

Dots & Boxes

## REPORT OF PROJECT

By / Abdelrahman alsayed ahmed abdelrizk elwakel (23) && Gamal Eldeen ahmed khalaf alsayed(15)

**Contents**

GAME DESCRIPTION 2

FEATURES 2

[ASSUMPTIONS 3](#_TOC_250004)

[DATA STRUCTURE 4](#_TOC_250003)

DESCRIPTION OF IMPORTANTED FUNCTIONS 4

FLOW CHART AND PSEUDO CODE 6

[USER MANUAL 15](#_TOC_250002)

[SAMPLE RUNS 18](#_TOC_250001)

[REFERENCES 22](#_TOC_250000)

# GAME DESCRICTION :-

**Dots and Boxes** is a pencil-and-paper game for two players (sometimes more). It was first published in the 19th century by Édouard Lucas, who called it **la pipopipette**. It has gone by many other names, including the **game of**

**dots**, **boxes**, **dot to dot grid**, and **pigs in a pen**.

Starting with an empty grid of dots, two players take turns adding a single horizontal or vertical line between two *unjoined* adjacent dots. The player who completes the fourth side of a 1×1 box earns one point and takes another turn. (A point is typically recorded by placing a mark that identifies the player in the box, such as an initial). The game ends when no more lines can be placed. The winner is the player with the most points. The board may be of any size. When short on time, a 2×2 board (a square of 9 dots) is good for beginners. A 5×5 is good for experts.

# GAME FEATURES :-

* It contains an easy main menu with smooth movement between modes
* It supports using UP,DOWN,RIGHT,LEFT to move in whole game
* It contains PC mode (with good AI)
* It contains two modes (2\*2)&&(5\*5) , they are also available in PC mode
* There is a leader board to know your rank between all players
* You can do Undo & Redo during the game but with some restrictions
* You have a dynamic time to know how much time did you spend in the game
* There are three slots to save your game with an easy access
* In the game you can only play with arrow buttons
* You have the option to abort during the game and returning to main menu

# DESIGN OVERVIEW :-

In the beginning of the game there is a main menu it contains all options you will need in the game , it has option of start new game or load a game or to show leader board or to exit the game , you have an easy method to select those options only use UP,DOWN,ENTER buttons to select any option. In the game the board of game is made of asterisks when you connecting between two available asterisks they turn to hashtag , the line between asterisks colored with player’s color .Closed squares have the color of the player who close this

square .There are fields to score ,remaining lines, moves for each player, dynamic timer . There is a frame around the board tell you which player will play (colored with player’s color). When game is finished you are asked if you want to go back to the menu or play again.

-**OVERALL STRUCTURE** :-

Every mode has his own function, this function contains how to switch between two players (1P VS PC) , contains some functions as colors function , Undo & Redo function , arrows function , detect and check function (draw the movement in game’s array and check if this movement available or not) , function of save & load (algorithm save & load has doubled our code lines[as we copy and paste many times ]).

# ASSUMPTIONS :-

* we used an arrow to choose which two asterisks the player want to connect to avoid butting string or anything may crush our game and to easy control for our game
* we use backspace button to give the player chance to return if he has chosen something wrong and gives our game stable movement between different menus
* we made three slots to save games if our players need to save their games and continue at any time
* there is a handling if any player choose two asterisks which is impossible to connect line between them
* we use a frame that colors with player’s color in his turn as it an obvious method to declare who player will play
* there is a handling if any player choose to load game and there is not any saved game
* there is a handling if any player do redo before doing any undo
* if the player want to exit during the game so there is a handling for this
* there is a handling to have the right turn of players after load a saved game
* there a handling when any player load a saved game players can not do undo till they played any available movement
* after finishing the game and player want to player again we give him this option to choose if he want to play again or go to main menu
* when player want to save a game he will need to choose the slot that he want to save on it so there is a handling if he pressed any button else the available buttons to choose the slot he want
* in our game there is a lot of yes or no questions so there is a handling if the players have pressed incorrect button
* undo in PC mode undo PC movements and player’s movements (with a restriction)
* there is a handling if player has chosen a movement that had played before

\*\*Note :- any handling declared above you can try it to know what is happening 

# DATA STRUCTURE :-

* structs
* arrays (char & integers)
* integers
* time\_t

# DESCRIBTION OF IMPORTANT FUNCTIONS :-

* **Colors function :-** we used this to print everything in our game when we clear screen , it prints lines with suitable colors also squares , print our interface
* **Arrows function :-** this to detect the position of the arrow and print the new position of arrow
* **Detect\_check function :-** this to take coordinates of dots and test them (if this line has played before or not , the possibility of playing this movement)
* **Drawing function :-** this draw the movement in the array of the game and store the movement in the array of movements
* **S1 & S2 functions :-** they are used to calculate score for each player also PC , they store the coordinates of closed square in order to calculate score

accurately and not to increase score every time you check this closed square

* **Undo function :-** in this function we control the counter of array of movements in order to remove drawing line , returning hashtag to asterisks , removing colored lines (as we used this counter in the printing colored lines’ algorithm) , check if and square has opened or not and remove coordinates for the array of coordinates but we have 1D array holds a flag to know the closed square belongs to which player (helping us in printing colored squares and helping in redo logic), it is obvious that when player does undo remaining lines decrease , moves of players.

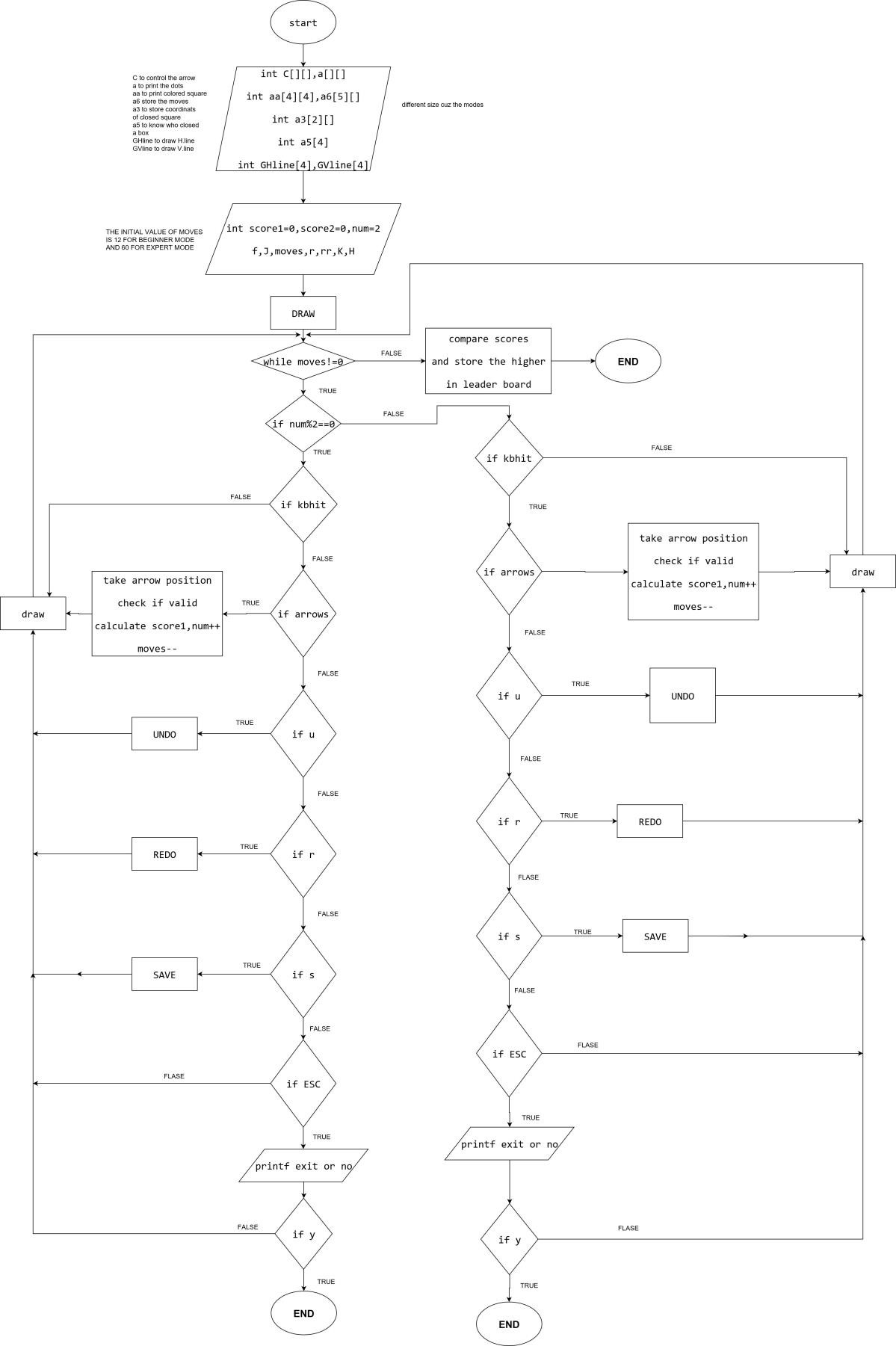
\*\* Undo in mode P1 VS P2 when any player does undo the turn goes directly to another player

\*\*Undo in PC mode when the player does undo computer and player movements will removed till the game’s turn goes to player not PC

* **Redo function :-** reflecting all what we do in undo 
* **Save & Load functions :-** they are obvious but to make slots’ system we have to copy and paste many lines this leads to have those many lines of code (Unfortunately , there is no time to optimize this system but it is efficient)
* **Gotoxy function :-** the greatest function  it helps us a lot in printing and to make our interface as it gives us the ability to change position of the cursor
* **Kbhit() function :-** this function exists in c libraries it helped us in making time dynamic with interrupt way as it checks if you pressed and key or not without pausing the code this lead to update time every second
* **Clock() function :-** it helped us to calculate time
* **AI functions :-** in this function we have three parts . The first part check if there is any square can be close (there are three lines) and set the PC to play this movement which will close the square . The second part we check every square if there is one line or there is not any line we set the PC to play in any position in this square (In the code the loop that check this part I have varied its limits in 2\*2 I have made the limits random to play in the empty square or the square which has one line randomly but in 5\*5 after two game you will understand how the PC will play in his first movements as the limits without random [it is obvious in code] . The third part is to play randomly if all above checks does not exist
* **Mode function (as a example [p1Vsp2]) :-** these functions contains all above functions , they contain the loop that control game’s switch and finishing the game

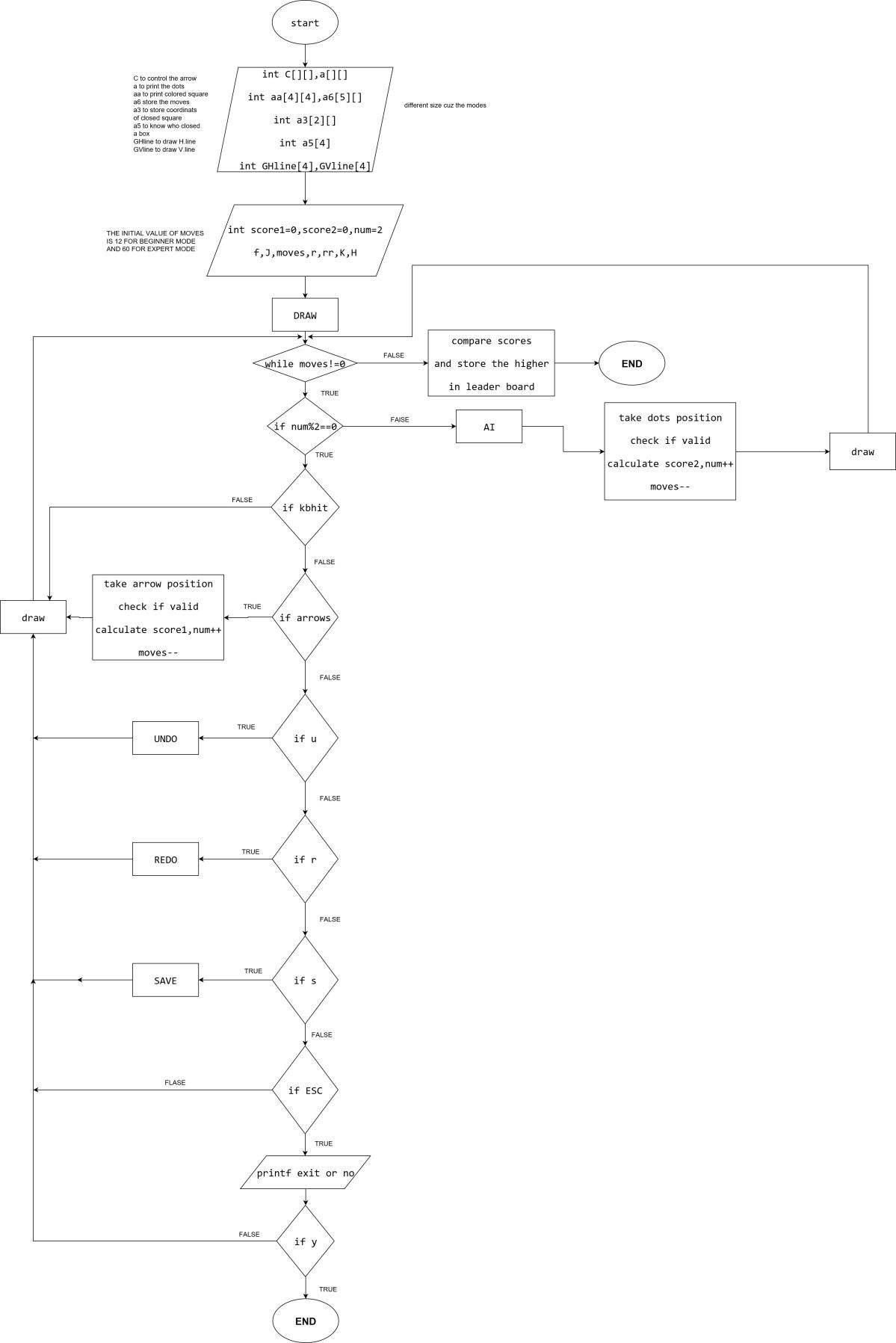
\*\***Header files that we used**

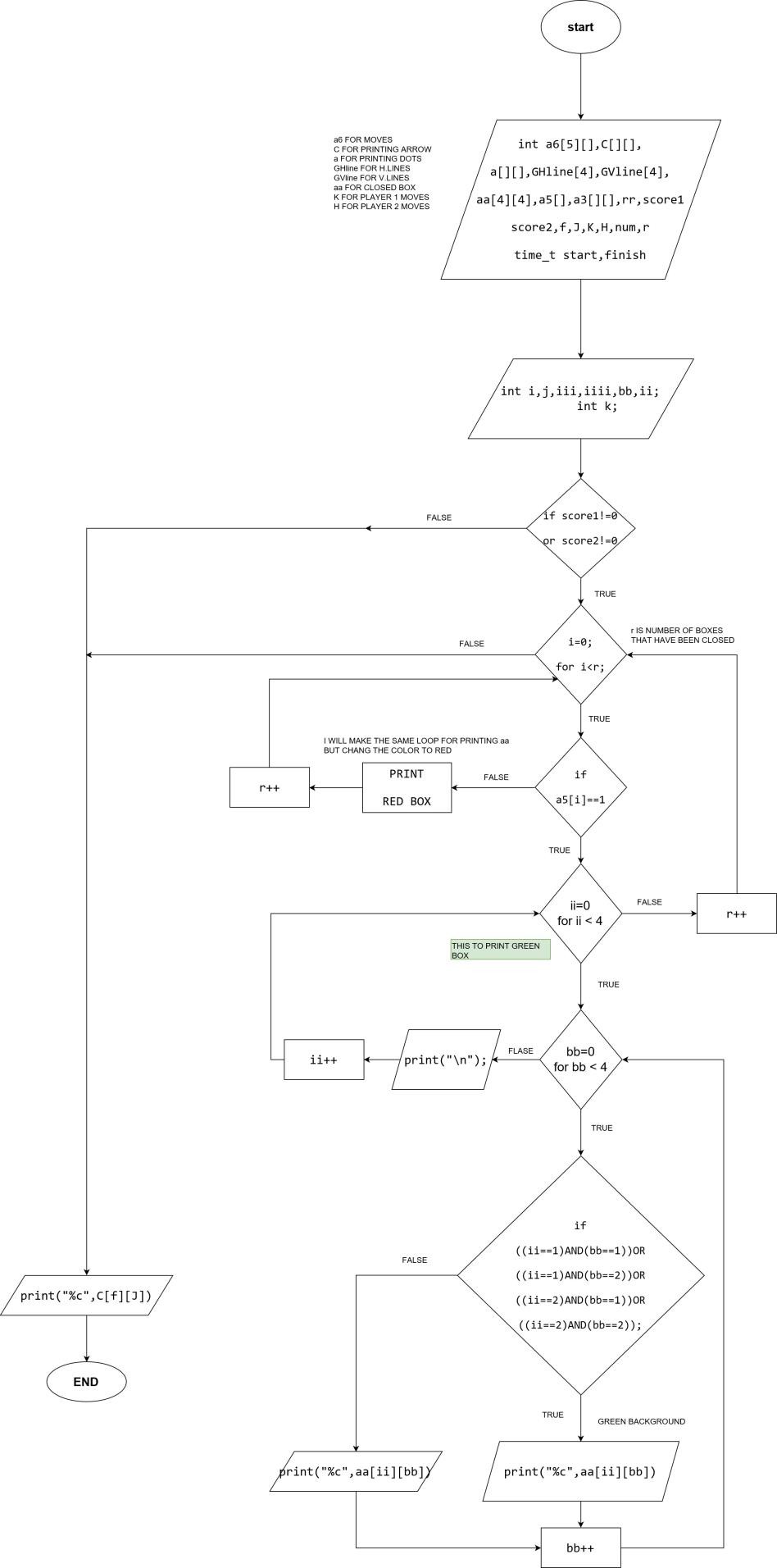
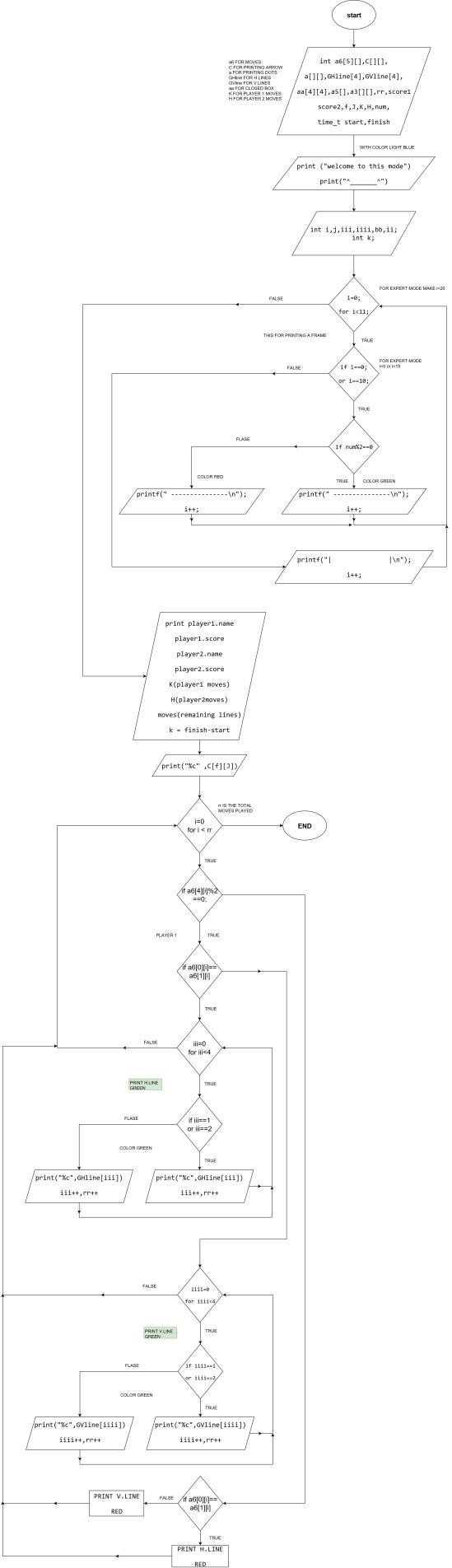
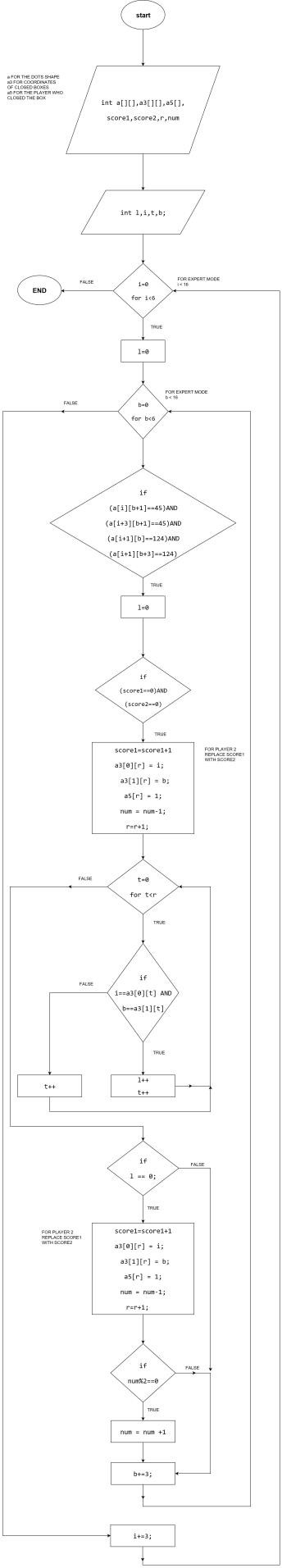
* **Header files that we made :-**
  + function.h >> it is used to use the functions of 1P VS 1P (2\*2 mode)
  + functionspc.h >> it is used to use the functions of 1P VS PC (2\*2 mode)
  + function5.h >> it is used to use the functions of 1P VS 1P (5\*5 mode)
  + functionpc\_5.h >> it is used to use the functions of 1P VS PC (5\*5 mode)
* **Header files in C** :-
  + **stdio.h**
  + **stdlib.h**
  + **math.h**
  + **string.h**
  + **ctype.h**
  + **windows.h**
  + **conio.h**
  + **time.h**

FLOWCHARTS AND PSEUDO CODE :-

* flowcharts :-

1P VS 2P

1P VS PC

* Print colored squares
* Printing colored lines
* Calculating score
* Pseudo code
  + For undo **:-**

set a6[][],int a[][],int a3[][],a5[],r,score1,score2,flag,f2,moves, K, H,rr,undo,k,v,m,u;

set i,b,l=0,P=0;

if (rr!=0){ //to check if there is movements or not if (a6[4][rr-1]%2==0){

K=K-1; //to decrease movement

}else{

H=H-1; //to decrease movement

}end else

k=a6[0][rr-1]; //Start removing drew lines v=a6[1][rr-1];

u=a6[2][rr-1];

m=a6[3][rr-1];

flag=2;

rr=rr-1; //In order not to print that removed line with color BUMP undo; //to know how many times can i do redo if(k==v){ //if rows were equal

BUMP moves;

a[k][m]=a[v][u]=42; //ascii for asteriks for(i=k;i<(k+1);i++){

for(b=u+1;b<(u+3);b++){ a[i][b]=32; //ascii for space

}end for

}end for

}else if(u==m){ //if columns were equal BUMP moves;

a[k][m]=a[v][u]=42;

for(b=u;b<(u+1);b++){

for(i=k+1;i<(k+3);i++){

a[i][b]=32;

}end for

}end for

}end else if

for(i=0;i<6;i+=3){ //these loops to check if there is a square opened or not

l=0;

for(b=0;b<6;b+=3){

if((a[i][b+1]==45)&&(a[i+3][b+1]==45)&&(a[i+1][b]==124)&&(a[i+1][b+3]=

=124)){

l=0;

}else {

for (P=0; P<4; P++){

if((a3[0][P]==i)&&(a3[1][P]==b)){ //removing coordinates a3[0][P]=2;

a3[1][P]=2;

BUMP f2; //we use it if there are two squares had opened at the same time to give the turn correctly

r=r-1;

if (a5[P]==1){

score1=score1-1;

}else { score2=score2-1;

}end else

}end for

}end else

}end if

}end for

}end for

}else {

OUTPUT Invalid; delay(500);

cls;

}

}

* + **for redo**

set a6[][],rr,undo,k,v,m,u,a[][],a3[][],a5[],r,score1,score2,flag,f2,moves,K,H

set i,b,t,l=0; flag=2;

if (undo==0){ //to check if there is an undo or not gotoxy(0,13);

OUTPU "You can't do Redo without Undo";

}else { //returning the value of moves of two players if (a6[4][rr]%2==0){

BUMP K;

}else{ BUMP H;

}end else

k=a6[0][rr]; //starting returning removed line v=a6[1][rr];

u=a6[2][rr];

m=a6[3][rr]; BUMP rr;

undo=undo-1; //in order not to make any more redo if(k==v){

moves=moves-1;

a[k][m]=a[v][u]=35; //ascii for asteriks for(i=k;i<(k+1);i++){

for(b=u+1;b<(u+3);b++){ a[i][b]=45; //ascii for line

}end for

}end for

}else if(u==m){ moves=moves-1; a[k][m]=a[v][u]=35;

for(b=u;b<(u+1);b++){

for(i=k+1;i<(k+3);i++){ a[i][b]=124; //ascii for line

}end for

}

}

for(i=0;i<6;i+=3){ //this loop for checking if there is any square had closed again

l=0;

for(b=0;b<6;b+=3){

if((a[i][b+1]==45)&&(a[i+3][b+1]==45)&&(a[i+1][b]==124)&&(a[i+1][b+3]=

=124)){

l=0;

if ((score1==0)&&(score2==0)){ a3[0][r]=i;

a3[1][r]=b;

BUMP f2

if (a5[r]==1){ BUMP score1;

}else {

BUMP score2;

}end else BUMP r;

}else {

for (t=0; t<4; t++){ //checking if closed square has calculating before or not

if ((i==a3[0][t])&&(b==a3[1][t])){ l++;

}end if

}end for

if (l==0){ //it is did not calculating before a3[0][r]=i;

a3[1][r]=b; BUMP f2;

if (a5[r]==1){ BUMP score1;

}else {

BUMP score2;

}end else BUMP r;

}end if

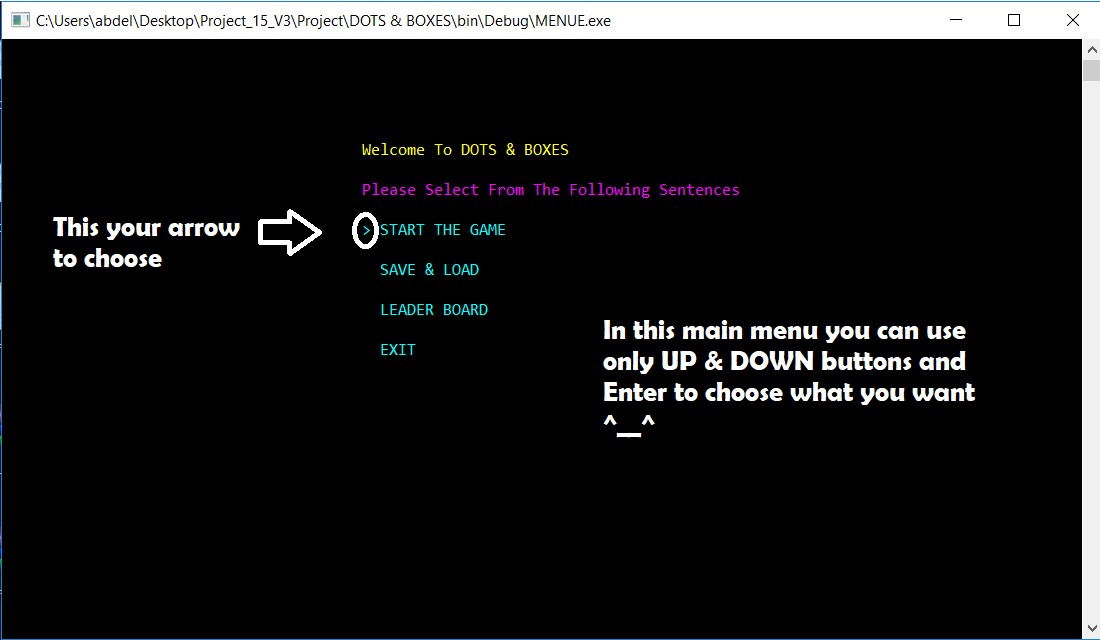
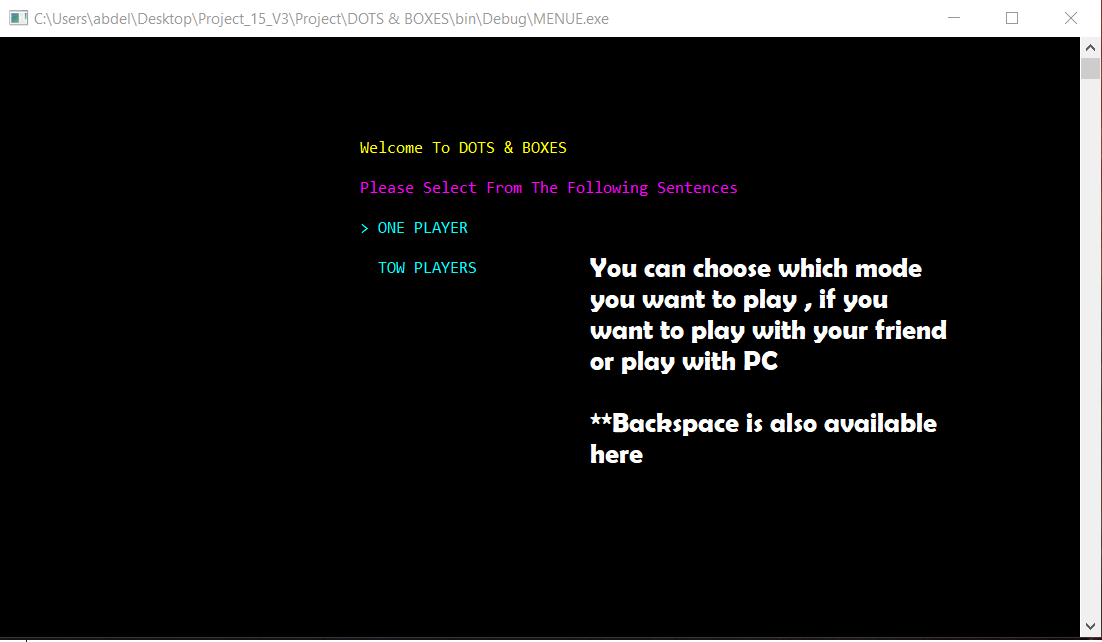
}end if

}end for

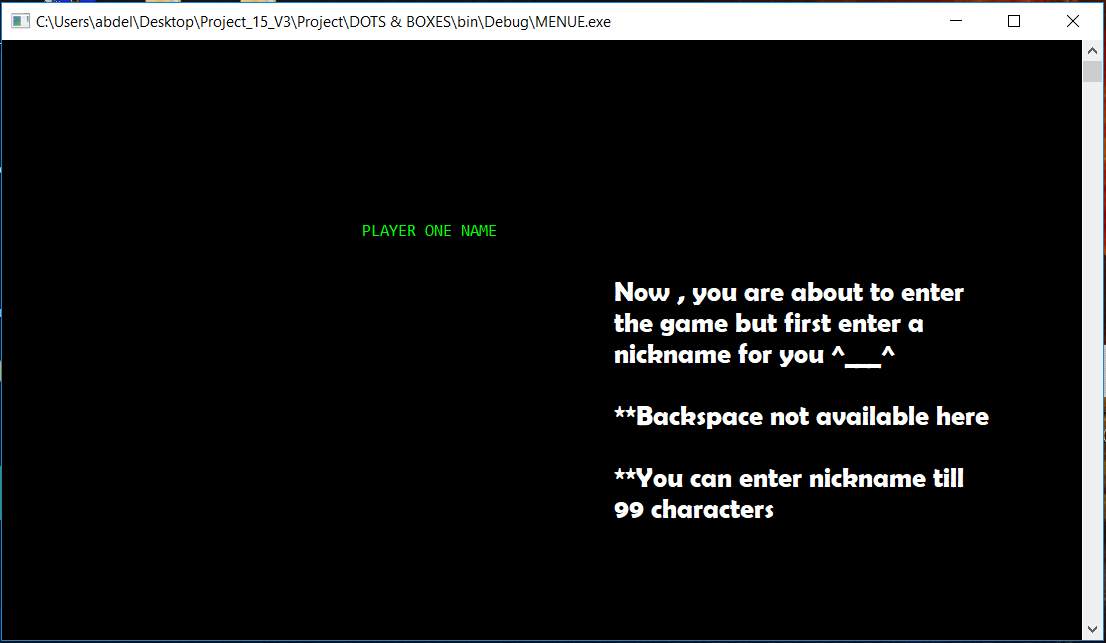
}end for

# USER MANUAL :-

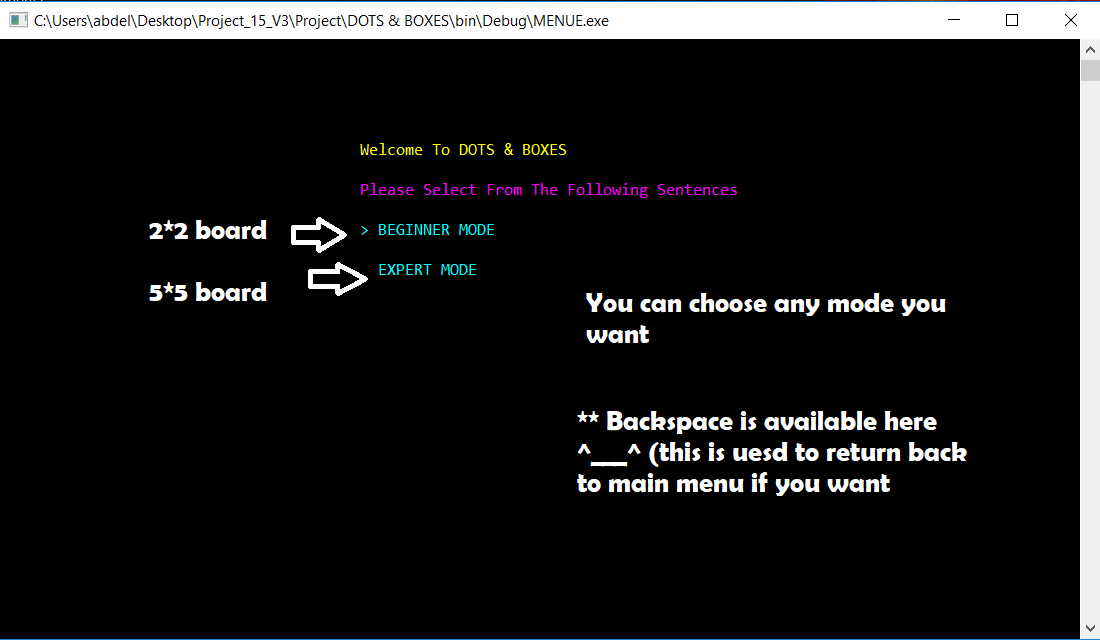
* **How to enter a new game :-**



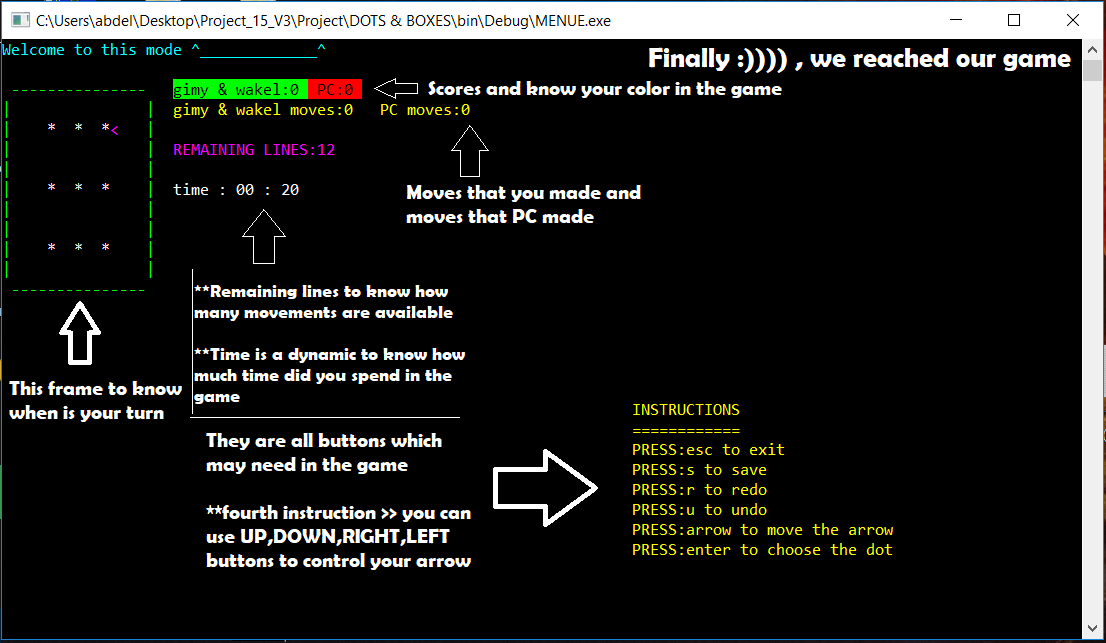
Figure(1)



Figure(2)



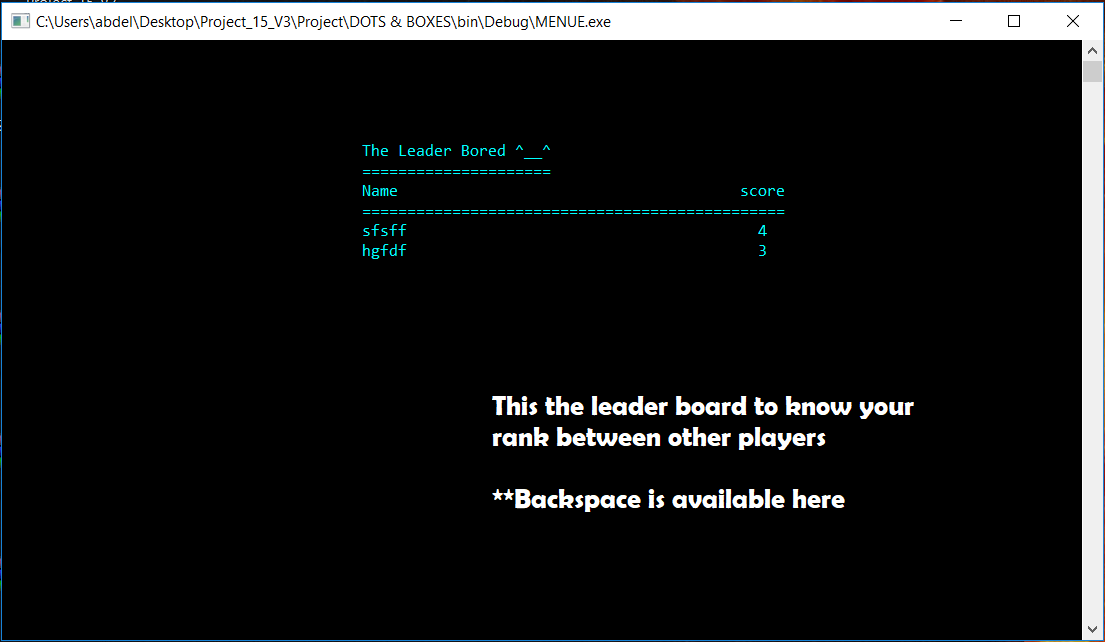
Figure(3)



Figure(4)

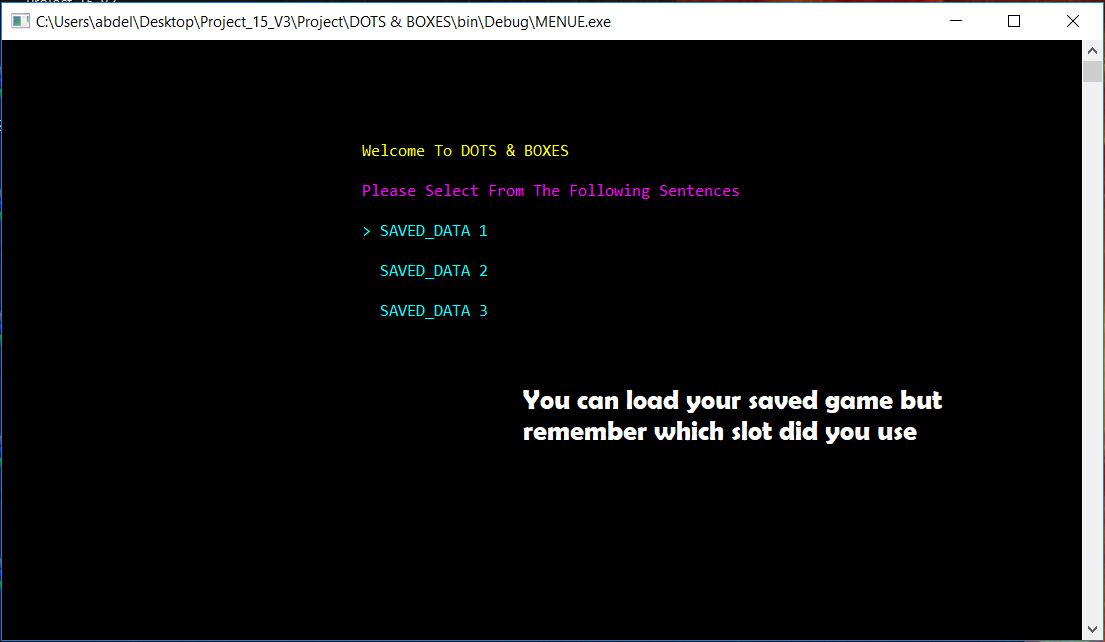
\*\***Note** :- this is tutorial is for one mode but you can do the same things to enter Other modes

* **How to open the leader board :-**

From the main menu in fig.(1) , choose Leader board option then you will have the fig(5)

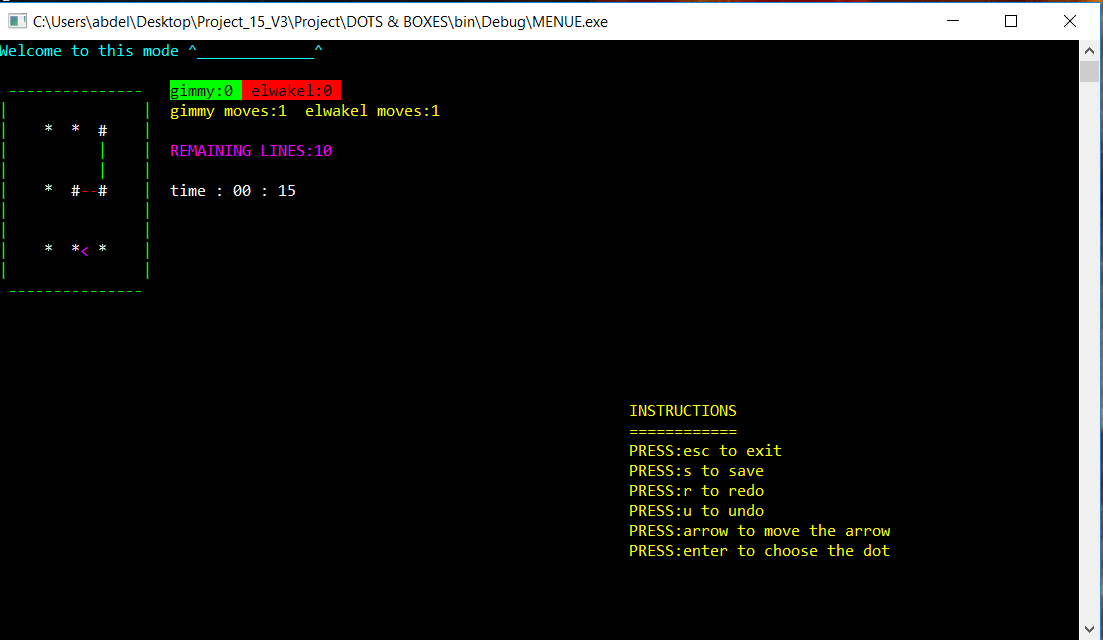
# Figure(5)

* **How to load a saved game :**-

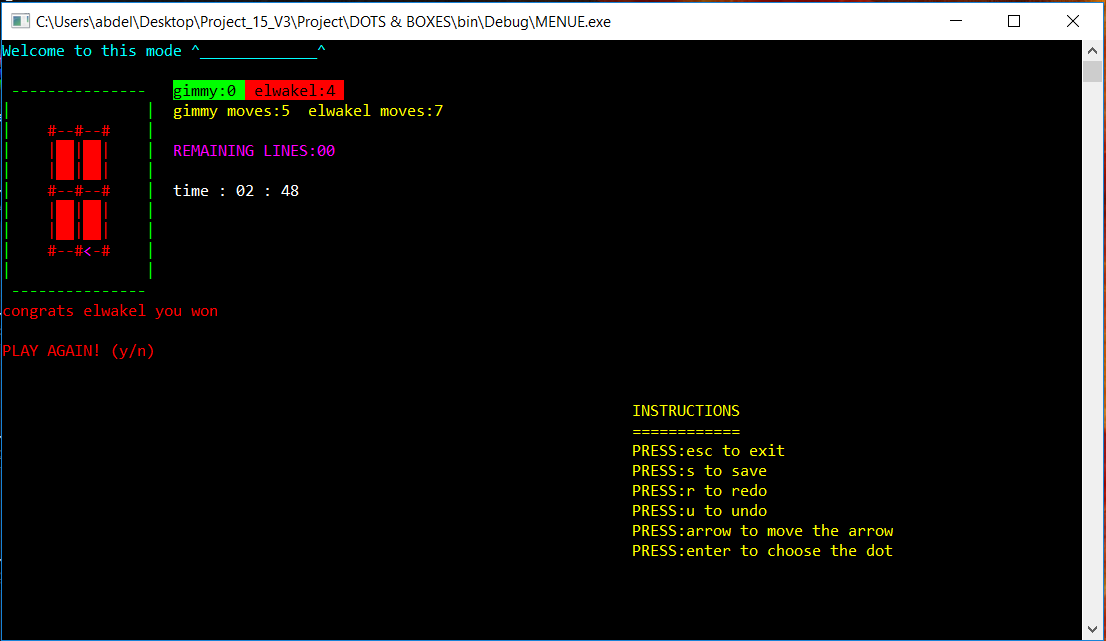
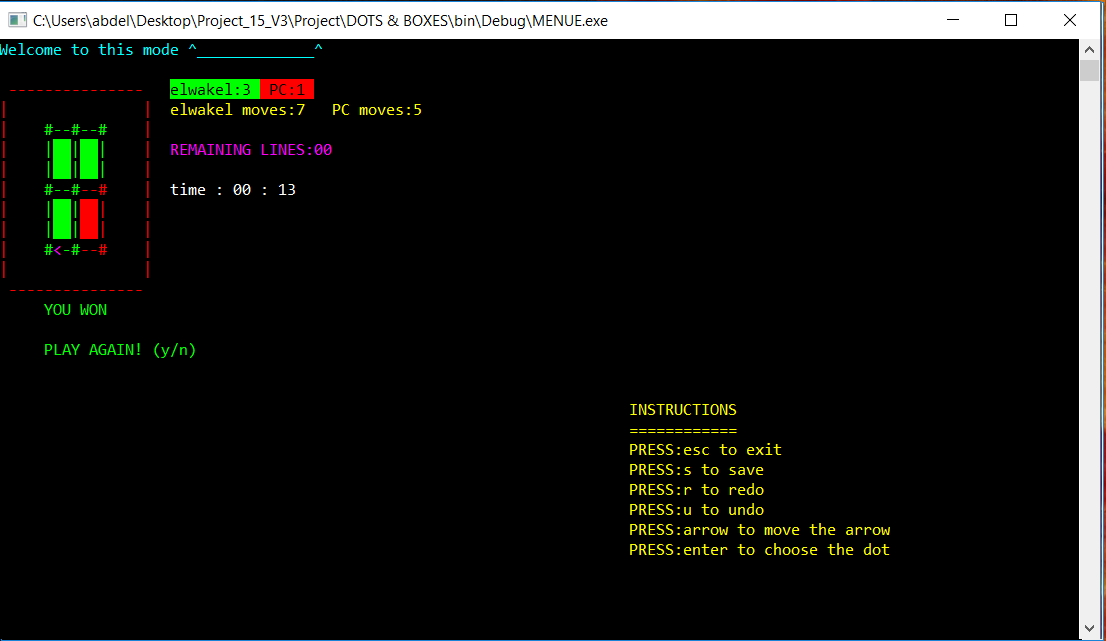
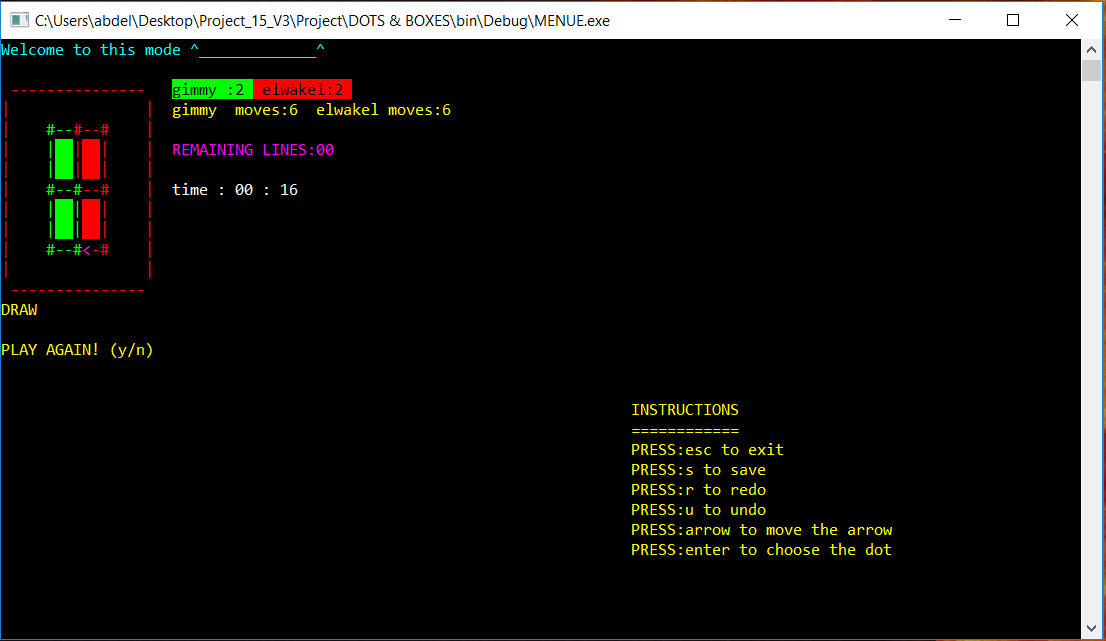
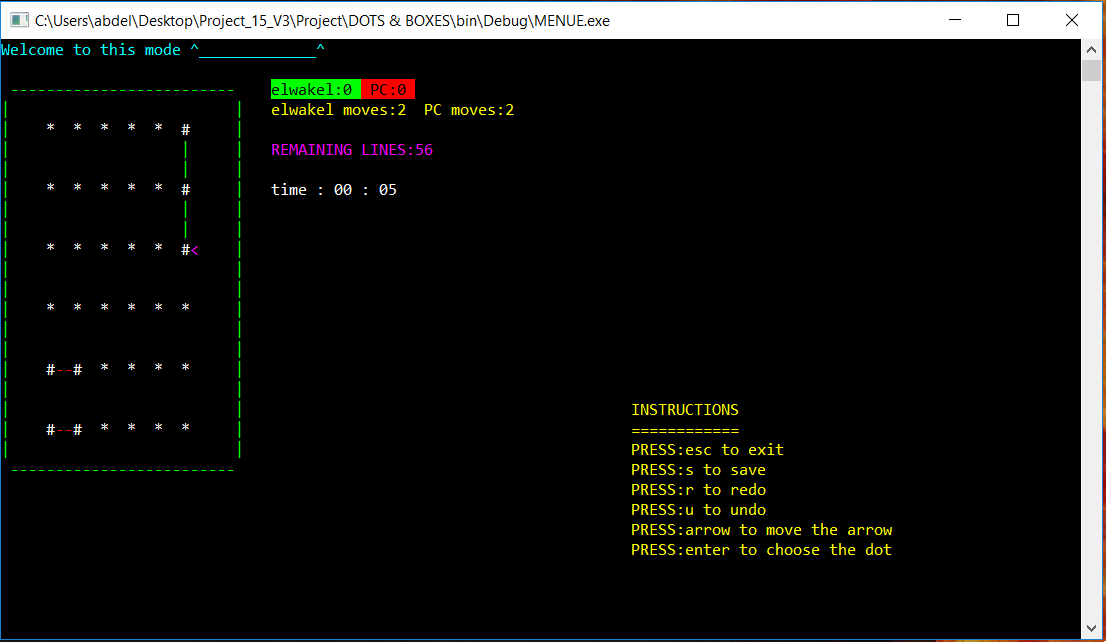
From fig(1) choose SAVE & LOAD option then you have the fig (6)

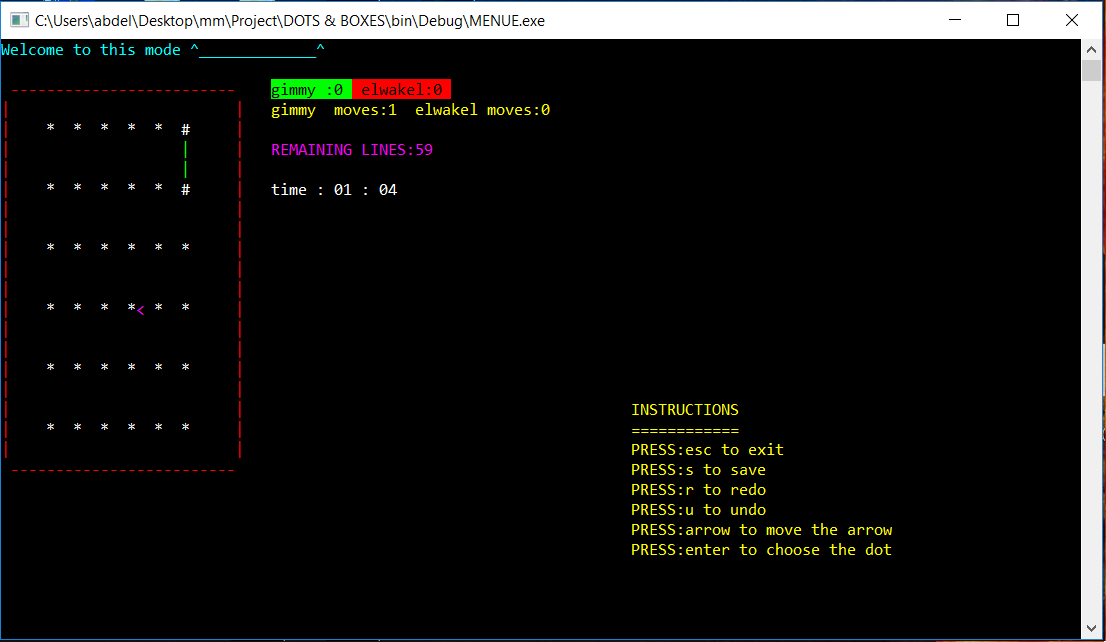
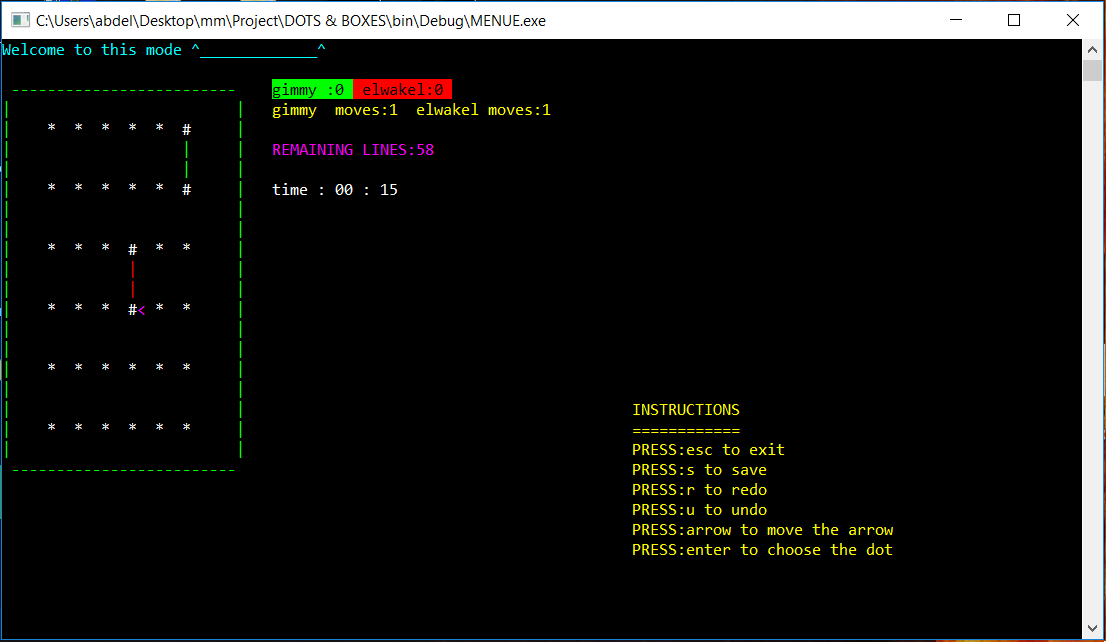
# Figure(6)

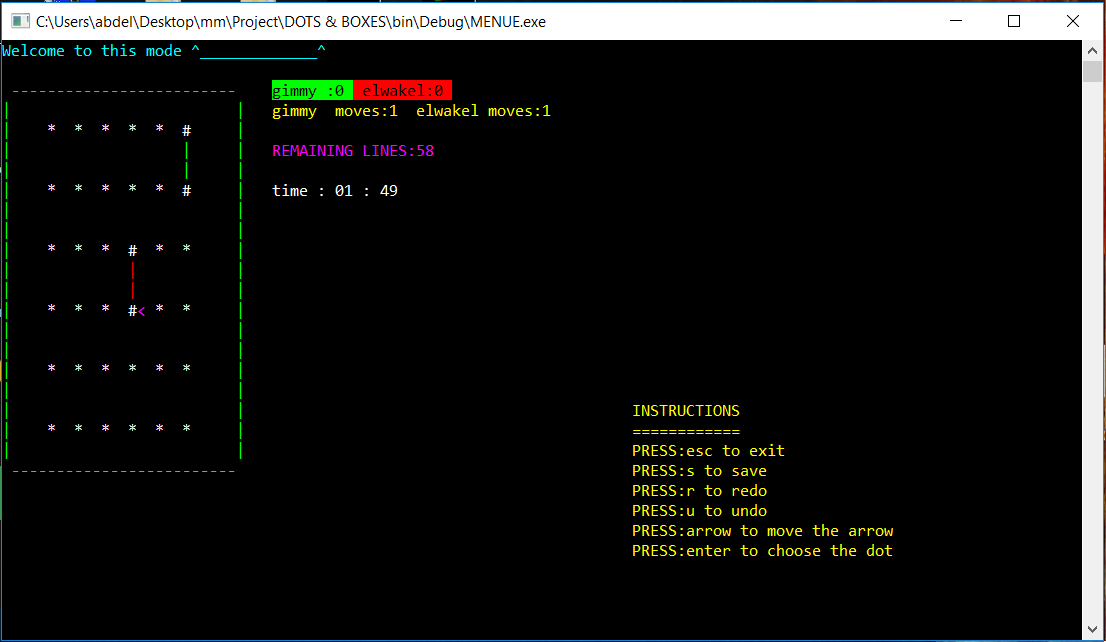
# SAMPLE RUNS :-

* Playing two lines in 2\*2 mode P1 VS P2
* One player closed a square



* One of the players has won
* Beating PC
* Draw between two players
* Playing with PC in 5\*5 board
* Playing two players mode and doing UNDO & REDO





# References :-

* C programming : A modern Approach (as most of our algorithm depended on which had studied this semester)
* Stack Overflow’s answers (they helped us in making our time dynamic , helped us in printing things in colors) [link : <https://stackoverflow.com/questions>]
* <https://www.tutorialspoint.com/cprogramming/>
* <https://www.cprogramming.com/>
* <http://www.codeincodeblock.com/>