COMPUTER PROJECT

-BY TANMAY SARAIYA

<12-H>

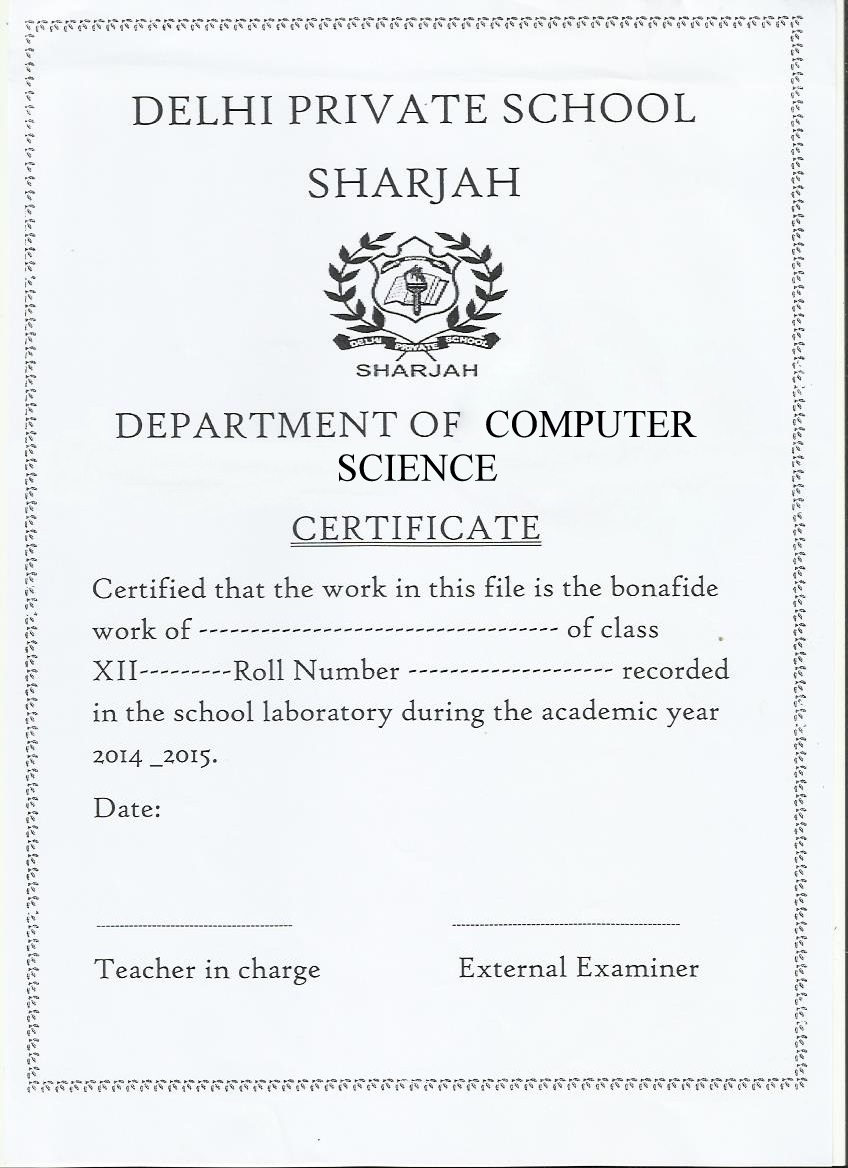
A CODE DEVELOPED BY TANMAY SARAIYA AND MADHAV SUBRAMANIYAM AS A PART OF THEIR CBSE BOARD PROJECT.THE CODE IS FOR THE GAME TIC-TAC-TOE.IT REQUIRED LOGICAL THINKING AND KNOWLEDGE OF THE LANGUAGE C++.

2014-15

TANMAY

DELHI PRIVATE SCHOOL, SHARJAH

2014-15



**INDEX**

|  |  |  |  |
| --- | --- | --- | --- |
| Sr.no. | content | Page no | |
| 1. | Acknowledgement | | 3 |
| 2. | Aim | | 4 |
| 3. | About | | 4 |
| 4. | Concepts used | | 5 |
| 5. | Features | | 5 |
| 6. | Code | | 6 |
| 7. | User Manual | | 22 |
| 8. | Screenshots | | 25 |
| 9. | Compact Disk | | 29 |

Acknowledgement

I have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations. I would like to extend my sincere thanks to all of them.

I am highly indebted to my teacher

Mrs.Gurpreet Modi and other COMPUTER DEPT. members for their guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing the project.

I would like to express my gratitude towards my parents & colleagues for their kind co-operation and encouragement which help me in completion of this project.

AIM:

The aim is to highlight the abilities of algorithmic formulation, modular programming, optimization preparation,

Systematic documentation and other associated aspects of software development.

ABOUT:

This program is made of several concepts taught to us at school.

This program involves logical thinking and knowledge about the computer language C++.

The several concepts used here are:-

1. Object oriented programming
2. Inheritance
3. Classes and objects
4. Polymorphism
5. Functions
6. Data structures
7. Data file handling
8. Flow of control

FEATURES:

The program various features;

1. Initially it is very user friendly and allows the user to quit the game at any instance of time.

2. The program has a “high score” feature that allows you to review your previous scores.

TIC-TAC-TOE

<Code>

//header files required

#include <stdlib.h>

#include <string.h>

#include <stdio.h>

#include <time.h>

#include <fstream.h>

#include <process.h>

#include <conio.h>

class board

{

char playspace[3][3];//main board

char p\_na; //\*player\_nature//for X and O

char c\_na; //\*computer\_nature//for X and O

int c1; //coordinates

int c2; //coordinates

int getAImove();//AI

int AItactics;//AI

int translate\_coord(int cord);//to place the c1 and c2

public:

int check\_w(); //1= p wins;0=p looses 2= draw

int player\_in();//player input

char get\_p\_na();//return p\_na;

char get\_c\_na();//return c\_na;

void switch\_players();

void comp\_in();//comp input

void printboard();//display board

void resetboard();//initialize board

int assign\_coord(int cord, char na);//assign c1&c2

void inp\_na();//input for X or 0 \*nature

board()//constructor

{

resetboard();

}

};

class extras:public board

{

int p\_score; //player score

int c\_score; //computer score

char user\_name[10]; //user name

public:

void change\_score();//change scores after gameplay

void display\_score();//display after game

void show\_score();//view high scores

void file\_write();//save scores

void copy\_name(char name[])//copy user\_name to name for display in high scores

{

strcpy(name,user\_name);

}

void enter\_name(char abc[])//to write the user\_name

{

strcpy(user\_name,abc);

}

extras()//constructor

{

p\_score=0;

c\_score=0;

strcpy(user\_name,"USER 1");

}

};

extras player;//obj of class extras

char ignore[2];

void extras::file\_write()//to write score into file

{

ofstream fout("highscore.bin",ios::binary|ios::app);

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

fout.close();

}

void extras::show\_score()//for file o/p

{

cout<<" "<<p\_score<<" "<<c\_score<<endl;

}

void extras::change\_score()

{

if(board::check\_w()==1) //player wins

p\_score+=1;

else if(board::check\_w()==0) //player looses

c\_score+=1;

else if(board::check\_w()==2) //add 1 pt to each if draw

{

p\_score+=1;

c\_score+=1;

}

}

void board::switch\_players()

{

char temp=p\_na;

p\_na=c\_na;

c\_na=temp;

}

void extras::display\_score()

{

cout<<" scores"<<endl;

cout<<user\_name<<" "<<p\_score<<endl;

cout<<"computer"<<" "<<c\_score<<endl;

file\_write();

}

void board::resetboard()

{

playspace[0][0]='1';

playspace[0][1]='2';

playspace[0][2]='3';

playspace[1][0]='4';

playspace[1][1]='5';

playspace[1][2]='6';

playspace[2][0]='7';

playspace[2][1]='8';

playspace[2][2]='9';

}

char board::get\_p\_na()

{

return p\_na;

}

char board::get\_c\_na()

{

return c\_na;

}

void board::inp\_na() //to input the nature of user (X/O)

{

while(1)

{

cout<<"to choose 'X' press 1"<<endl;

cout<<"to choose '0' press 0"<<endl;

int no;

cin>>no;

if(no==1)

{

p\_na='X';

c\_na='O';

break;

}

else if(no==0)

{

p\_na='O';

c\_na='X';

break;

}

else

{

cout << "Invalid" << endl;

}

}

}

int board::translate\_coord(int cord)//to fill in the array

{

if(cord==1)

{ c1=0;c2=0; }

else if(cord==2)

{ c1=0;c2=1; }

else if(cord==3)

{ c1=0;c2=2; }

else if(cord==4)

{ c1=1;c2=0; }

else if(cord==5)

{ c1=1;c2=1; }

else if(cord==6)

{ c1=1;c2=2; }

else if(cord==7)

{ c1=2;c2=0; }

else if(cord==8)

{ c1=2;c2=1; }

else if(cord==9)

{ c1=2;c2=2; }

else

return 0;

return 1;

}

int board::assign\_coord(int cord, char na)//to assign coordinates to user input

{

if(!translate\_coord(cord))

{

return 1;

}

if(playspace[c1][c2]=='X'|| playspace[c1][c2]=='O')

return 2;

else {

playspace[c1][c2]=na;

return 0;

}

}

int board::player\_in()//player's chance

{

cout<<"your turn(enter 10 to quit) ="<<endl;

int cord;

cin>>cord;

if(cord==10)

return 0;

while(assign\_coord(cord,p\_na))

{

cout << "Invalid move" << endl;

cout<<"your turn(enter 10 to quit) ="<<endl;

cin>>cord;

if(cord==10)

return 0;

}

clrscr();

printboard();

check\_w();

return 1;

}

int board::getAImove()

{

//Priority 1 moves (Winning)

int i;

for(i=0;i<3;i++) // Row Check

{

int flag[3]={0,0,0};

for(int k=0;k<3;k++)

{

if(playspace[i][k]==c\_na)

flag[k]=1;

else if(playspace[i][k]==p\_na)

flag[k]=2;

}

if(flag[0]==1 && flag[1]==1 && flag[2]!=2)

return (i\*3)+3;

else if(flag[1]==1 && flag[2]==1 && flag[0]!=2)

return (i\*3)+1;

else if(flag[2]==1 && flag[0]==1 && flag[1]!=2)

return (i\*3)+2;

}

for(i=0;i<3;i++) // Column Check

{

int flag[3]={0,0,0};

for(int k=0;k<3;k++)

{

if(playspace[k][i]==c\_na)

flag[k]=1;

else if(playspace[k][i]==p\_na)

flag[k]=2;

}

if(flag[0]==1 && flag[1]==1 && flag[2]!=2)

return 7+i;

else if(flag[1]==1 && flag[2]==1 && flag[0]!=2)

return 1+i;

else if(flag[2]==1 && flag[0]==1 && flag[1]!=2)

return 4+i;

}

for(i=0;i<2;i++) // Diagonals Check

{

int flag[3]={0,0,0};

for(int k=0;k<3;k++)

{

if(i==0)

{

if(playspace[k][k]==c\_na)

flag[k]=1;

else if(playspace[k][k]==p\_na)

flag[k]=2;

}

else if(i==0)

{

if(playspace[k][2-k]==c\_na)

flag[k]=1;

else if(playspace[k][2-k]==p\_na)

flag[k]=2;

}

}

if(i==0)

{

if(flag[0]==1 && flag[1]==1 && flag[2]!=2)

return 9;

else if(flag[1]==1 && flag[2]==1 && flag[0]!=2)

return 1;

else if(flag[2]==1 && flag[0]==1 && flag[1]!=2)

return 5;

}

else if(i==1)

{

if(flag[0]==1 && flag[1]==1 && flag[2]!=2)

return 7;

else if(flag[1]==1 && flag[2]==1 && flag[0]!=2)

return 3;

else if(flag[2]==1 && flag[0]==1 && flag[1]!=2)

return 5;

}

}

//Priority 2 (High priority defence)

for(i=0;i<3;i++) // Row Check

{

int flag[3]={0,0,0};

for(int k=0;k<3;k++)

{

if(playspace[i][k]==p\_na)

flag[k]=1;

else if(playspace[i][k]==c\_na)

flag[k]=2;

}

if(flag[0]==1 && flag[1]==1 && flag[2]!=2)

return (i\*3)+3;

else if(flag[1]==1 && flag[2]==1 && flag[0]!=2)

return (i\*3)+1;

else if(flag[2]==1 && flag[0]==1 && flag[1]!=2)

return (i\*3)+2;

}

for(i=0;i<3;i++) // Column Check

{

int flag[3]={0,0,0};

for(int k=0;k<3;k++)

{

if(playspace[k][i]==p\_na)

flag[k]=1;

else if(playspace[k][i]==c\_na)

flag[k]=2;

}

if(flag[0]==1 && flag[1]==1 && flag[2]!=2)

return 7+i;

else if(flag[1]==1 && flag[2]==1 && flag[0]!=2)

return 1+i;

else if(flag[2]==1 && flag[0]==1 && flag[1]!=2)

return 4+i;

}

for(i=0;i<2;i++) // Diagonals Check

{

int flag[3]={0,0,0};

for(int k=0;k<3;k++)

{

if(i==0)

{

if(playspace[k][k]==p\_na)

flag[k]=1;

else if(playspace[k][k]==c\_na)

flag[k]=2;

}

else if(i==0)

{

if(playspace[k][2-k]==p\_na)

flag[k]=1;

else if(playspace[k][2-k]==c\_na)

flag[k]=2;

}

}

if(i==0)

{

if(flag[0]==1 && flag[1]==1 && flag[2]!=2)

return 9;

else if(flag[1]==1 && flag[2]==1 && flag[0]!=2)

return 1;

else if(flag[2]==1 && flag[0]==1 && flag[1]!=2)

return 5;

}

else if(i==1)

{

if(flag[0]==1 && flag[1]==1 && flag[2]!=2)

return 7;

else if(flag[1]==1 && flag[2]==1 && flag[0]!=2)

return 3;

else if(flag[2]==1 && flag[0]==1 && flag[1]!=2)

return 5;

}

}

if(c\_na=='X')

{

int flag=0;

for(int i=1;i<10;i++)

{

translate\_coord(i);

if(playspace[c1][c2]==p\_na||playspace[c1][c2]==c\_na)

flag=1;

}

if(flag==0) //Priority 3-X-1 (Opening as X)

{

srand(time(NULL));

AItactics=rand()%2;

if(AItactics==0)

{

return 5;

}

else

return 9;

}

else //Priority 3-X-2 (Continuing the attack)

{

if(playspace[1][1]==c\_na && playspace[0][0]==c\_na) // Trap finishing moves Set 1

{

if(playspace[0][2]!=p\_na)

return 3;

else if(playspace[2][0]!=p\_na)

return 7;

}

else if(playspace[1][1]==c\_na && playspace[2][0]==c\_na)

{

if(playspace[0][0]!=p\_na)

return 1;

else if(playspace[2][2]!=p\_na)

return 9;

}

else if(playspace[1][1]==c\_na && playspace[0][2]==c\_na)

{

if(playspace[0][0]!=p\_na)

return 1;

else if(playspace[2][2]!=p\_na)

return 9;

}

else if(playspace[1][1]==c\_na && playspace[2][2]==c\_na)

{

if(playspace[0][2]!=p\_na)

return 3;

else if(playspace[2][0]!=p\_na)

return 7;

}

else if(playspace[0][0]==c\_na && playspace[0][2]==c\_na)//Trap finishing moves Set 2

{

if(playspace[2][0]!=p\_na)

return 7;

else if(playspace[2][2]!=p\_na)

return 9;

}

else if(playspace[0][0]==c\_na && playspace[2][0]==c\_na)

{

if(playspace[0][2]!=p\_na)

return 3;

else if(playspace[2][2]!=p\_na)

return 9;

}

else if(playspace[0][0]==c\_na && playspace[2][2]==c\_na)

{

if(playspace[0][2]!=p\_na)

return 3;

else if(playspace[2][0]!=p\_na)

return 7;

}

else if(playspace[0][2]==c\_na && playspace[2][2]==c\_na)

{

if(playspace[2][0]!=p\_na)

return 7;

else if(playspace[0][0]!=p\_na)

return 1;

}

else if(playspace[0][2]==c\_na && playspace[2][0]==c\_na)

{

if(playspace[2][2]!=p\_na)

return 9;

else if(playspace[0][0]!=p\_na)

return 1;

}

else // Secondary moves

{

if(playspace[0][0]!=p\_na && playspace[0][0]!=c\_na)

return 1;

else if(playspace[0][2]!=p\_na && playspace[0][2]!=c\_na)

return 3;

else if(playspace[2][0]!=p\_na && playspace[2][0]!=c\_na)

return 7;

}

}

}

else

{

int flag=0;

for(int i=1;i<10;i++)

{

translate\_coord(i);

if(playspace[c1][c2]==p\_na||playspace[c1][c2]==c\_na)

flag++;

}

if(flag==1) // Priority 3-O-1(Opening O)

{

if(playspace[1][1]!=p\_na)

return 5;

else

return 9;

}

else

{

if(playspace[0][0]==p\_na&&playspace[2][2]==p\_na)

if(playspace[1][0]!=p\_na && playspace[1][0]!=c\_na)

return 4;

else if(playspace[0][1]!=p\_na && playspace[0][1]!=c\_na)

return 2;

else if(playspace[1][2]!=p\_na && playspace[1][2]!=c\_na)

return 6;

else if(playspace[2][1]!=p\_na && playspace[2][1]!=c\_na)

return 8;

}

}

while(1)

{

srand(time(NULL));

int x=rand()%9+1;

translate\_coord(x);

if(playspace[c1][c2]!=p\_na && playspace[c1][c2]!=c\_na)

return x;

}

}

void board::comp\_in()

{

assign\_coord(getAImove(),c\_na);

clrscr();

printboard();

check\_w();

}

int board::check\_w()

{

int plr=1;

int cpt=1;

for(int i=0;i<3;i++)

{

if((playspace[i][0]==playspace[i][1])&&(playspace[i][0]==playspace[i][2])) //for row check

{

if(playspace[i][0]==p\_na)

plr=0;

else

cpt=0;

break;

}

else if((playspace[0][i]==playspace[1][i])&&(playspace[0][i]==playspace[2][i]))//for col check

{

if(playspace[0][i]==p\_na)

plr=0;

else

cpt=0;

break;

}

else if((playspace[0][0]==playspace[1][1])&&(playspace[0][0]==playspace[2][2]))//for major d. check

{

if(playspace[0][0]==p\_na)

plr=0;

else

cpt=0;

break;

}

else if((playspace[0][2]==playspace[1][1])&&(playspace[1][1]==playspace[2][0]))//for minor d. check

{

if(playspace[0][2]==p\_na)

plr=0;

else

cpt=0;

break;

}

}

if(plr==0)//player matches the conditions

return 1;

else if(cpt==0)//comp matches the conditions

return 0;

else

return 2;//no result

}

void board::printboard()//show the array

{

for(int i=0;i<2;i++)

{

cout<<"\t\t\t\t";

for(int j=0;j<2;j++)

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

cout << playspace[i][2];

cout << endl;

cout << "\t\t\t -----------" << endl;

}

cout<<"\t\t\t\t";

for(int j=0;j<2;j++)//run the loop for the first two elements of third row

cout << playspace[2][j]<<" | ";

cout << playspace[2][2] << endl << endl;

}

int playtime()//main gameplay

{

int i;

cout<<"player starts"<<endl;

for(i=1;i<10;i++) //to play 9 times max

{

player.board::check\_w(); //check for win or draw

if(i%2!=0)

{

if(player.board::get\_p\_na()=='X')

{

if(!player.board::player\_in())

return 0;

if(player.board::check\_w()==1)

break;

}

else

{

player.board::comp\_in();

if(player.board::check\_w()==0)

break;

}

}

else

{

if(player.board::get\_p\_na()=='X')

{

player.board::comp\_in();

if(player.board::check\_w()==0)

break;

}

else

{

if(!player.board::player\_in())

return 0;

if(player.board::check\_w()==1)

break;

}

}

}

if(player.board::check\_w()==1) //assign the scores with the match played

cout<<"you win"<<endl;//add point to player

else if(player.board::check\_w()==0)

cout<<"you loose."<<endl;//add point to computer

else if(player.board::check\_w()==2 && i==9)

cout<<"draw"<<endl;//add 1 point to both

player.change\_score();

return 1;

}

//view high score function//

int view\_hghscr() // main menu-return 0;

{

ifstream fin("highscore.bin",ios::binary);

if(!fin)

{

cout<<"file not fund"<<endl;

return 0;

}

char name[10];

player.copy\_name(name);

cout<<" SCORES"<<endl;

cout<<"computer "<<name<<endl;

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

{

player.show\_score();

}

fin.close();

}

int to\_play()//function to accommodate choice 1

{

char choice1;

char pname[20];

cout<<"enter your name = ";

gets(ignore);

gets(pname);//to accept player name

player.enter\_name(pname);

player.board::inp\_na();

clrscr();

while(1)

{

player.resetboard();

player.board::printboard();

if(!playtime())

break;

clrscr();

player.display\_score();

player.resetboard();

player.switch\_players();

cout<<"do you wish to continue?(Y/N)"<<endl;

cin>>choice1;

clrscr();

if(choice1=='n' || choice1=='N')

break;

}

return 0;

}

void main()

{

while(1){

clrscr();

cout<<" @@@@@@@@ @@@@@ @@@ @@@@@@@@ @ @@@ @@@@@@@@ @@@@ @@@@@@"<<endl;

cout<<" @ @ @ @ @ @ @ @ @ @ @ @ @ "<<endl;

cout<<" @ @ @ @ @ @ @ @ @ @ @ @@ "<<endl;

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

cout<<" @ @ @ @ @ @ @ @ @ @ @ @ @ "<<endl;

cout<<" @ @@@@@ @@@ @ @ @ @@@ @ @@@@ @@@@@@"<<endl;

cout<<endl<<endl;

cout<<"-------------------------------MAIN-MENU-------------------------------"<<endl;

cout<<" PRESS FUNCTION"<<endl;

cout<<" 1 TO PLAY"<<endl;

cout<<" 2 TO VIEW HIGH SCORES"<<endl;

cout<<" 3 TO EXIT"<<endl;

int ch1,ch2,flag=0;

cin>>ch1;

switch(ch1)

{

case 3:flag=1;

break;

case 2:while(view\_hghscr())

{

break;

}

while(1)

{

cout<<"To return to MAIN MENU press 0"<<endl;

cin>>ch2;

if(ch2==0)

{

break;

}

else

cout<<"invalid"<<endl;

}

break;

case 1:while(!to\_play())

{

break;

}

break;

default:cout<<"error"<<endl;

break;

}

if(flag==1)

break;

}

exit(1);

}

System Requirements

100 Mhz Processor or above

512 KB RAM

200 KB Free Disk Space

User Manual

Classes Used:

1. Board

Used to create a representation of the board, store its current status and information about the players (User and computer).

Public Functions Used in Board:

1. check\_w() – Used to check whether any of the players has achieved victory.
2. player\_in() – Used to get input from the human player.
3. comp\_in() – Used to get input from the computer.
4. printboard() – Used to display a representation of the board.
5. assign\_coord() – Used to validate the input and call translate\_coord() to translate it.

Private Functions Used in Board:

1. getAImove() – Used to generate the AI move.
2. translate\_coord() – Used to translate the input into coordinates which can be used to modify the board.
3. Extras

Used to extend the board class with additional functions used to handle and save scores.

Public Functions Used in Extras:

1. change\_score() – Used to change the scores.
2. display\_score() – Used to display the current score after each game.
3. show\_score() – Used to display the current win/loss record with the computer.
4. file\_write() – Used to save the scores to an external file.
5. enter\_name() – Used to accept a player name to be saved along with the scores.

Independent Functions:

1. view\_highscr() – Used to access the file and read the high scores and call extras.show\_score() to display them.
2. playtime() – Constitutes the main game loop.
3. to\_play() – Used to set up the environment when the game is first started.

Screen shots:

Main menu



Enter info:

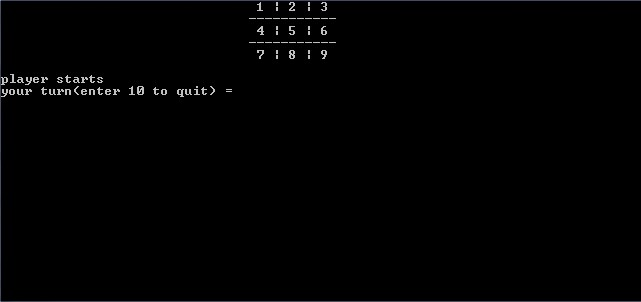


If choice not valid:

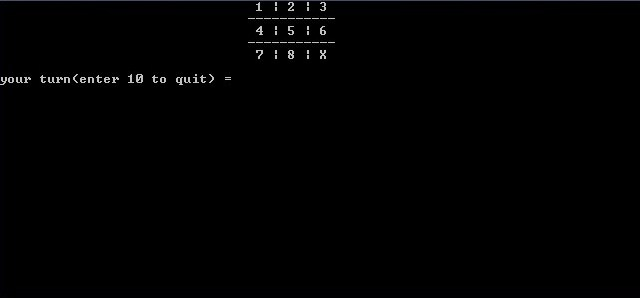


X always starts:

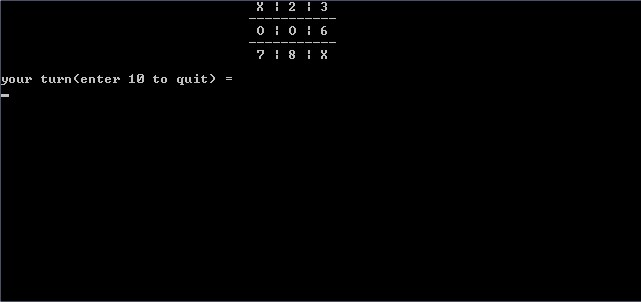
If user is ‘X’

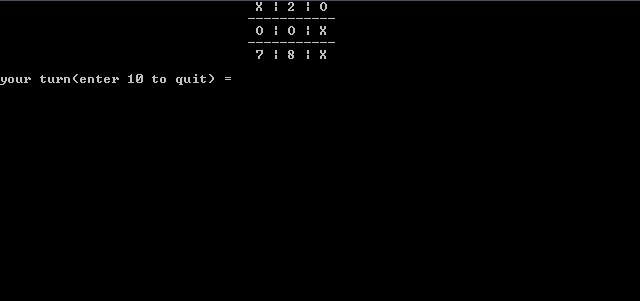


If computer is ‘X’.



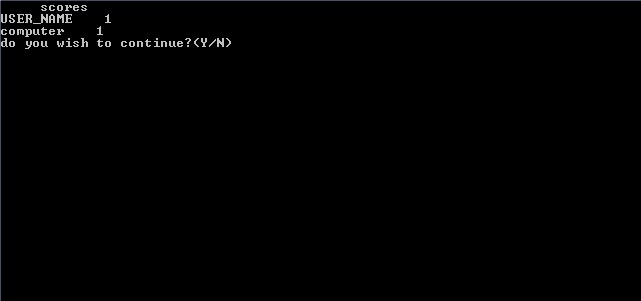
GAME PLAY:



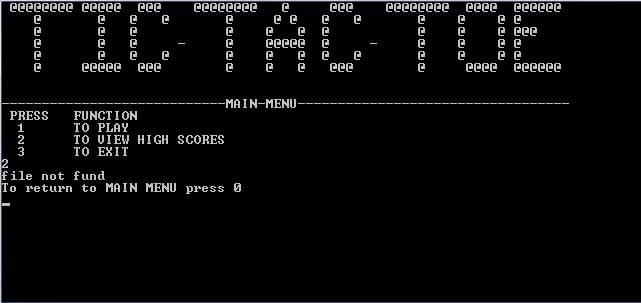


At the end of one game:

If yes than game restarts…



If playing for first time:



Displaying the previous scores:



**Compact disk:**

A copy of the code.