
straight it is going to kill

    { if(r!=-1) // check if object exists

        {   c[r].il('D');    // killing the opponent

            return 1;  }

        }

else return 0;

    }

} // close else component

return 0; } // close the function

```

```

int self(int a, int b, char col) // check if self kill
{
int s=objno(a,b);
if(s==-1)
    return 0;
else if(c[s].retc() == col && c[s].retl() == 'L' ) // check if same member
and if he if alive
    return 1;
else
    return 0;
}

```

```

int checker(intxi,intyi,char col) // check if object exists and doesnt
belong to user
{
int a,b,retv=1;
for(inti=0;i<32;++i)
{
a=c[i].retx(); //finding the object
b=c[i].rety();
if(xi==a && yi==b)
    if(c[i].retc()==col && c[i].retl()=='L') // see if object belongs to user
        retv=0;
}
}

```

```

    }
return retv;
}

int input(int a) // act on user vs user game
{
    if(a%2==0)
    {
        col='W';
        cout<<"It is whites turn ";
    }
    else
    {
        col='B';
        cout<<"It is Blacks turn ";
    }
    start: cout<<"\nPlease enter initial coordinates ";
    cin>>xi>>yi;
    if(xi==9 &&yi==9) // save and exit
    {
        save(a);exit(0);}
    if(xi==11 || yi == 11) // forfeit turn
    {
        return 0;}
    if(checker(xi,yi,col)) // making sure object exists
    {
        cout<<"Wrong co-ordinates .Try again \n ";
        goto start;
    }
    z=objno(xi,yi); // to find the object number . wont give a wrong
    number (>32 || <0 ) as checker makes sure of that
    efc: cout<<"Enter final coordinates ";

```

```

cin>>xf>>yf;
if(xf == 0 || yf == 0)
    goto start;
if(self(xf,yf,col)) // self kill ? function works fine
    {cout<<"Cannot kill your own members ";
    getch();
    goto efc; }
wit=c[z].retn();
switch(wit) //calling respective logic
{
    case 'P':yrn=logicip(xi,yi,xf,yf,col);
        break;
    case 'K':yrn=logick(xi,yi,xf,yf,col);
        break;
    case 'R':yrn=logicr(xi,yi,xf,yf,col);
        break;
    case 'B':yrn=logicb(xi,yi,xf,yf,col);
        break;
    case 'Q':yrn=logicq(xi,yi,xf,yf,col);
        break;
    case '*':yrn=logic_(xi,yi,xf,yf,col);
        break;
}

```

```

if(yrn!= 1)
    { cout<<"Wrong coordinates ";
      goto efc; }
c[z].ix(xf);
c[z].iy(yf);
return 0;    }// close input function

```

```

void graph() // displaying chessboard

```

```

{ system("color 0F");
  inti=1,j=1,k=0,z;
  cout<<" ";
  for(;i<=8;++i)
      cout<<" "<<i<<" ";
  cout<<endl<<endl<<"1 ";
  for(i=1;k<=64;++i,++k)
      {
          if(i%9 == 0 )
              {
                  ++j; i=1;
                  cout<<endl<<endl;
                  if(j<=8)cout<<j<<" ";
              }
      }
  cout<<" ";

```

```

z=objno(i,j);
if(z!=-1) // checking if object exists
{
    if(c[z].retl() == 'L') // checking if object is alive
    if(c[z].retc()=='B')
        {color(2);
        cout<<c[z].retn();
        color(15); }
else
    cout<<c[z].retn();
}
else
    cout<<" ";
cout<<" "; }
}

void save(int x) // save file
{ofstream coin;          //creation of object
coin.open("file.txt");    //creation of text file
for(inti=0;i<32;++i) // loop to run 32 times
    coin<< c[i].retn()<<" "<<c[i].retl()<<" "<<c[i].retc()<<" "<<c[i].retx()<<"
"<<c[i].rety()<<" "<<c[i].retm()<<endl;//save all information regarding
each coin
    coin<<x;

```

```
coin.close(); //close text file
```

```
}
```

```
int input(intk,charai) // Man vs comp game
```

```
{ if(k%2==0)
```

```
{
```

```
col='W';
```

```
cout<<"It is whites turn ";
```

```
start: cout<<"\nPlease enter initial coordinates ";
```

```
cin>>xi>>yi;
```

```
if(xi==9 || yi==9) // save and exit
```

```
{save(k);exit(0);}
```

```
if(xi==11 || yi == 11) // forfeit turn
```

```
{return 0;}
```

```
if(checker(xi,yi,col)) // making sure object exists
```

```
{ cout<<"Wrong co-ordinates .Try again \n ";
```

```
goto start; }
```

```
z=objno(xi,yi); // to find the object number . Will not give a wrong  
number (>32 || <0 ) as checker makes sure of that
```

```
efc: cout<<"Enter final coordinates ";
```

```
cin>>xf>>yf;
```

```
if(xf == 0 || yf == 0)
```

```
goto start;
```



```

if(self(xf,yf,col)) // self kill
{
    cout<<"Cannot kill your own members ";
    getch();
    goto efc; }
wit=c[z].retn();
switch(wit) // calling respective logic
{
    case 'P':yrn=logicip(xi,yi,xf,yf,col);
        break;
    case 'K':yrn=logick(xi,yi,xf,yf,col);
        break;
    case 'R':yrn=logicr(xi,yi,xf,yf,col);
        break;
    case 'B':yrn=logicb(xi,yi,xf,yf,col);
        break;
    case 'Q':yrn=logicq(xi,yi,xf,yf,col);
        break;
    case '*':yrn=logic_(xi,yi,xf,yf,col);
        break;
}
if(yrn!=1)
{
    cout<<"Wrong coordinates ";
    goto efc; }

```

```

c[z].ix(xf);
c[z].iy(yf); } // closing users turn
else
{ initai:
xi=retrand();
yi=retrand();
col='B';
if(checker(xi,yi,col))          // making sure object exists
    goto initai;
z=objno(xi,yi); // to find the object number . Will not give a wrong
number (>32 || <0 ) .checker makes sure of that
finai:  xf=retrand();
yf=retrand();
if(self(xf,yf,col)) // self kill
    goto finai;
wit=c[z].retn();
switch(wit) // calling respective logic
{
    case 'P':yrn=logicip(xi,yi,xf,yf,col);
        break;
    case 'K':yrn=logick(xi,yi,xf,yf,col);
        break;
    case 'R':yrn=logicr(xi,yi,xf,yf,col);

```

```

        break;
    case 'B':yrn=logicb(xi,yi,xf,yf,col);
        break;
    case 'Q':yrn=logicq(xi,yi,xf,yf,col);
        break;
    case '*':yrn=logic_(xi,yi,xf,yf,col);
        break;
    }
if(yrn!=1)
{ goto initai;}
c[z].ix(xf);
c[z].iy(yf);
    }
return 0; }

int main()    // S8

{ system("title A project by Ajay and Vineet "); // display chess as the
title of the .exe file

process();

getch();

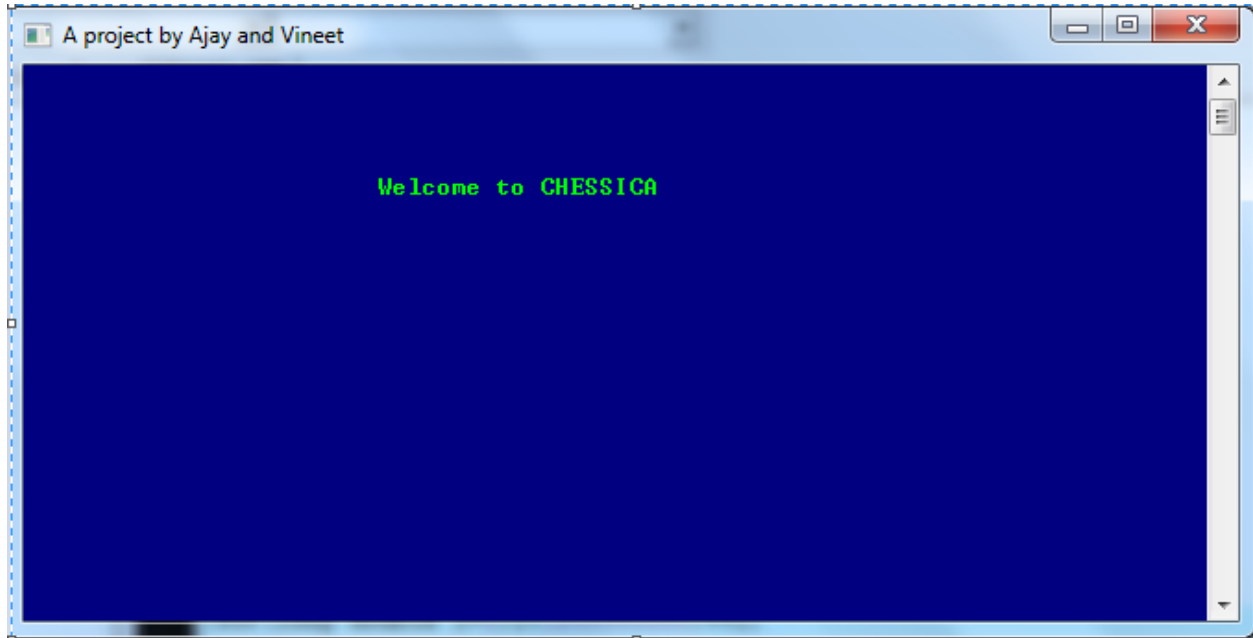
return 0;

}

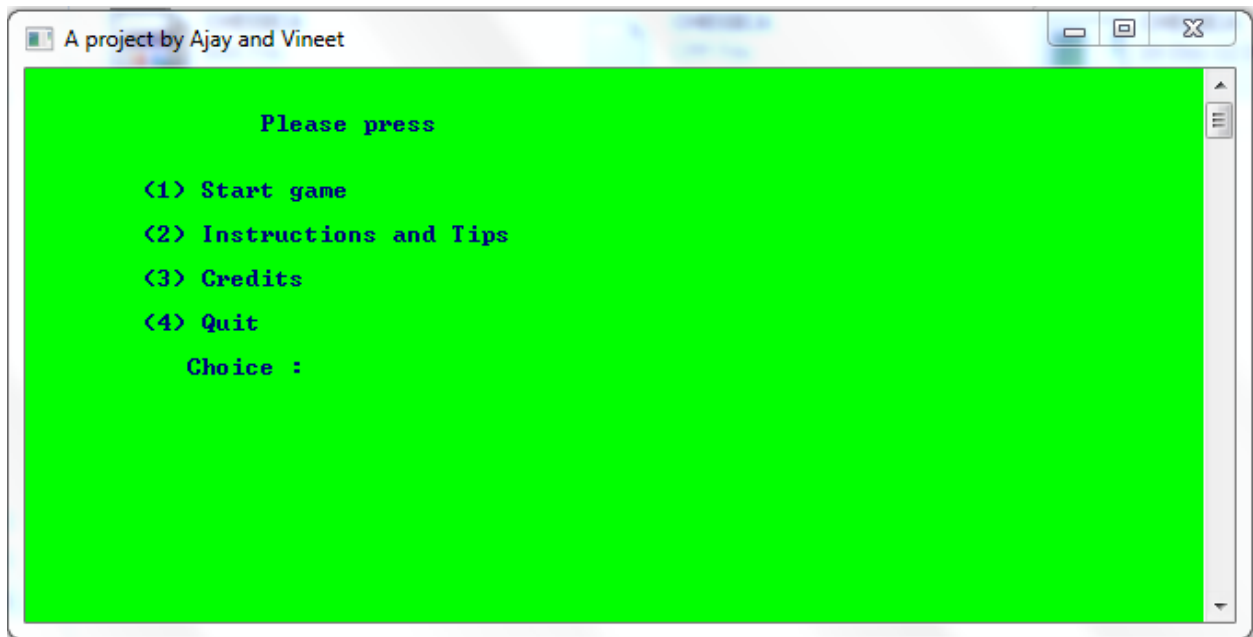
```

SAMPLE OUTPUT

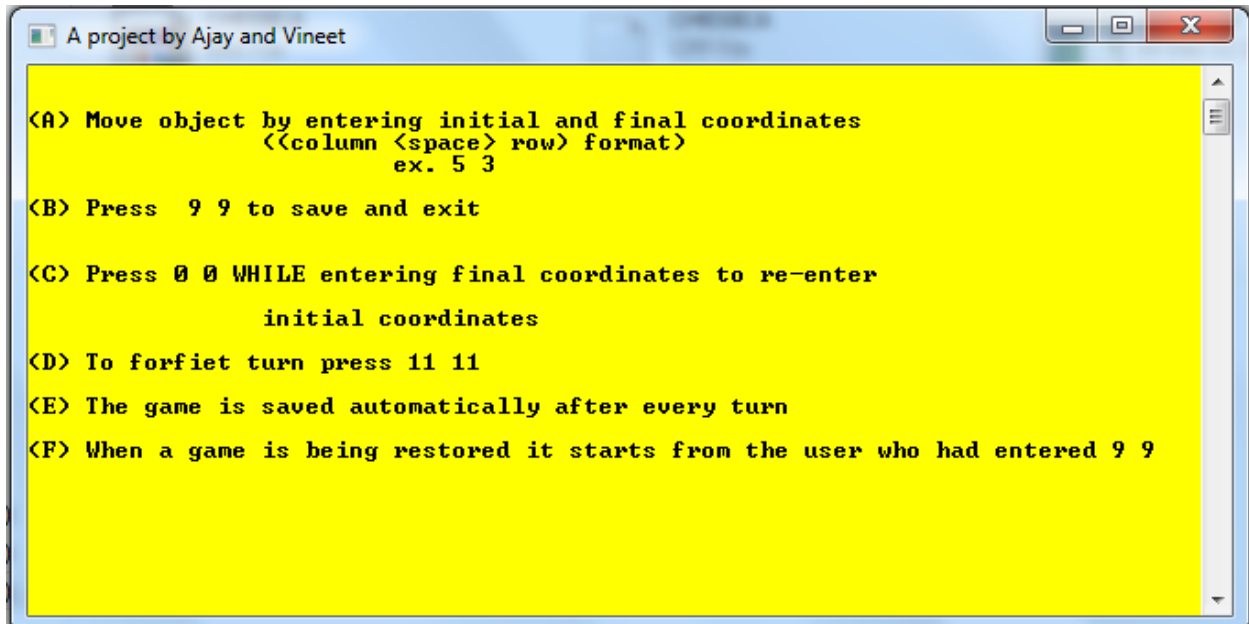
+ Introduction Page



+ Main Page



+ Instructions Page



```
A project by Ajay and Vineet

<A> Move object by entering initial and final coordinates
      <<column <space> row> format>
      ex. 5 3

<B> Press 9 9 to save and exit

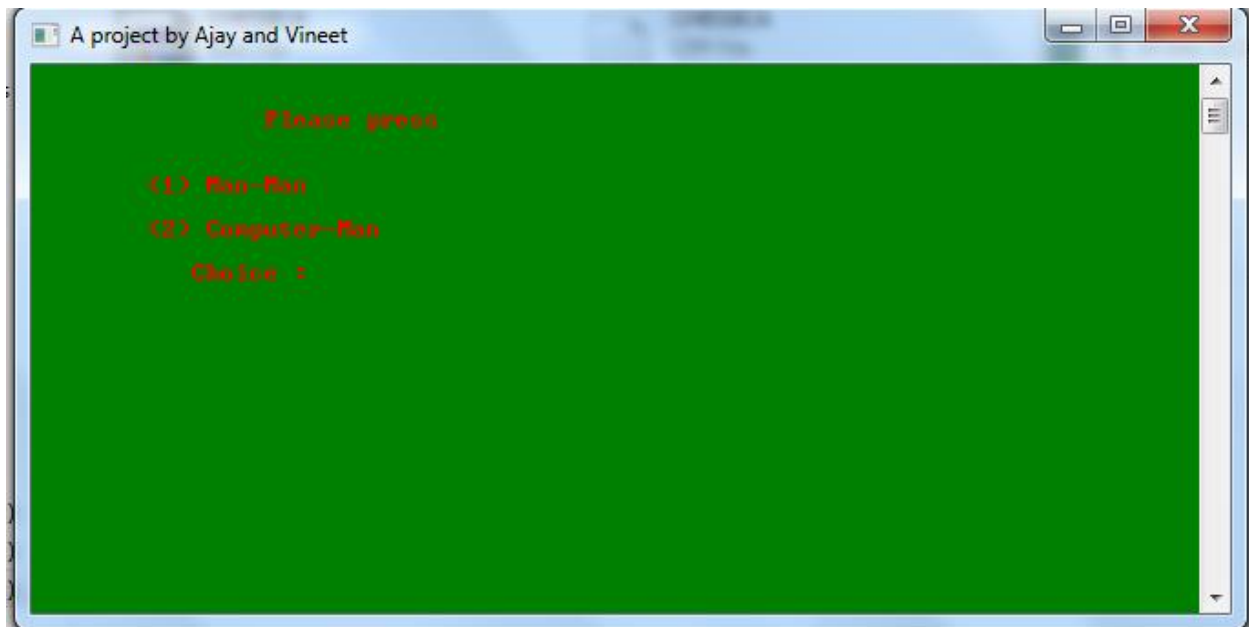
<C> Press 0 0 WHILE entering final coordinates to re-enter
      initial coordinates

<D> To forfeit turn press 11 11

<E> The game is saved automatically after every turn

<F> When a game is being restored it starts from the user who had entered 9 9
```

+ Type of chess game Menu

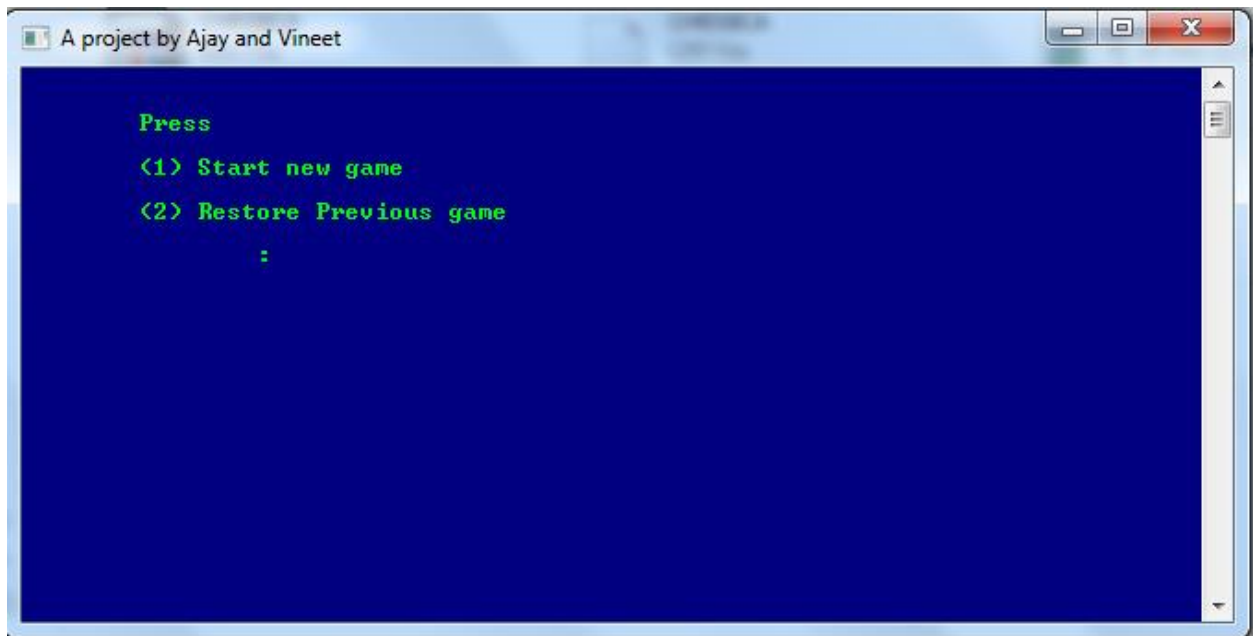


```
A project by Ajay and Vineet

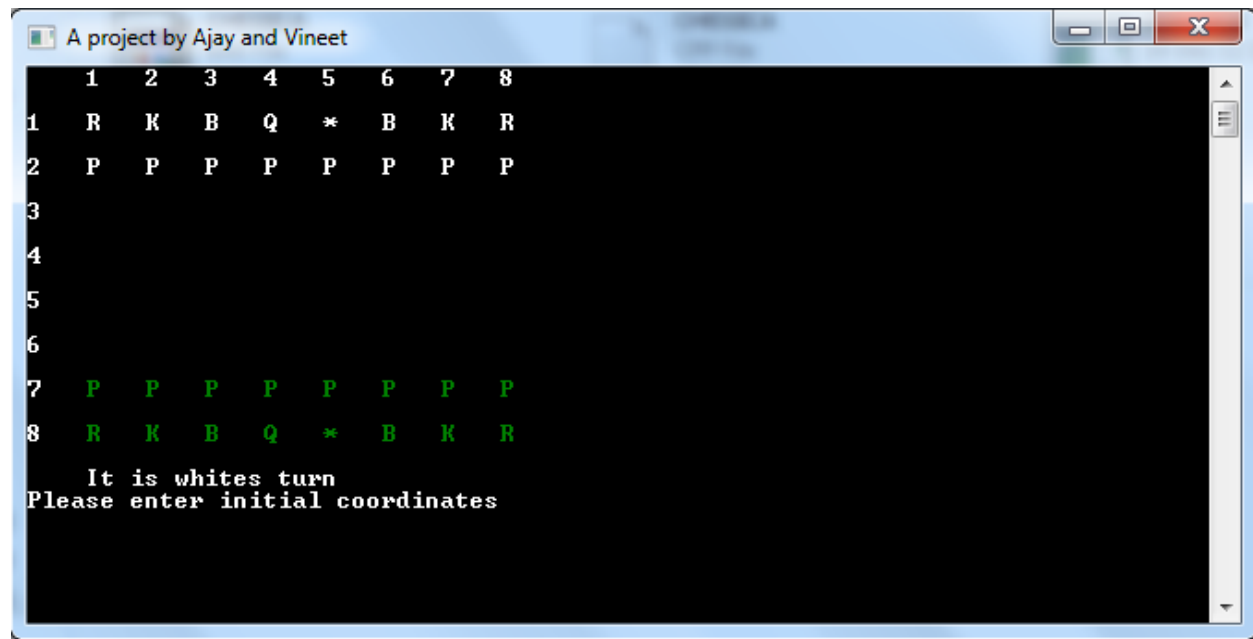
Please press

<1> Man-Man
<2> Computer-Man
Choice :
```

+ Choice of starting a new game or loading an old one



+ Chess board at the beginning of the game



```
A project by Ajay and Vineet

 1  2  3  4  5  6  7  8
1  R  K  B  Q  *  B  K  R
2  P  P  P  P  P  P  P  P
3
4
5
6
7  P  P  P  P  P  P  P  P
8  R  K  B  Q  *  B  K  R

It is whites turn
Please enter initial coordinates 1 2
Enter final coordinates 1 3
```

Chess game after the first move

```
A project by Ajay and Vineet

 1  2  3  4  5  6  7  8
1  R  K  B  Q  *  B  K  R
2      P  P  P  P  P  P  P
3  P
4
5
6
7  P  P  P  P  P  P  P  P
8  R  K  B  Q  *  B  K  R

It is Blacks turn
Please enter initial coordinates
```

When wrong coordinates are entered

```
A project by Ajay and Vineet

  1  2  3  4  5  6  7  8
1  R  K  B  Q  *  B  K  R
2      P  P  P  P  P  P  P
3  P
4
5
6
7  P  P  P  P  P  P  P  P
8  R  K  B  Q  *  B  K  R

    It is Blacks turn
Please enter initial coordinates
1 8
Enter final coordinates 1 7
Cannot kill your own members
```

```
A project by Ajay and Vineet

  1  2  3  4  5  6  7  8
1  R      Q  *  B      R
2  P      P  P  P  P      P
3  B      K
4      P      P
5      K      P
6      P  R
7  P  P  P      P  P
8  R  K  B  Q  *  B  K

    It is Blacks turn
Please enter initial coordinates 3 4
Wrong co-ordinates .Try again
Please enter initial coordinates _
```


Chess game at a later stage

```
C:\ A project by Ajay and Vineet
 1  2  3  4  5  6  7  8
1  R      Q * B      R
2  P      P  P  P  P      P
3  B          K
4      P          P
5          K          P
6          P  R
7  P  P  P          P  P
8  R  K  B  Q  *  B  K

It is Blacks turn
Please enter initial coordinates
```

Winner's page

```
C:\ A project by Ajay and Vineet

Black is the winner
```

BIBLIOGRAPHY

1. Computer Science with C++ by Sumita Arora
2. Cplusplus.com
3. Cprogramming.com
4. Daniweb.com
5. Youtube C++ videos

