DOTS &BOXES

CONSOLE-BASED GAME

By

Mohamed elnashar & tarek Mohamed

CSED21

1) Game Description

Dots and Boxes is a game for two players (or more). It was first published in the 19th century by Edouard Lucas, who called it originally la pipopipette. Starting with an empty grid of dots, two players take turns adding a single horizontal or vertical line between two unjointed adjacent dots. The player who completes the fourth side of a 1x1 box (or groups of one or more adjacent boxes) earns one point (s) and takes another turn.

2)Design overview

- This project was totally designed, debugged and tested by Mohamed Alnashar and Tarek Mohamed.

- The code is divided into various functions. each function has its own purpose and is as independent as possible from the rest of the code making the code divided into smaller semi-independent units

-There is a tree menu with different choices make the user choose his favorite game type.

3)Features

1-Player can choose any size to play the game with.

2-Player can choose his/her color to play.

3-There are sound effects (errors and completing boxes).

4-Music playing during the game.

5-Difficulty of computer (greedy or random).

6-After loading you can undo and redo pre-existing plays.

4)Assumptions

-The game will end when number of remaining dots end.

-Player can't play in a pre-existing place.

-When the player chooses his/her name it must be smaller than 15 characters.

-Player can't play in diagonal play and can't join between adjacent dots.

-Player can't enter unexisted row or column.

-Computer won't save in the top ten menu.

-Player must choose his/her color from the menu.

-Player mustn't load an empty file.

-Player can't enter character instead of integer.

5)Data Structure

-To ensure the best performance and to reduce coding complexity, one the basic and simple data types were used.

For determining each element in the game grid only a two-dimensional char array was used to define each element and further simplify the coding.

-To save the score and moves and the name of each player we used structures and we used it to compare between two scores and determined which player won.

-We used structure arrays to save the information of each player and to print the top ten players and to print the rank of the winner.

-Another clear data structure of simple types easily understood by the developer ton understand such as: int, double and char.

-We used various functions for different purpose (undo, redo, computer, colors, file and player).

6)Functions

1)MENU():

1. Parameters: NULL.

2. Return Type: void

3. Description: this function takes inputs from user to decide which type of game he want to play like:

1-start game->

(1) Beginner-> [human vs human or human vs computer]

(2) expert-> [human vs human or human vs computer]

(3) choose the size [human vs human or human vs computer]

2-help

3-load game

4-top ten

5-exit

2)SAVE:

1. Parameters: row, column, char array[row][column].

2. Return Type: void

3. Description: Saves the current game and saves every parameter needed to resume the game afterwards.

3)Player:

1. Parameters: row, column, char array[row][column].

2. Return Type: void

3. Description: makes the player input the two dots to play between them and calculate his/her moves and score.

4)Colour:

1. Parameters: NULL.

2. Return Type: void

3. Description: makes the player choose which color he wants to play with.

5)TopTen:

1. Parameters: NULL.

2. Return Type: void

3. Description: arranges the players and print the top ten players with the highest 10 scores.

6)UNDO:

1. Parameters: row, column, char array[row][column].
2. Return Type: void
3. Description: removes the last played move and give the player another chance to play.

7)REDO:

1. Parameters: row, column, char array[row][column].
2. Return Type: void
3. Description: returns the last removed play that the player has just undo it.

8)PRINT:

1. Parameters: row, column, char array[row][column].
2. Return Type: void
3. Description: displays the player interface containing all the data of each player, time, remaining dots and whose turn to play.

9)Computer:

1. Parameters: row, column, char array[row][column].
2. Return Type: void
3. Description: a computer plays against a player in random or greedy mode with AI to make the game more challenging and fun.

10)Last Menu:

1. Parameters: NULL.
2. Return Type: void
3. Description: prints the winner player and his/her score and save this information in file to print his/her rank.

7) User Manual:

-In this game you will enter the number of row and column

of the first point then the row and the column of the

second point you want to join between.

-if you want to choose any choices you must input the (number beside this choice).

- Enter

1) to undo: - row1(1111)

  - column1(any int)

  -row2(any int)

  -column2(any int)

2) to redo: - row1(2222)

  - column1(any int)

  -row2(any int)

  -column2(any int)

3) to save: - row1(3333)

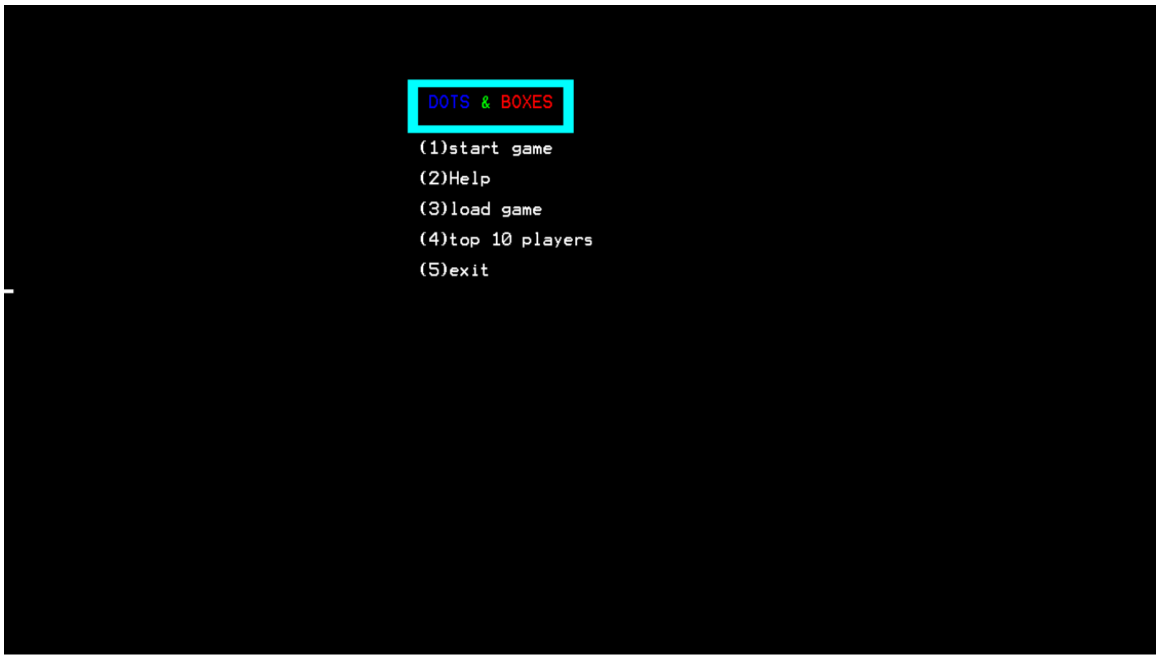
  - column1(any int)

  -row2(any int)

  -column2(any int)

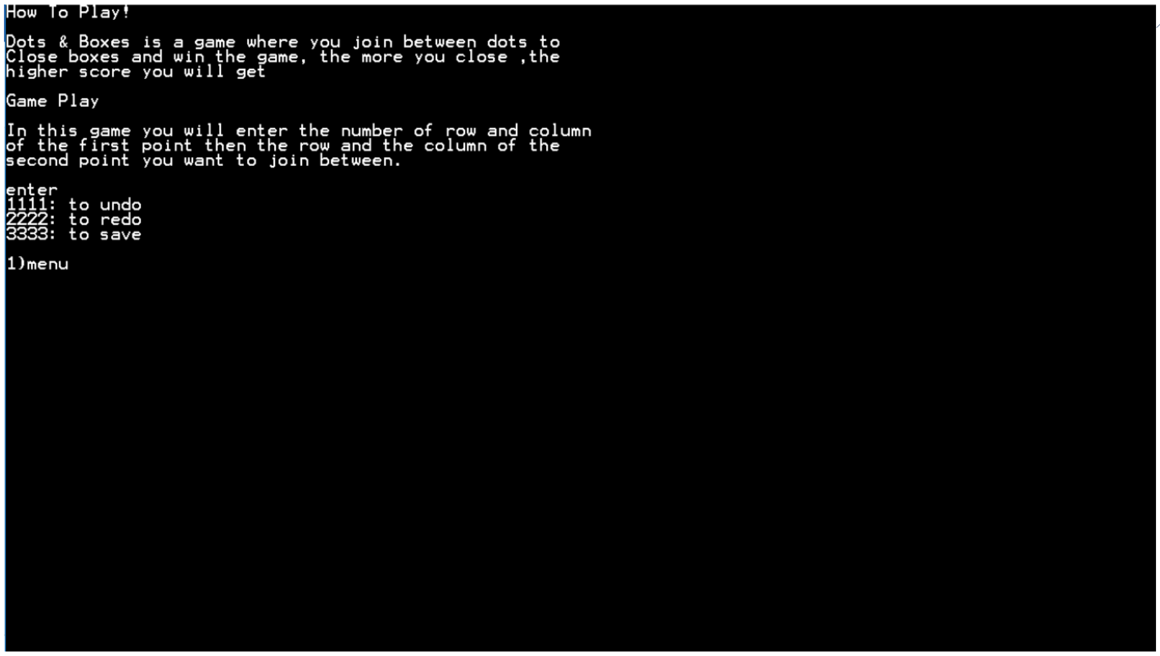
SCREENSHOTS OF THE GAME

1)Starting menu



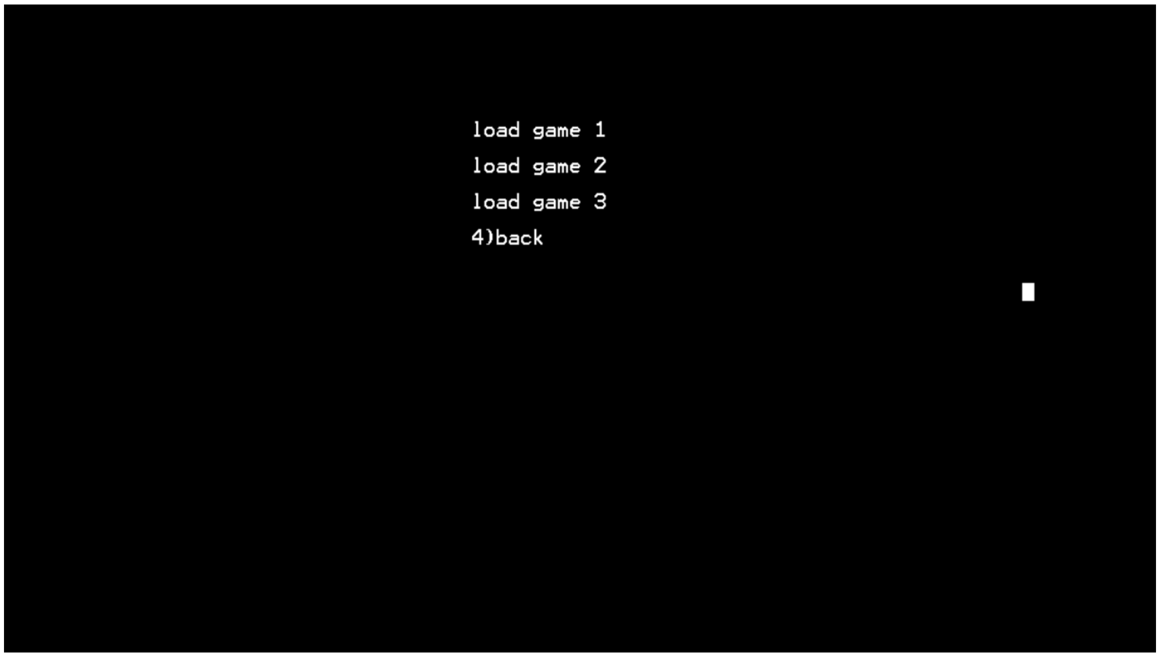
```
DOTS & BOXES
(1)start game
(2)Help
(3)load game
(4)top 10 players
(5)exit
```

2)How to play (player guide)



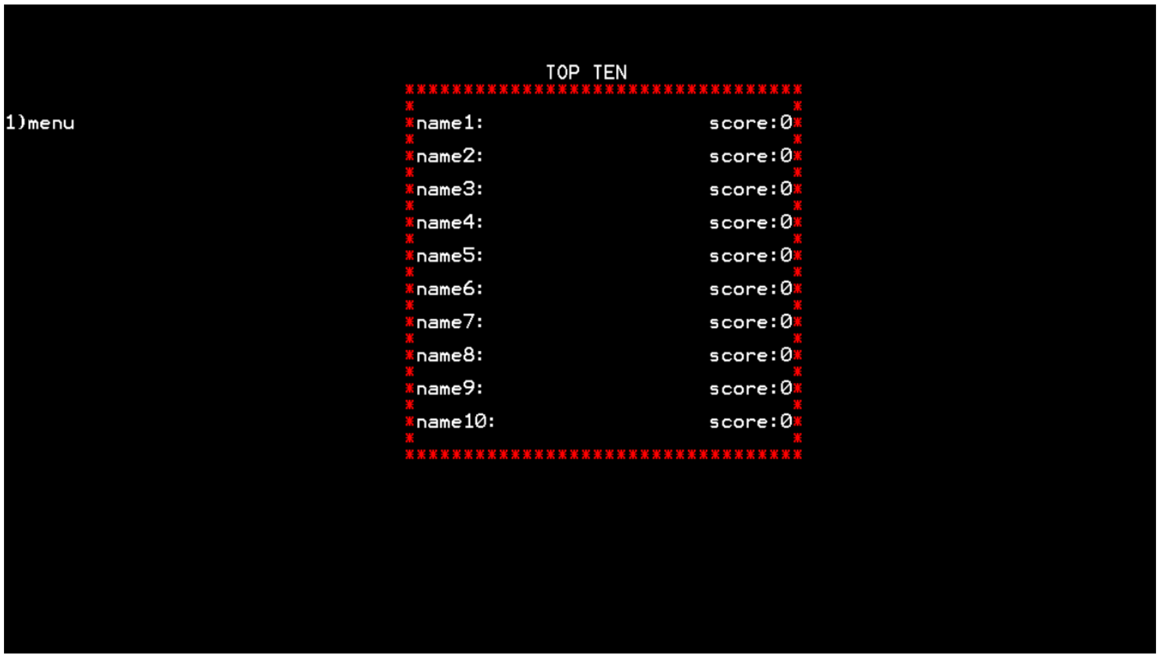
```
How To Play!

Dots & Boxes is a game where you join between dots to
Close boxes and win the game, the more you close ,the
higher score you will get

Game Play

In this game you will enter the number of row and column
of the first point then the row and the column of the
second point you want to join between.

enter
1111: to undo
2222: to redo
3333: to save

1)menu
```

3)Load menu



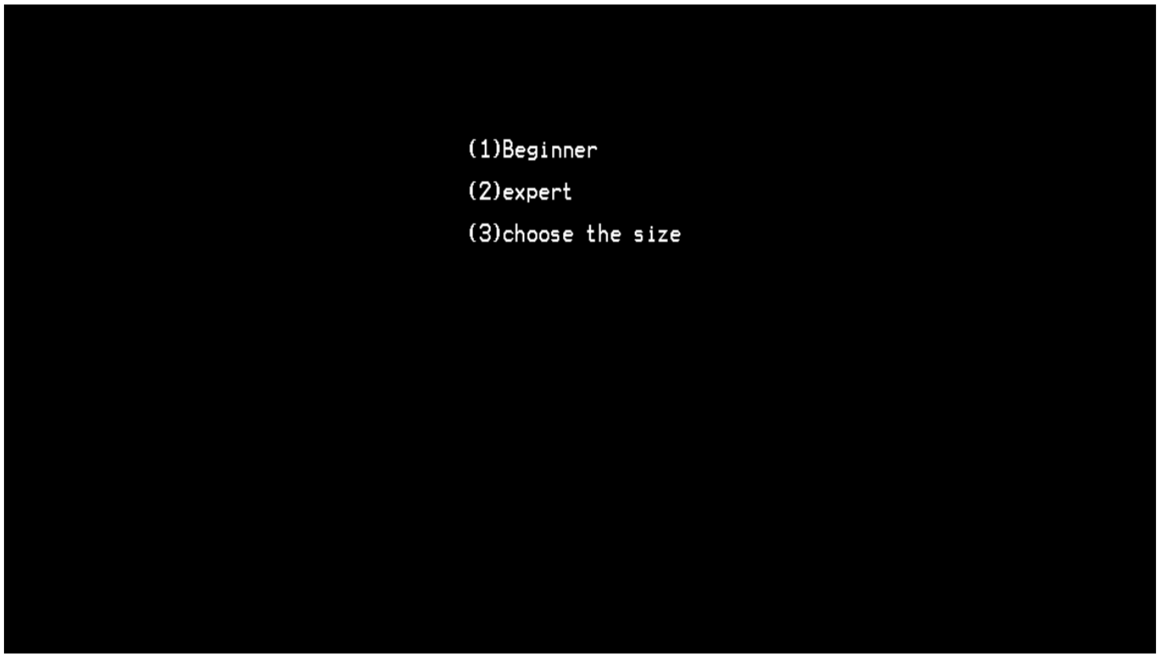
```
load game 1
load game 2
load game 3
4)back
```

4)Leaderboards menu



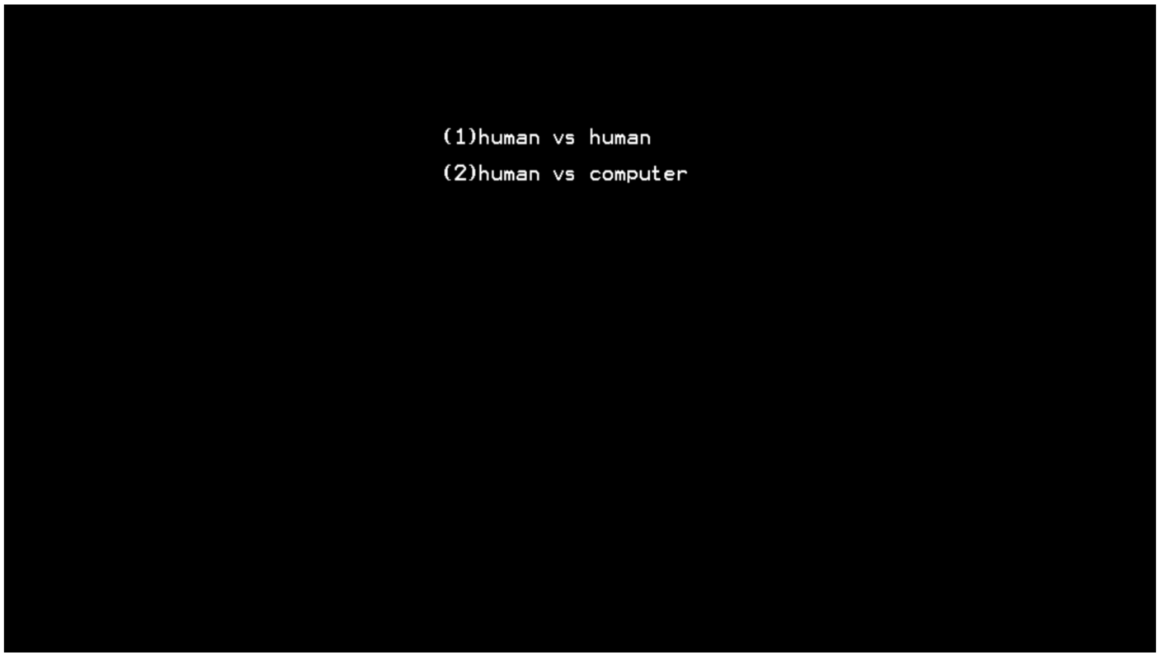
```
                              TOP TEN
                    ********************************
1)menu              *name1:                  score:0*
                    *name2:                  score:0*
                    *name3:                  score:0*
                    *name4:                  score:0*
                    *name5:                  score:0*
                    *name6:                  score:0*
                    *name7:                  score:0*
                    *name8:                  score:0*
                    *name9:                  score:0*
                    *name10:                 score:0*
                    ********************************
```

5)Choose your own type (beginner 2x2, expert 5x5, or any size you want).



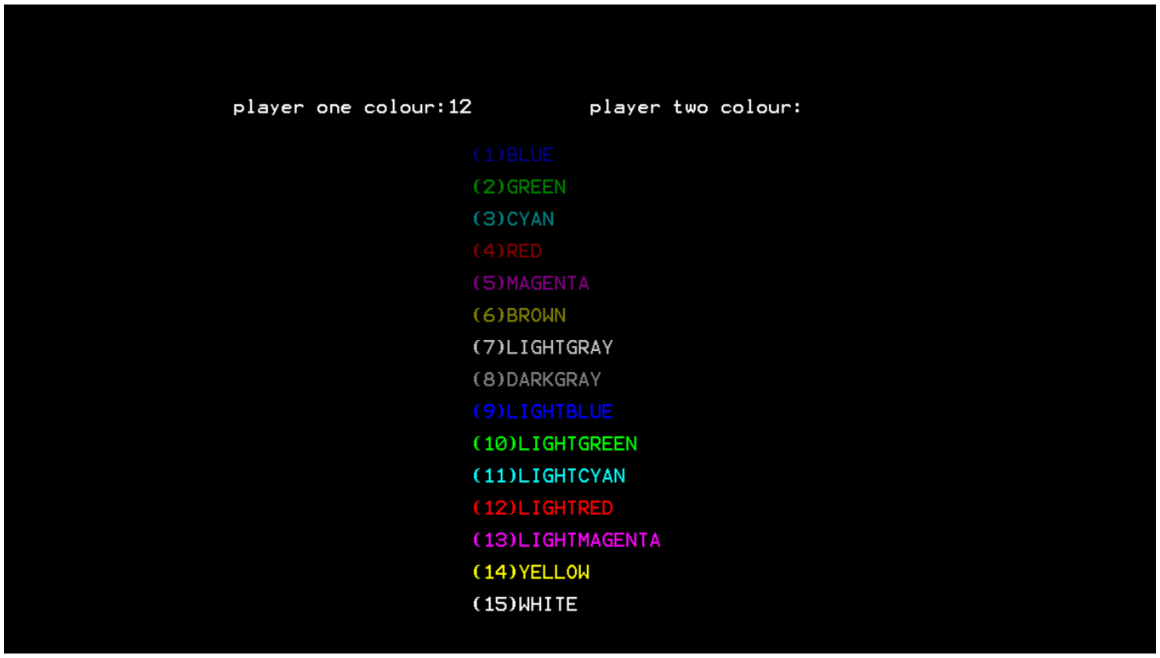
```
(1)Beginner
(2)expert
(3)choose the size
```

6)Choose the mode to play against someone or computer



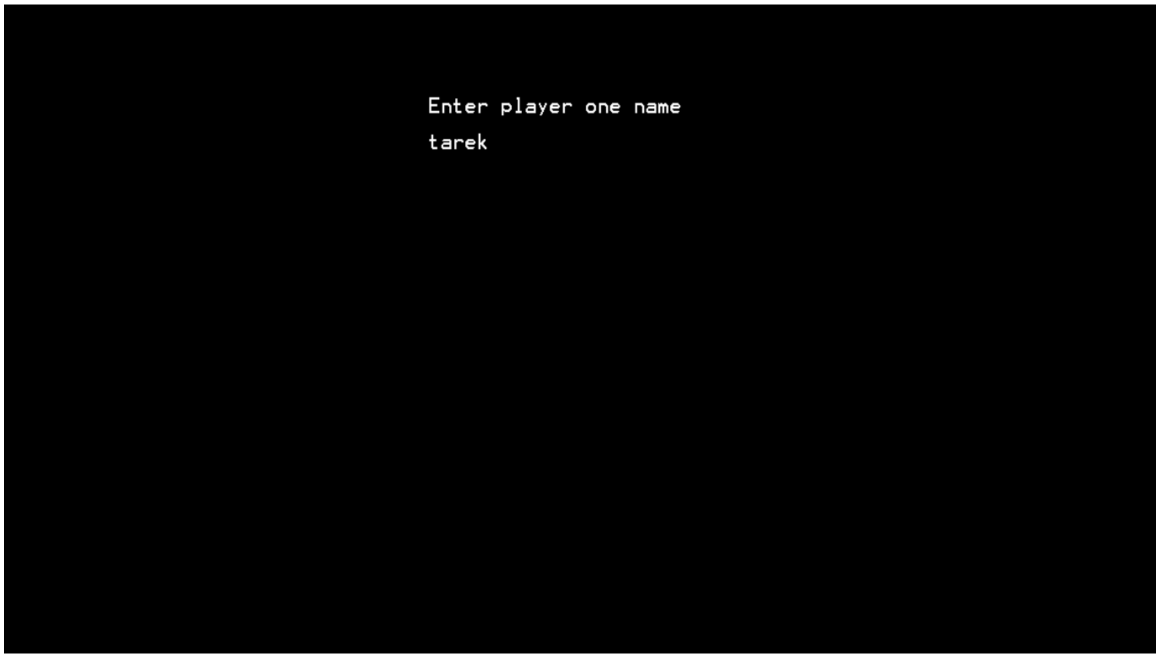
```
(1)human vs human
(2)human vs computer
```

7)Players choose their own colors to play with



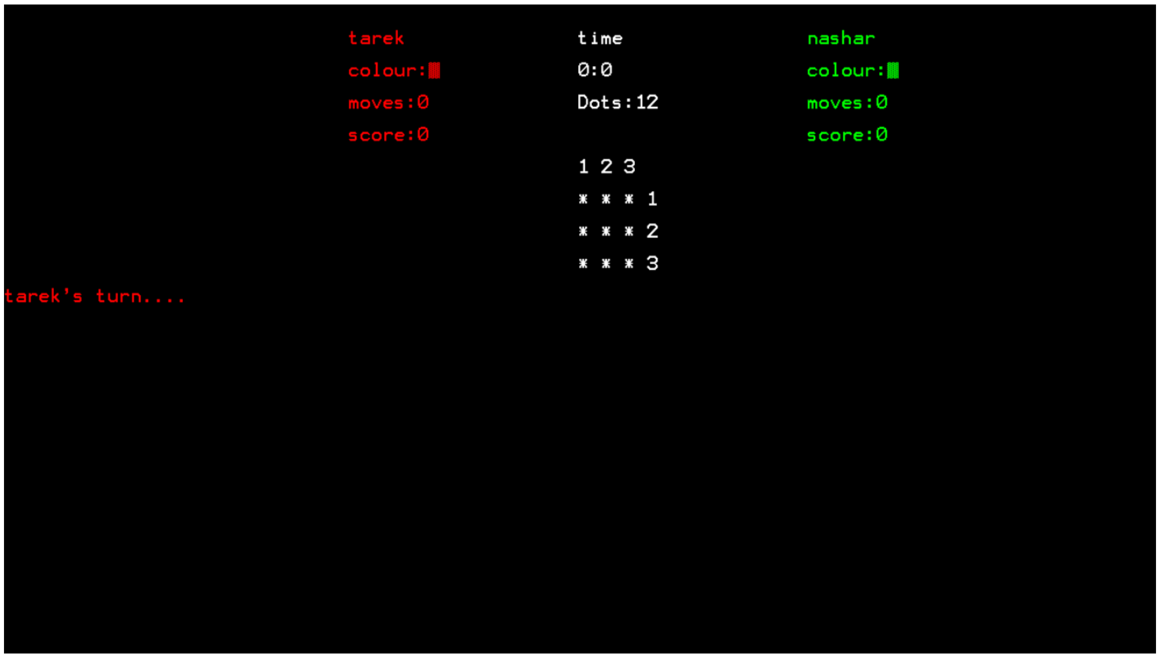
```
player one colour:12          player two colour:

                 (1)BLUE
                 (2)GREEN
                 (3)CYAN
                 (4)RED
                 (5)MAGENTA
                 (6)BROWN
                 (7)LIGHTGRAY
                 (8)DARKGRAY
                 (9)LIGHTBLUE
                 (10)LIGHTGREEN
                 (11)LIGHTCYAN
                 (12)LIGHTRED
                 (13)LIGHTMAGENTA
                 (14)YELLOW
                 (15)WHITE
```

8)Take the player's name one by one



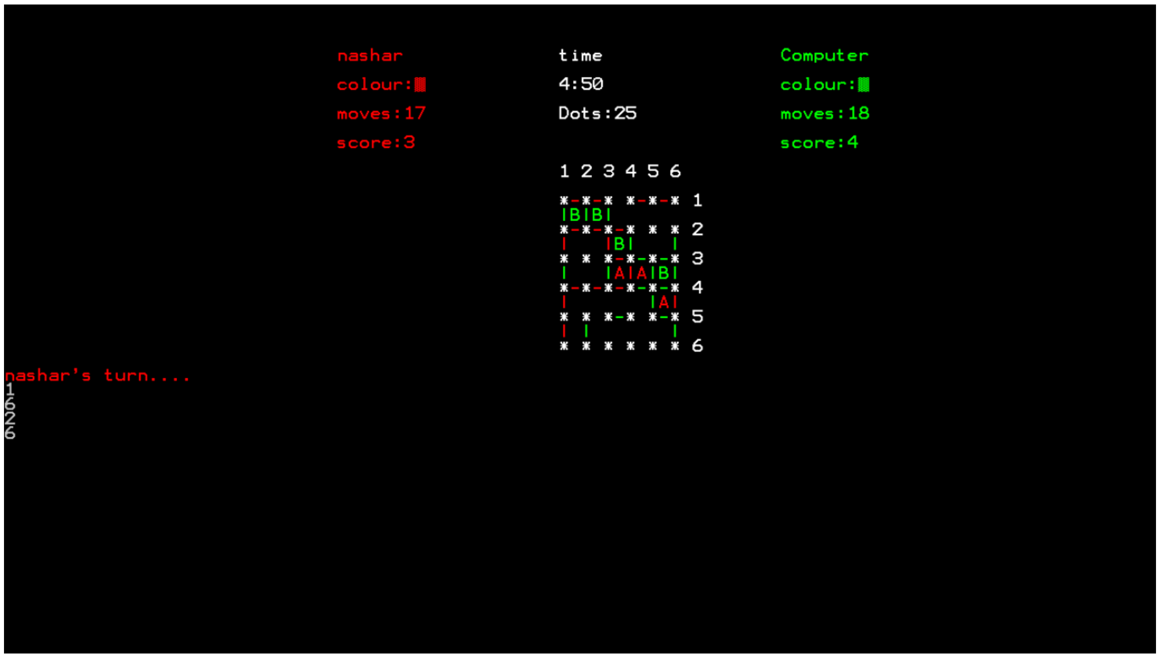
```
Enter player one name
tarek
```

9)Player interface at starting of a mode



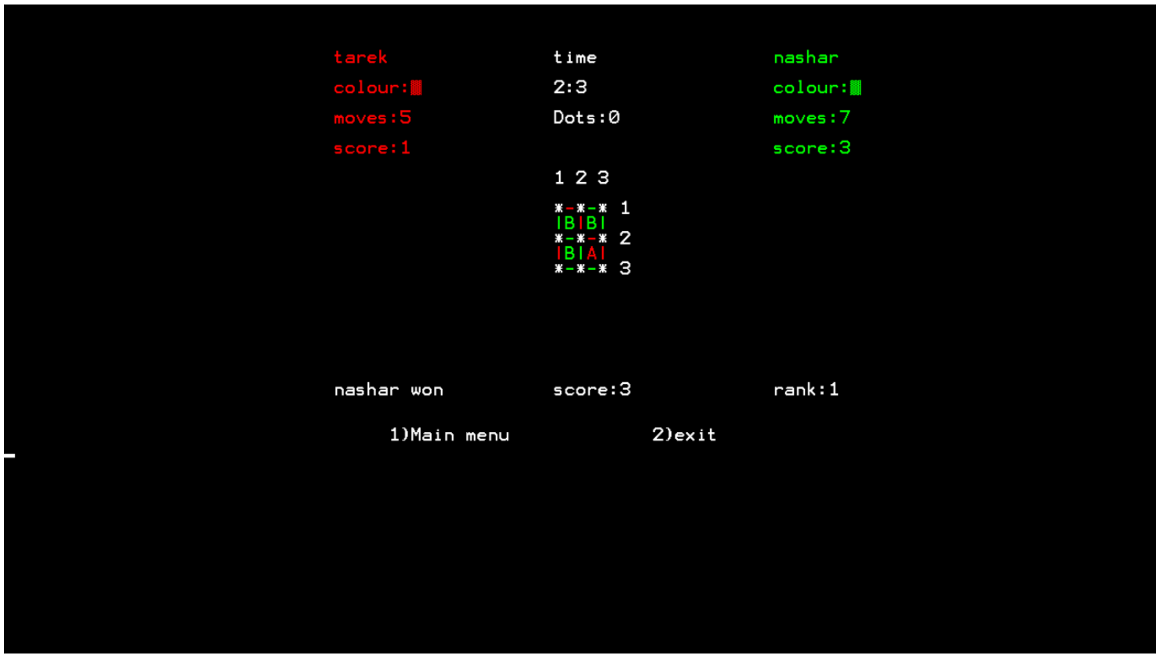
```
                        tarek            time           nashar
                        colour:🟥        0:0            colour:🟩
                        moves:0          Dots:12        moves:0
                        score:0                         score:0

                                         1 2 3
                                         * * * 1
                                         * * * 2
                                         * * * 3
tarek's turn....
```

10)Entering input and closing boxes



```
                    nashar              time                Computer
                    colour:🟥           4:50                colour:🟩
                    moves:17            Dots:25             moves:18
                    score:3                                 score:4
                                        1 2 3 4 5 6
                                        *-*-* *-*-* 1
                                        |B|B|
                                        *-*-*-* * * 2
                                        |   |B|   |
                                        * * *-*-*-* 3
                                        |   |A|A|B|
                                        *-*-*-*-*-* 4
                                        |       |A|
                                        * * *-* *-* 5
                                        | |       |
                                        * * * * * * 6
nashar's turn....
1
6
2
6
```

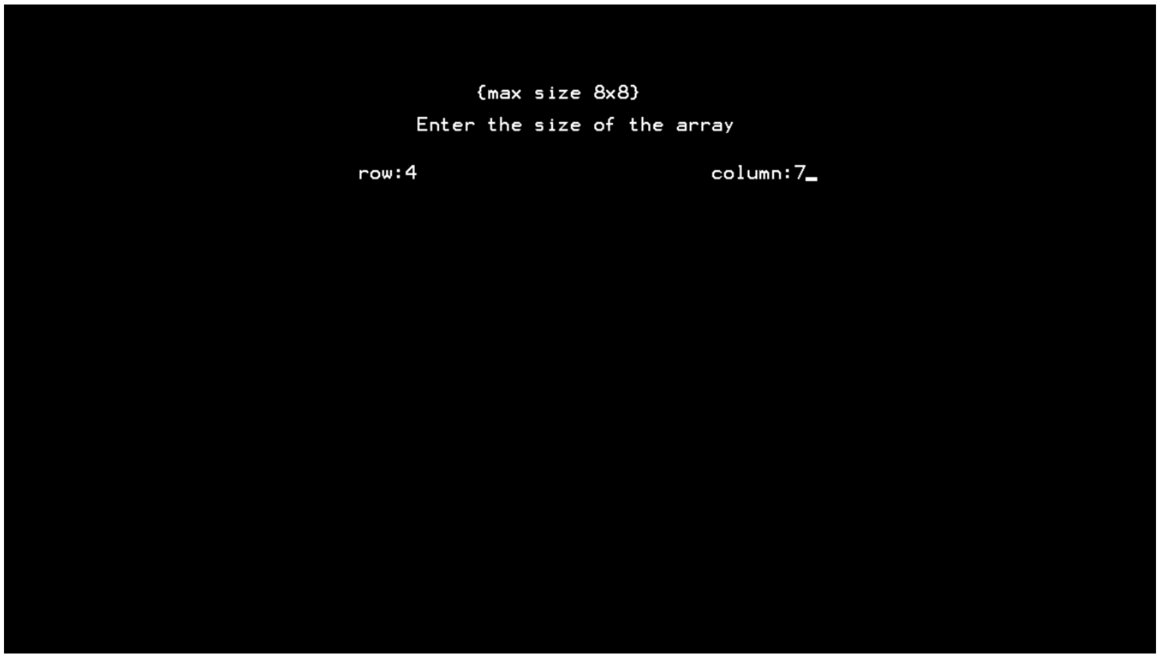
11)Player rank and score is displayed after the game ends



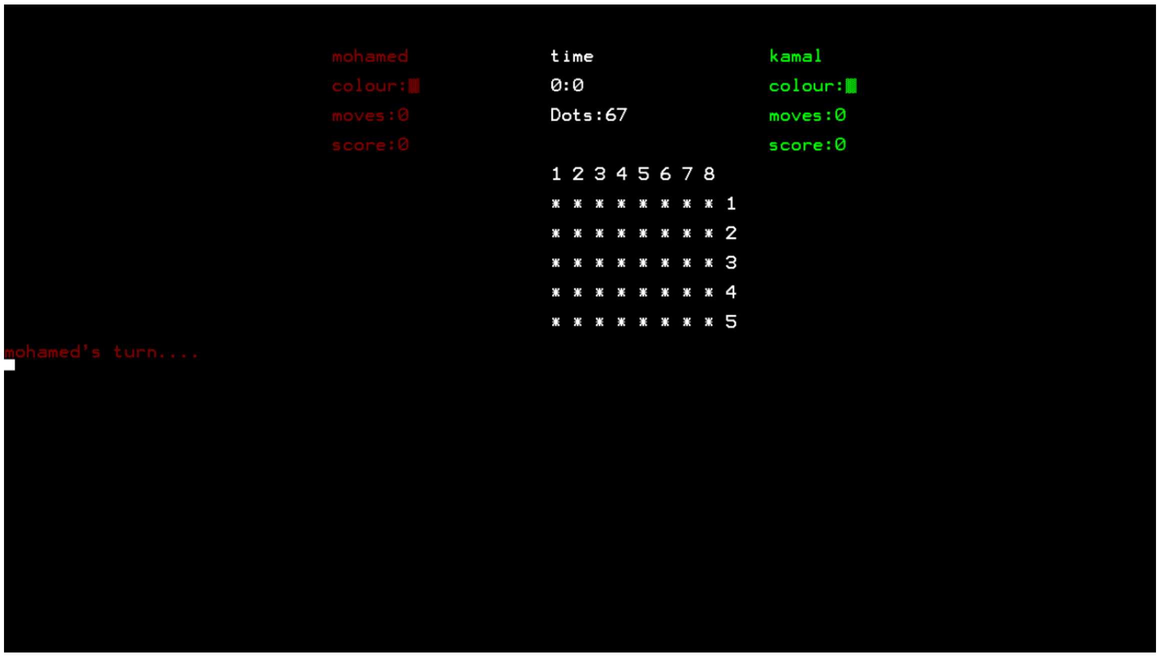
```
                    tarek               time                nashar
                    colour:🟥           2:3                 colour:🟩
                    moves:5             Dots:0              moves:7
                    score:1                                 score:3
                                        1 2 3
                                        *-*-* 1
                                        |B|B|
                                        *-*-* 2
                                        |B|A|
                                        *-*-* 3


                    nashar won          score:3             rank:1

                         1)Main menu              2)exit
_
```

12)Choose any size to play at

(if the player want to play at any size)

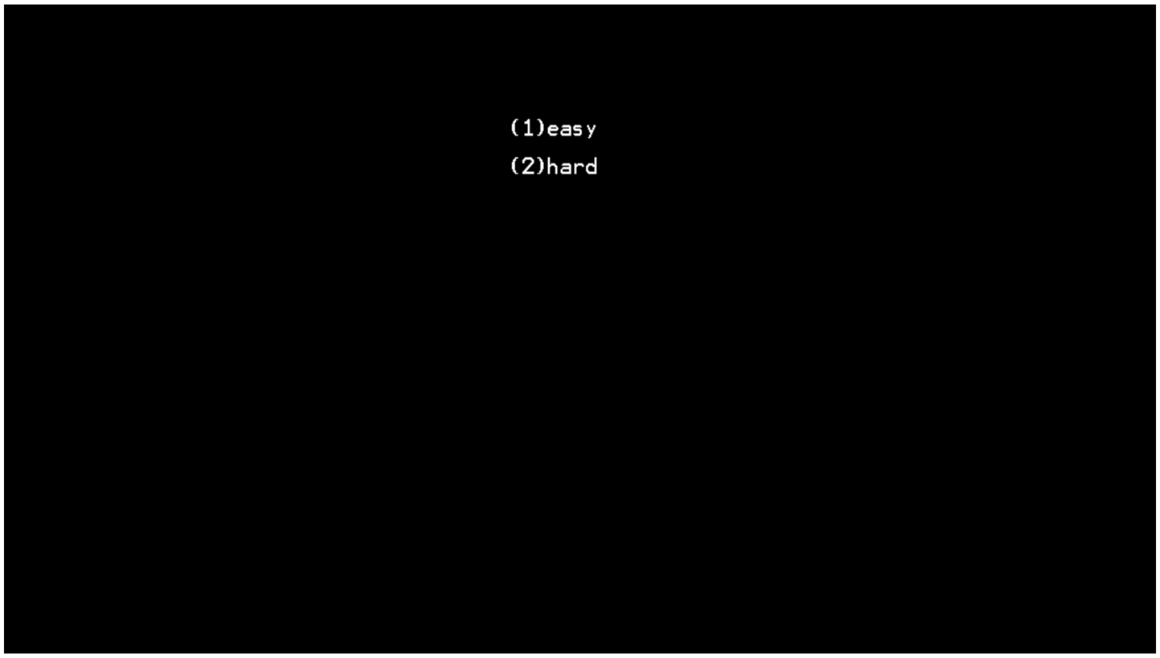


```
                    {max size 8x8}
                Enter the size of the array

     row:4                              column:7_
```



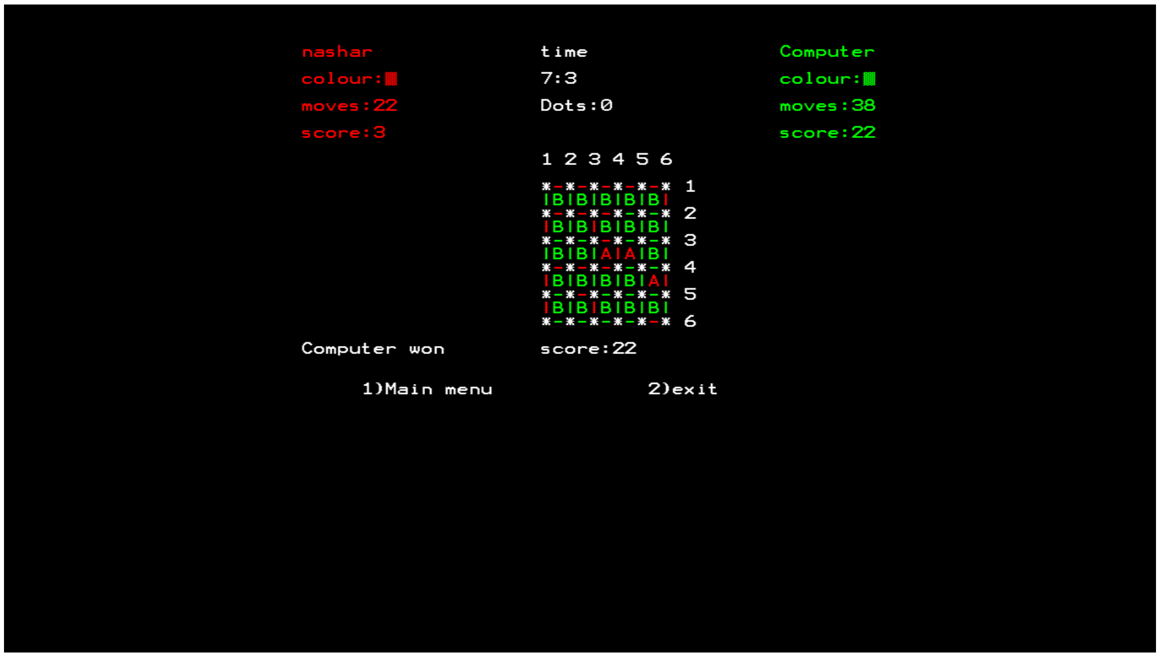
```
mohamed            time           kamal
colour:█           0:0            colour:█
moves:0            Dots:67        moves:0
score:0                           score:0
                   1 2 3 4 5 6 7 8
                   * * * * * * * * 1
                   * * * * * * * * 2
                   * * * * * * * * 3
                   * * * * * * * * 4
                   * * * * * * * * 5
mohamed's turn....
```

13)Choose the difficulty of the computer (random or greedy)



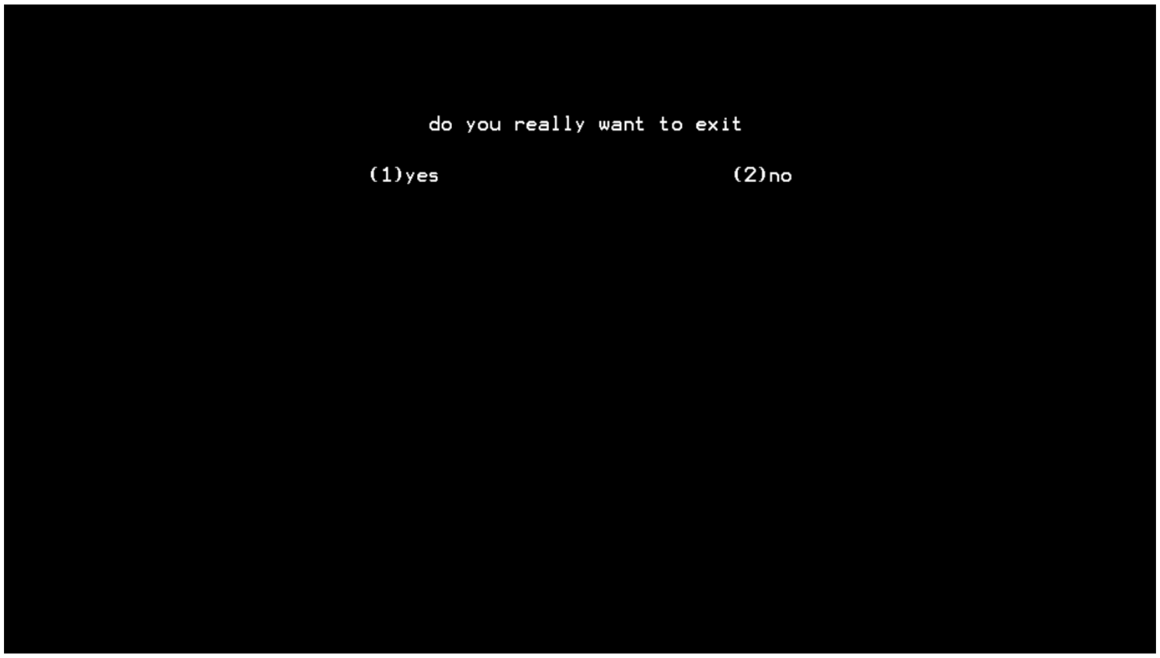
```
(1)easy
(2)hard
```

14) Lost against computer ☹ computer not put in the top 10



```
nashar              time           Computer
colour:🟥           7:3            colour:🟩
moves:22            Dots:0         moves:38
score:3                            score:22

                    1 2 3 4 5 6

                    *-*-*-*-*-* 1
                    |B|B|B|B|B|
                    *-*-*-*-*-* 2
                    |B|B|B|B|B|
                    *-*-*-*-*-* 3
                    |B|B|A|A|B|
                    *-*-*-*-*-* 4
                    |B|B|B|B|A|
                    *-*-*-*-*-* 5
                    |B|B|B|B|B|
                    *-*-*-*-*-* 6
Computer won        score:22

     1)Main menu              2)exit
```

15)Exit menu after ending a game



```
do you really want to exit

(1)yes                              (2)no
```

PSEUDOCODE

1)TOP TEN FUNCTION:

-arranges all the players from higher to lower scores and print the top ten players with the highest scores.

# open the file and load data

```
Open the file;

For(i=0;i<200;i++){

     Scan(struct array x[i].name)

     Scan(struct array x[i].score)

}

Close the file;
```

# arrange the players

```
For(j=0;j<200;j++){

    For(i=j+1;j<200;j++){

        If(struct array x[i]>struct array x[j]){

               Struct array y=struct array x[i];

               Struct array x[i]=struct array x[j];

               Struct array x[j]=struct array y;

         }

    }

}
```

# print the top 10 players in a box

```
For(i=0;i<10;i++){

Print(struct array x[i].name);

Print(struct array x[i].score);
}
```

2)Print Function:

-prints the information of each player (name, moves, score), prints the time and prints the interface of the game (dots and number of row and column).

<u># print the information;</u>

```
Print(players names );
Print(players scores);
Print(players moves);
Print(number of remaining dots);
Print(time in mins and seconds);
```

<u>#print the shape of the game</u>

```
for(k=0; k<row*2-1; k++){
    for(j=0; j<col*2-1; j++){
        if(array z[k][j]==0){
            print(dots and spaces);
          }
        if(array z[k][j]==1){
            print(move of the first player);
          }
        if(array z[k][j]==2){
            print(move of the second player);
          }
    }
}
```

3)Computer Function:

-makes the computer plays (greedy).

<u># greedy</u>

```
for(i=0; <row*2-1; i++){
        for(j=0; j<col*2-1; j++){
               if(i%2==1&&j%2==1){
                     if(the box is empty)
                            if(one side is opened && the other sides are closed){
                                   array greedy[i][j]=3;
                               }
                        }
                  }
          }
}
for(i=0; <row*2-1; i++){
        for(j=0; j<col*2-1; j++){
              if(i%2=1&&j%2==1&& array greedy[i][j]=3){
                    if(one side is opened && the other sides are closed){
                              put the last play which will close this box;
                              number of moves --;
                               number of remaining dots --;
                               turn ++;
                    for(v=0; v<row*2-1; v++){
                        for(m=0; m<col*2-1; m++){
                            if(v%2==1&&m%2==1){
                                if(all the sides of the box are closed){
                                    print character 'B' inside the box;
                                     score ++;
                                       turn--;}
                               }
                           }
                       }
                  }
             }
         }
     }



         else{        # random
                 use random function to get the value of row1 and column 1;
                  if(row1%2=0&&column1%2=1){
                      put '-' in the array;
                      number of moves --;
                      number of remaining dots --;
                      turn ++;}
                        for(v=0; v<row*2-1; v++){
                            for(m=0; m<col*2-1; m++){
                               if(all the sides of the box are closed){
                                   print character 'B' inside the box of main array;
```

```
                          score ++;
                          turn--;}
                        }
                      }
        else{
            put '|' in the array;
            number of moves --;
            number of remaining dots --;
            turn ++;}
             for(v=0; v<row*2-1; v++){
                  for(m=0; m<col*2-1; m++){
                     if(all the sides of the box are closed){
                        print character 'B' inside the box;
                        score ++;
                        turn--;}
                     }
               }
      }
```

4)UNDO FUNCTION:

```
for(u1=0; u1<row*2-1; u1++)
  {
     for(u2=0; u2<col*2-1; u2++)
     {
```

```
                If(the last play is found && not special case){
                        Change the value of this play in undo array by zero;
                        Print space in this place in the main array;
                        Number of remaining dots --;
                         If(this play belongs to player ){
                             Turn--;
                              moves of the player --;
                               for(m=0;m< row*2-1;m++){
                                     for(n=0;n< col*2-1;n++){
                                            if(the play before the last one belongs to the same player){
                                                  this play will be a special case;}
                                             }
                                         }
                               }
                   If(the last play is found && special case){
                        Change the value of this play in undo array by zero;
                         Print space in this place in the main array;
                        Number of remaining dots --;
                         Moves of this player --;}
   }
}
for(u1=0; u1<row*2-1; u1++)
   {
      for(u2=0; u2<col*2-1; u2++)
      {
           If(a side of a box is removed and this box is opened after it was closed){
            Print space instead of the character inside this box;
            Score of this player --;
      }
  }
```

REFERENCE

1)color function:

https://cboard.cprogramming.com/cplusplus-programming/59300-color-text-windows-h.html

2)time function:

https://stackoverflow.com/questions/5248915/execution-time-of-c-program

3)gotoxy function:

https://cboard.cprogramming.com/c-programming/34324-how-use-gotoxy.html

4)music function:

https://stackoverflow.com/questions/19895468/background-music-in-c

5)sound effects:

https://stackoverflow.com/questions/3845590/how-to-produce-beep-sound-using-a-escape-character

https://stackoverflow.com/questions/29493837/how-to-make-a-beep-sound-in-c-on-windows

6)isdigit(function)

https://www.tutorialspoint.com/c_standard_library/c_function_isdigit.htm

7)ispunct(fuction)

https://www.tutorialspoint.com/c_standard_library/c_function_ispunct.htm

*isaplha(function)

https://www.tutorialspoint.com/c_standard_library/c_function_isalpha.htm

NOTE:

-FLOWCHARTS ARE IN ANOTHER FOLDER DUE TO THEIR BIG SIZE

-MENU(ALOGRITHIUM)IN JPG.IMAGE

-PLAYER(ALOGRITHM) IN PDF.

TO KEEP THEIR HIGHT QUALITY.

COPYRIGHT

MADE BY:

MOHAMED ELNASHAR & TAREK MOHAMED

ID:44 ID:22