“CONNECT 4”

Programming 1 project.

PREPARED BY :

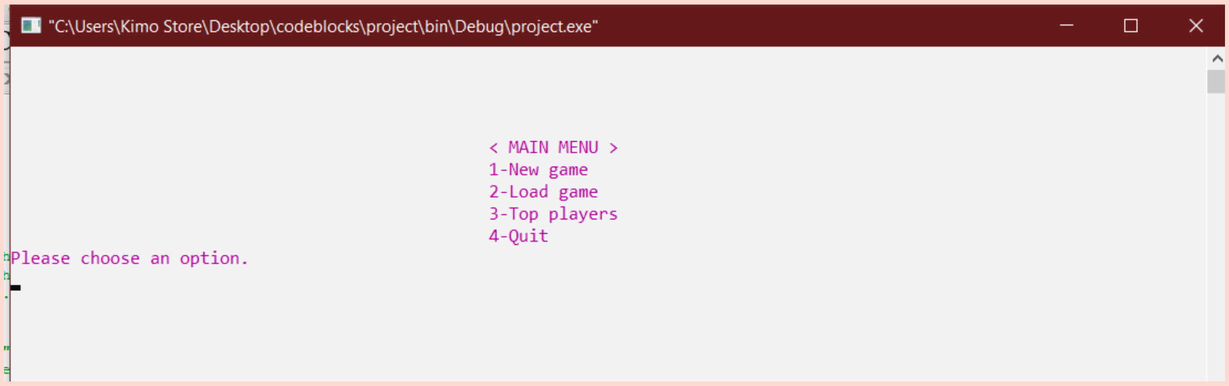
Name	ID
1. Ranime Ahmed Elsayed Shehata.	21010531
2. Youssef Tarek Hussein Yousry.	21011595

✦ REPORT CONTENTS:

1) **DESCRIPTION:**

- This is a game application which is "Connect 4". It's a 2 player game. Attached with the game an XML file from which the game parameters are read in the beginning of the game. If the file was found corrupted or missing, then the game starts with the default values which are: Height = 7 and Width = 9.
- **A main menu** is shown to choose one of four options:

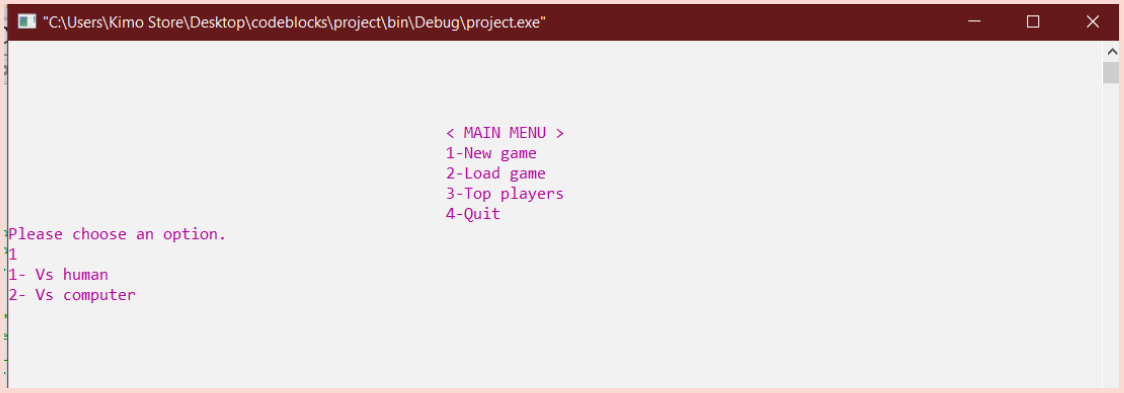
1- New game.
2- Load game.
3- Top players.
4- Quit.



```
"C:\Users\Kimo Store\Desktop\codeblocks\project\bin\Debug\project.exe"

                                        < MAIN MENU >
                                        1-New game
                                        2-Load game
                                        3-Top players
                                        4-Quit
Please choose an option.
```

- If the user chooses **'New game'** then he will be asked which mode he wants (Vs. Human / Vs. Computer).



```
"C:\Users\Kimo Store\Desktop\codeblocks\project\bin\Debug\project.exe"

                                        < MAIN MENU >
                                        1-New game
                                        2-Load game
                                        3-Top players
                                        4-Quit
Please choose an option.
1
1- Vs human
2- Vs computer
```

- Next, the game starts and player 1 is asked to enter a column number. We read a move from him and check whether it's a valid input or not. If it's a

"Vs. Human" game, the same logic is repeated for player2. While in case of "Vs. computer" game, the computer plays randomly according to an executed logic.

- A user interface is updated in each turn displaying the score and number of moves of each player, time passed so far and whose turn it is.
- An in-game menu is also displayed, giving each player the option to undo, redo, save his game or return to the menu.
- When the game ends, the winner is asked for his name and a high scores list is updated and displayed sorted.
- Eventually, the player is asked if he wants to return to menu or exit.

*****

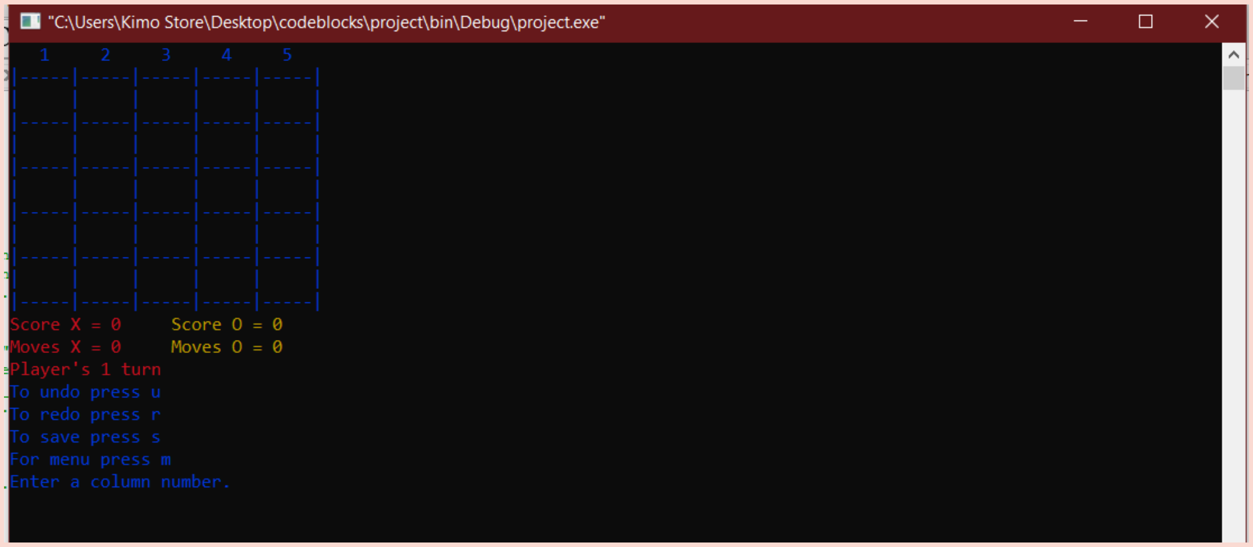
2) FEATURES:

- Attached with the application a configuration file in XML format to load the game parameters. If the fill was missing or corrupted, the user is asked to enter the file path 3 times. If it was still invalid, then default values are loaded (Height = 7 and Width = 9).
- A **main menu** is displayed at the beginning:

  1- New game.
  2- Load game.
  3- Top players.
  4- Quit.
- **Top player:** if the user enters '3', a list showing the first X top players sorted descendingly with their highscores (X is a defined in the configuration XML file with default value = 10).
- **Load game:** if the user enters '2', he will have to choose one of 3 games to load. When a game is loaded, it's continued from the player's turn where they left and the undo/redo data will be found saved as well.
- **New game:** if the user enters '1', he will have to choose the game mode:

  1- Vs. Human.
  2- Vs. Computer.
- Suppose the user chooses a "Vs. human" mode. The game starts and the board will be displayed asking player 1 to enter a column.

- If a player enters a character or an out of scope number, a "Enter a valid column number!!" message is displayed asking for his turn once again till he enters a valid number.
- If a player enters a column number which is full, a ""Please choose an empty column!!" message is displayed asking for his turn once again till he chooses an empty column.
- Turns are taken between the two players and as they play, the score of each is displayed as well as their moves so far and whose turn to play.
- The time passed is also displayed on screen since, the user chooses the game mode till the game ends. Time can be displayed in seconds, minutes and hours.
- An **<u>in-game menu</u>** is displayed on screen:
  - To undo press u.
  - To redo press r.
  - To save press s.
  - For menu press m.



```
"C:\Users\Kimo Store\Desktop\codeblocks\project\bin\Debug\project.exe"
   1     2     3     4     5
|-----|-----|-----|-----|-----|
|     |     |     |     |     |
|-----|-----|-----|-----|-----|
|     |     |     |     |     |
|-----|-----|-----|-----|-----|
|     |     |     |     |     |
|-----|-----|-----|-----|-----|
|     |     |     |     |     |
|-----|-----|-----|-----|-----|
|     |     |     |     |     |
|-----|-----|-----|-----|-----|
Score X = 0      Score O = 0
Moves X = 0      Moves O = 0
Player's 1 turn
To undo press u
To redo press r
To save press s
For menu press m
Enter a column number.
```

- During the game, if a player wants to undo the last move, he can press 'u' or 'U' and if he wants to redo the undid move, he can press 'r' or 'R'.
- If a player plays after he presses 'u', then he can't redo the deleted move.
- If the player wants to save his progress in the game, he can press 's' or 'S'. He will be asked to choose a file of 3 files to save his game into. He can load his saved game and continue or view it from the main menu displayed when running the program in the beginning.

- The player can exit from the game at any time by pressing the ‘esc’ button.
- When the game ends, the winner is declared and asked to enter his name. If he won before, his score will be compared to his old one in the highscores list. If he scores a new highscore, it will be updated and the list will be displayed sorted descendingly. If it’s his first win, his name will be appended to the list with his score and the list will also be displayed sorted descendingly.
- The highscores list displays the rank, player’s name and score sorted from the highest.
- The player is asked when the game ends if he wants to return to menu or exit.

*****

3) ASSUMPTIONS:

1.We gave an assumption that the rows and the columns shouldn’t be less than 4 and the highscores not less than 3, it will give an error when reading the XML and will ask for a path.

2. Player’s name shouldn’t exceed 50.

3. The highscore file can’t hold more than 100 person.

*****

4) DATA STRUCTURE:

- Structures:

- play: typedef struct, its members: name, score, moves.
- scores: typedef struct, its members: name, score.

- Arrays:

- a : a 2D array for the game’s grid of size equals to the parameters of the board.
- undoarr: a 2D array for implementing the undo function.
- redoarr: a 2D array for implementing the redo function.
- x_column: for reading the input from the player when choosing a column.

- c1, c2, h1, h2, w1, w2, hi1, hi2: for reading the XML file and checking its validity.

*****

5) IMPLEMENTED FUNCTIONS:

- **player1:**

The function is called in “start_game” when it’s player’s 1 turn (turn=1). The player should enter the a valid column number. So, the function : first, checks the value entered; if it was a a one character either a ‘u’ or a ‘U’ so it calls the “undo” function and either calls “player2” to play again if it’s a “Vs. human” game or prints the board for player1 to play again if it’s “Vs. computer” game. Second, if the entered value was a one character ‘r’ or ‘R’ it calls the redo function and player 2 is then asked to play his turn. Third, if ‘s’ or ‘S’ was entered, the game is saved by calling the “save_game” function. Finally, if the entered value was none of those mentioned before, then the scenario will be as follow:

1. If it’s not a number(any character) or a number out of scope then the player will be asked to enter a valid number.
2. If he entered a full column number then he’s asked to choose an empty column.
3. Best case scenario; if he enters a column number which is not full, then an ‘X’ is assigned in a column in the game board corresponding to the value entered and player2 is then asked for his turn!!

- **player2:**

The function is called in “start_game” when it’s player’s 2 turn (turn=2 and computerturn =1). The player should enter the a valid column number. So, the function : first, checks the value entered; if it was a a one character either a ‘u’ or a ‘U’ so it calls the “undo” function and either calls “player1” to play again if it’s a “Vs. human” game or prints the board for player2 to play again if it’s “Vs. computer” game. Second, if the entered value was a one character ‘r’ or ‘R’ it calls the redo function and player 1 is then asked to play his turn. Third, if ‘s’ or ‘S’ was entered, the game is saved by calling the “save_game” function. Finally, if the entered value was none of those mentioned before, then the scenario will be as follow:

1- If it’s not a number(any character) or a number out of scope then the player will be asked to enter a valid number.
2- If he entered a full column number then he’s asked to choose an empty column.
3- Best case scenario; if he enters a column number which is not full, then an ‘O’ is assigned in a column in the game board corresponding to the value entered and player1 is then asked for his turn!!

- **computer:**

The function is called in “start_game” when it’s computer’s turn (computertrun=0). It randomly assigns an ‘O’ in the gameboard when we’re in a “Vs. computer” game. The mechanism of choosing a column is that by passing through the array elements one by one in order and once an empty cell is found, then an ‘O’ is assigned there.

- **printboard:**

It’s initially called in the main when starting a new game and prints the game board according to the values of the (row) and (column) parameters.

1. It prints an ‘X’ or ‘O’ in the board cells according to the number of column scanned from the player.
2. By calling the “calc_moves” function and the “hor_score” function and calculating the no of moves and the score for both players and printing them up to last update everytime.
3. It prints who’s turn to play according to the variable (turn). If turn=1 then it prints “Player’s 1 turn”. If turn=2 then it prints “Player’s 2 turn”.
4. An in-game menu is printed everytime:
   “To undo press u”
   “To redo press r”
   “To save press s”
   “For menu press m”

5. By the help of the variable “sent”, it asks the player for a column number according to the scenario. When calling the function “printboard” in the “player1” or “player2” functions, the parameter “sent” is given one of 3 values :
   - sent = 0: "Enter a valid column number!!" is printed.
   - sent = 1: "Enter a column number.” is printed.
   - sent = 2: "Please choose an empty column!!” is printed.

- ## calc_moves:

It calculates the number of moves for both players by passing through all board cells and incrementing the value of “play1.moves” by 1 everytime an ‘X’ appears and doing the same for “play2.moves” whenever an ‘O’ appears. (We initialized the variables “play1.moves” and play2.moves” to zero in the beginning).

- ## count_filled:

It counts the number of empty cells in the board.

- ## total_score:

It counts the score for each player by passing through all board cells and increments the value of “play1.score” and “play2.score” whenever any of them connects a four any of four ways (horizontally, vertically, diagonally right, diagonally left).

- ## undo:

It removes the last move(s) played. We check first whether there’s anything to be undone or not, by calculating the sum of the number of moves of both players (by the help of “calc_moves” function). If that sum equals zero, then a message is printed says “The board is already empty!!!”. Otherwise, we defined the “undoarr” in the main with the same number of rows and columns of the game board and initialized its elements to zero. Whenever a player plays, a number is written in the undoarr in a cell corresponding to where the player played. If a player pressed ‘u’ or ‘U’, the number will be deleted from the cell which was filled last and the same number is written in the “redoarr” in a place corresponding to that from which we deleted the number.

- **redo:**

It returns the character(s) removed by the undo function. Since we assigned a specific number to a specific cell in the “redoarr” when we undid a move, then when calling the redo function, it searches for the largest number and returns a character to the board corresponding to where the largest number was found. It returns ‘X’ or ‘O’ by the help of “ calc_moves” function. If movesx = moveo then it’s player’s 1 turn and an ‘X’ is returned. If movex>moveo, then it’s player’s 2 turn and an ‘O’ is returned.

If a player plays after a redo then the “redoarr” will be all zeros and if he tried to redo then a “There is nothing to be redone!!!” message appears.

- **start_game:**

It prints the board and switchs turns between the players according to the value of the “turn” variable.

If turn=1, the “player1” function is called. If turn=2 & computerturn=1, then “player2” function is called. If computerturn=0, then “computer” function is called.

In that function, time passed in the game is evaluated and printed in seconds, minutes and hours since the start of the game itself till it ends.

- **save_game:**

If a player entered ‘s’ or ‘S’, the “save_game” function is called and asks the user to choose one of 3 files to save his game into “File 1, File 2, File 3”.

For example, if he enters “1”, then a binary file with name “game1” will be created and the gameboard, undoarr and redoarr will be saved.

While choosing the file, if he enters a character or a digit other than 1 or 2 or 3, a “Please choose a valid file to save the game: 1-File 1   2-File 2   3-File 3 ” appears.

- **load_game:**

If a player chose to load a game from the mainmenu, then he will have to choose one of 3 files to access to the saved game into “File 1, File 2, File 3”.

For example, if he enters “1”, then a binary file with name “game1” will be accessed and the game which is saved into will appear so he can see it if it’s a completed game or if it’s not finished, he can continue it as well as the undo data will be saved.

While choosing the file, if he enters a character or a digit other than 1 or 2 or 3, a “Please choose a valid file to load the game: 1-File 1   2-File 2   3-File 3 ” appears.

- **xml:**

The function is called when running the application. It open the XML file containing the game configurations, reads what’s inside the file, writes it’s content in an array ignoring all the white spaces and closes the file. Some arrays are declared in the function (c1, c2, h1, h2, w1, w2, hi1, hi2). The function calls the “xml_ind” function.

- **xml_ind:**

The function takes the array “a” as a parameter which contains all contents of the XML file ignoring the spaces and the array “check” which contains a substring to check the occurrence of the tag. This function checks the occurrence of each tag in the array “a” (technically it checks it’s occurrence in the file) and returns the index of occurrence + string length of the tag to determine the last index of the tag. It also the checks the index of the tag close. This mechanism is repeated 3 times for reading 3 configuration values. Then, the “checkop” function is called.

- **checkop:**

The function checks what’s between the tag and its end and reads it. If it’s a valid number, then it’s returned to the xml function and the value is scanned as the game parameter. While if a character or any invalid corruption is detected in the file, the user is asked to enter the file path, if for 3 times he enters a wrong path, then the default values are loaded which are (Height=7, Width=9, Highscores=10).

- **read_scores:**

The function reads the leaderboard from a file.

- **write_scores:**

The function opens a file and writes the new highscore.

- **print_scores:**

The function prints the leaderboard from the file.

- **arrange:**

The function sorts the highscores in the leaderboard descendingly everytime a new highscore is appended.

- **menuu:**

The function is initially called in the main and a main menu is displayed.

If you enter 1, a new game starts and the “start_game” function is called.

If you enter 2, you will have to choose a saved game to load.

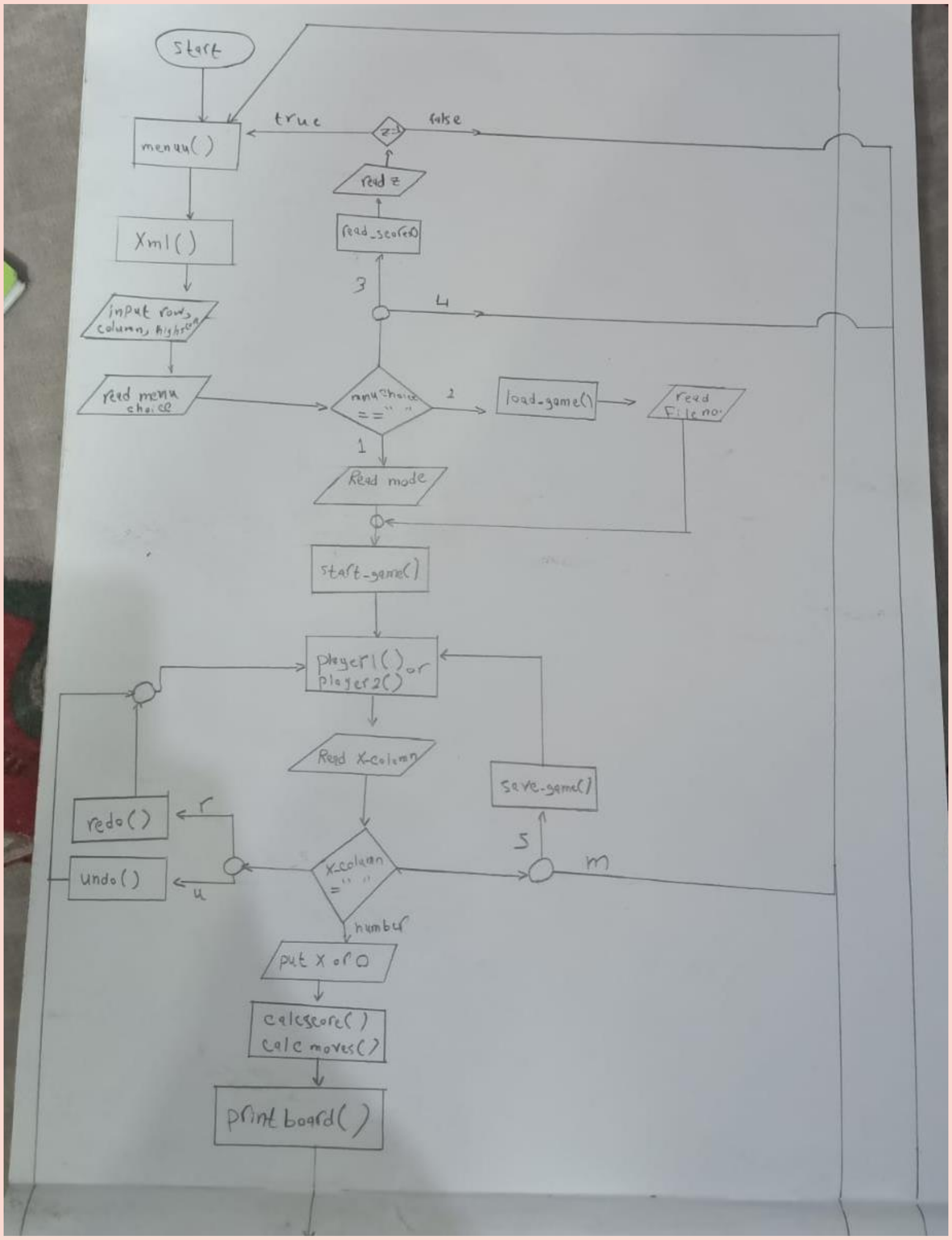
If you enter 3, the leaderboard is displayed with X top players; where X is a parameter given in the XML configuration file and it’s default value in case of file corruption is 10.

If you enter 4, you will exit from the game.

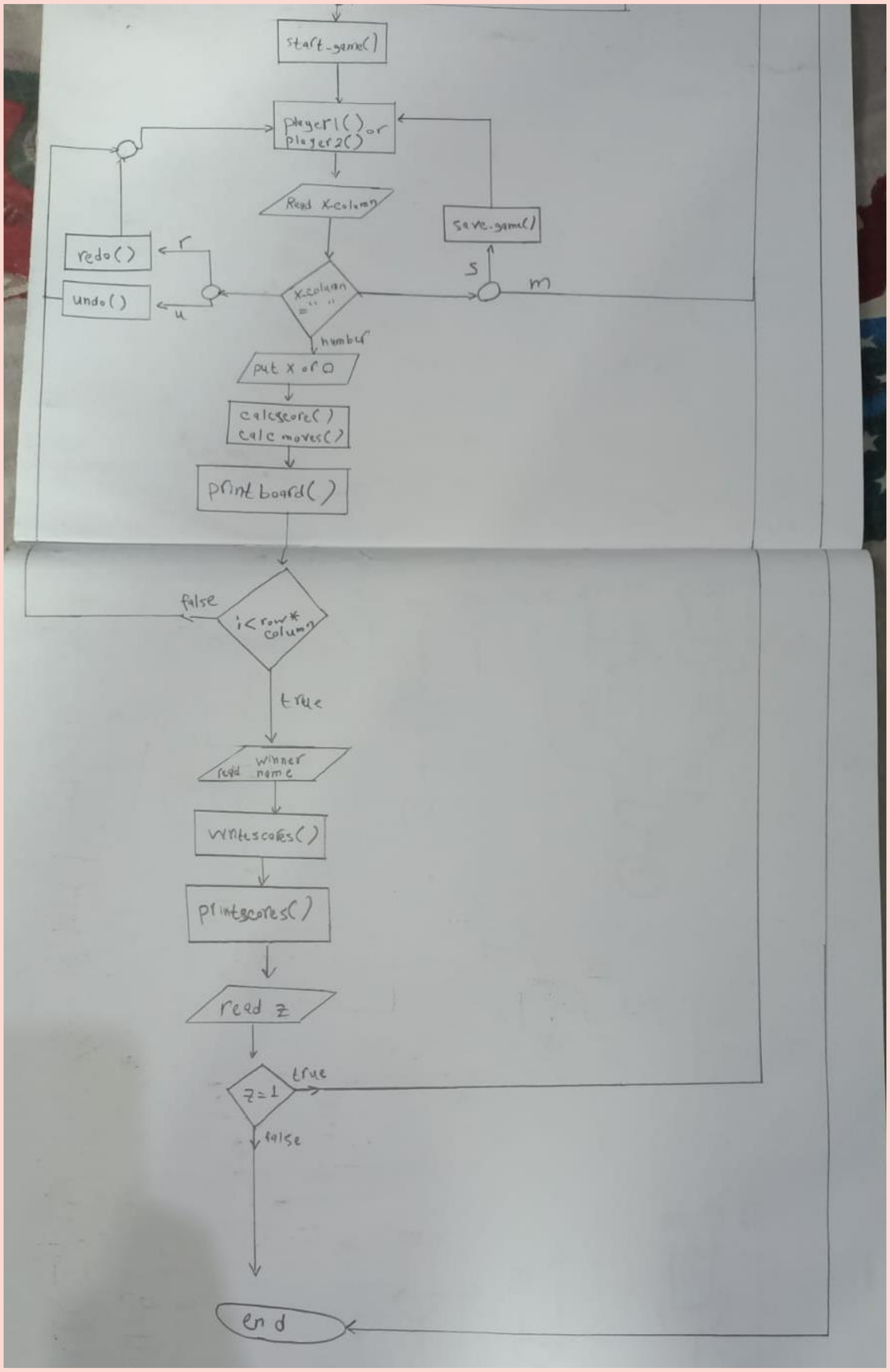
During the game, you can enter ‘m’ to return to the main menu and when the game ends the user is asked if he wants to return to main menu or exit.

*****

6) FLOW CHART:



start
menuu( )
true
z=1
false
read z
read_score()
Xml( )
3
4
input rows, column, highscore
read menu choice
mnu choice == " "
2
load_game()
read File no.
1
Read mode
start_game()
player1( ) or player2( )
Read X-column
save-game()
redo( )
r
undo( )
u
X-column = " "
5
m
number
put X or O
calcscore( )
calc moves( )
print board( )


start_game()
player1() or player2()
Read X-column
save_game()
redo()
r
undo()
u
X-column = " "
s
m
number
put X or O
calcscore()
calc moves()
print board()
false
i < row * column
true
read winner name
writescores()
printscores()
read z
z=1
true
false
end

*****

7) <u>USER MANUAL :</u>

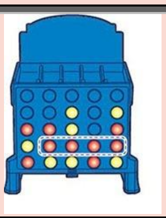
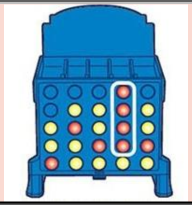
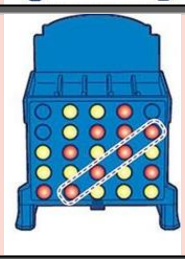
1. Run the application. An XML file attached to the game will be read by the program. If the file was found missing or corrupted or in a different directory, please enter the valid file path. If you failed to do so for 3 times, the default values will be loaded.
2. Main menu is now displayed!
   - ⇒ If you want to start a new game, enter 1.
   - ⇒ If you want to load a saved game, enter 2.
   - ⇒ If you want to have a look on the top players, enter 3.
   - ⇒ If you want to quit, enter 4.
3. If you enter 1 to start a new game, you have to chase game mode!
   - ⇒ For a “vs. human” mode, enter 1.
   - ⇒ For a “vs. computer” mode, enter 2.
4. The game starts now!! The game board is now printed on your screen and you and the other player (or computer) should take turns.
5. You should enter a valid column number (a column number in scope and not a character). If you enter an invalid character, a message is printed on your screen!
6. Choose an empty column to play!
7. While playing, your score, number of moves is displayed as well as the other player’s data. Time spent in the game is also shown in seconds, minutes and hours.
8. An in-game menu is displayed below!
   - ⇒ If you want to undo the last move played, press ‘u’.
   - ⇒ If you want to redo, press ‘r’.
   - ⇒ If you want to save your game, press ‘s’.
9. When you undo, the last move played by the second player is removed.
10. When you redo, the last undid move is returned in the board.
11. While playing in a “vs. computer” mode, the computer can’t undo or redo a move and if you undo, then the computer’s last move and your last move are

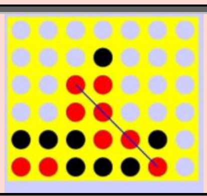
removed and if you redo, then those two moves are returned in the game board.

12. If you undo a move and played one, you can't redo the undid move.
13. Computer's moves are randomly played by executing some logic.
14. During the game, if you want to save the game, press 's'. Then, choose a file to save your progress into.
15. Game ends when board is full!!
16. When board becomes full, a winner is declared according to the highestscore.
17. The winner is asked to enter his name.
18. A leaderboard is displayed with the rank, name and score of winners so far sorted descendingly.
19. If your name is already in the leaderboard then a comparison between your recorded score and your score now is made. If you score a higher score, your new score will be updated and your name will be sorted with a new rank. If it's your first win, your name will be appended in the list.
20. After displaying the leaderboard, you can choose to return to the main menu or to exit the game.

21. <u>HOW TO SCORE A POINT ??</u>

If you connect four 'X's or 'O's horizontally, vertically, diagonally right or diagonally left.

✦ HORIZONTAL WIN.	
✦ VERTICAL WIN.	
✦ DIAGONALLY RIGHT WIN.	

✦ DIAGONALLY LEFT WIN.	

*****

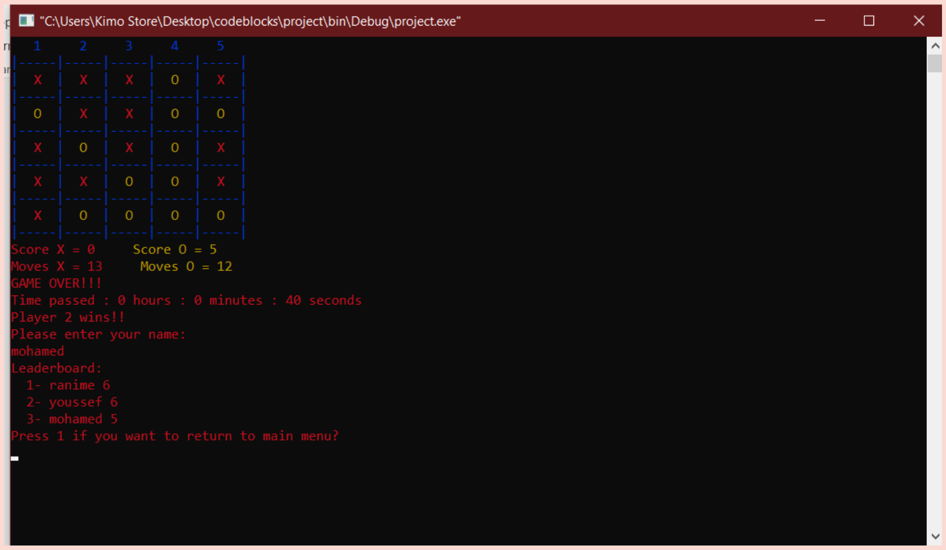
8) EXTRA APPLIED FEATURES:

1. Applying colors to the printboard, giving each player a color and changing background's color.
2. Saving the undo/redo data when saving & loading the game.

*****

9) SAMPLE RUNS:

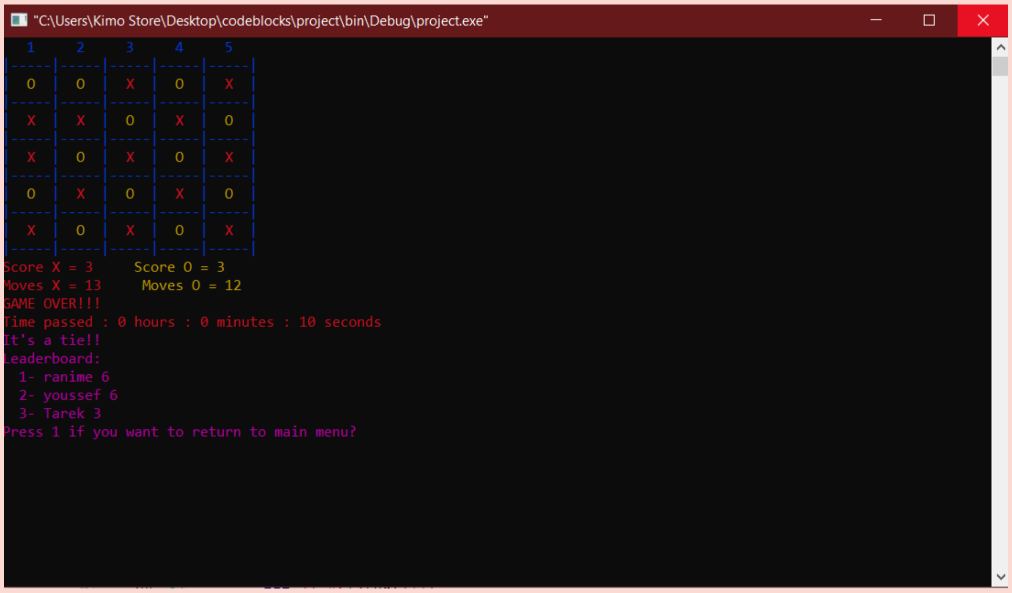
⇒ Player 2 wins:



"C:\Users\Kimo Store\Desktop\codeblocks\project\bin\Debug\project.exe"

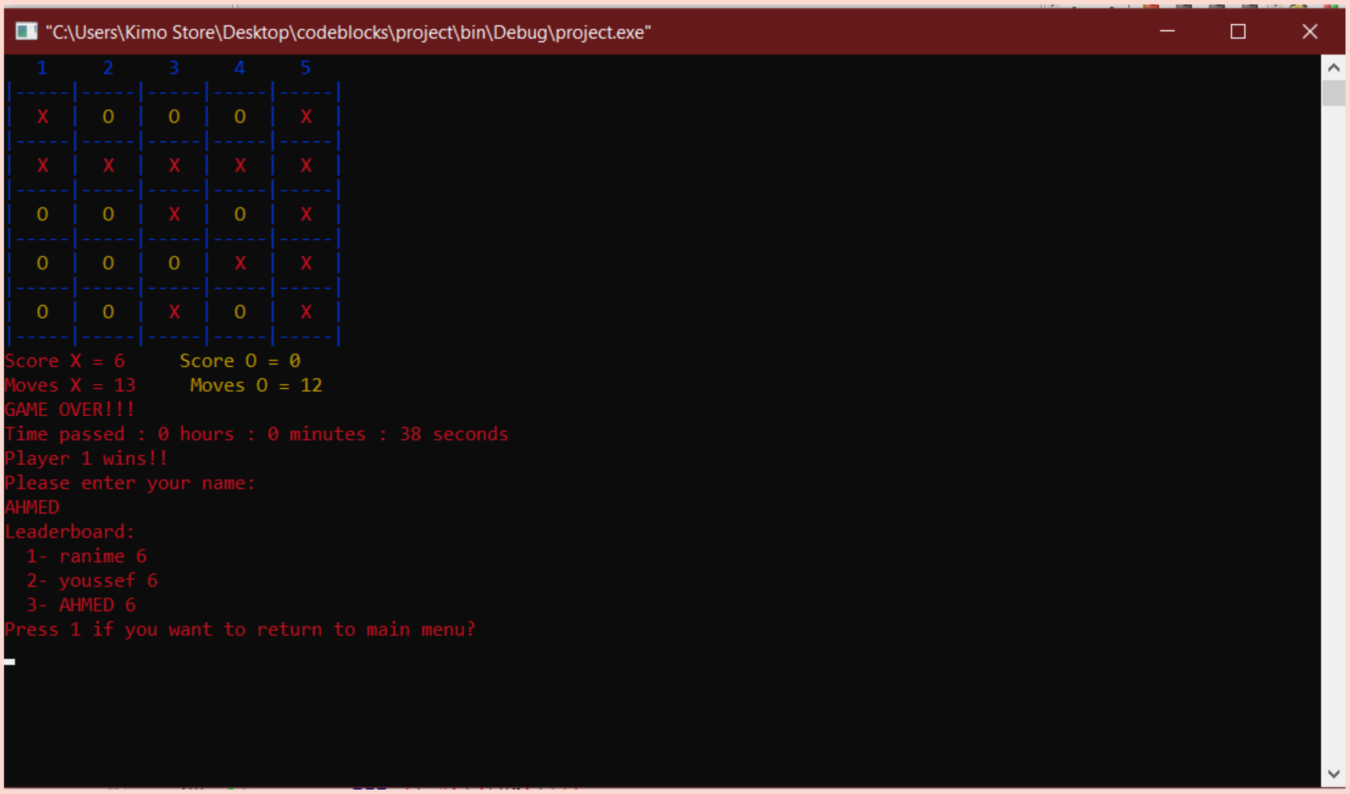
```
   1     2     3     4     5
|-----|-----|-----|-----|-----|
|  X  |  X  |  X  |  O  |  X  |
|-----|-----|-----|-----|-----|
|  O  |  X  |  X  |  O  |  O  |
|-----|-----|-----|-----|-----|
|  X  |  O  |  X  |  O  |  X  |
|-----|-----|-----|-----|-----|
|  X  |  X  |  O  |  O  |  X  |
|-----|-----|-----|-----|-----|
|  X  |  O  |  O  |  O  |  O  |
|-----|-----|-----|-----|-----|
Score X = 0      Score O = 5
Moves X = 13      Moves O = 12
GAME OVER!!!
Time passed : 0 hours : 0 minutes : 40 seconds
Player 2 wins!!
Please enter your name:
mohamed
Leaderboard:
  1- ranime 6
  2- youssef 6
  3- mohamed 5
Press 1 if you want to return to main menu?
```

⇒ It's a tie:



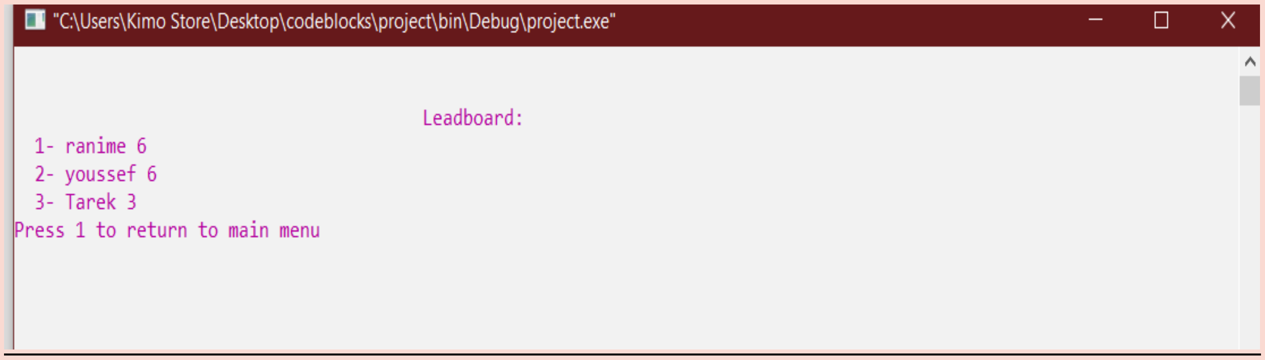
```
"C:\Users\Kimo Store\Desktop\codeblocks\project\bin\Debug\project.exe"
  1    2    3    4    5
|----|----|----|----|----|
| O  | O  | X  | O  | X  |
|----|----|----|----|----|
| X  | X  | O  | X  | O  |
|----|----|----|----|----|
| X  | O  | X  | O  | X  |
|----|----|----|----|----|
| O  | X  | O  | X  | O  |
|----|----|----|----|----|
| X  | O  | X  | O  | X  |
|----|----|----|----|----|
Score X = 3      Score O = 3
Moves X = 13      Moves O = 12
GAME OVER!!!
Time passed : 0 hours : 0 minutes : 10 seconds
It's a tie!!
Leaderboard:
  1- ranime 6
  2- youssef 6
  3- Tarek 3
Press 1 if you want to return to main menu?
```

⇒ Player 1 wins:



```
"C:\Users\Kimo Store\Desktop\codeblocks\project\bin\Debug\project.exe"
  1    2    3    4    5
|----|----|----|----|----|
| X  | O  | O  | O  | X  |
|----|----|----|----|----|
| X  | X  | X  | X  | X  |
|----|----|----|----|----|
| O  | O  | X  | O  | X  |
|----|----|----|----|----|
| O  | O  | O  | X  | X  |
|----|----|----|----|----|
| O  | O  | X  | O  | X  |
|----|----|----|----|----|
Score X = 6      Score O = 0
Moves X = 13      Moves O = 12
GAME OVER!!!
Time passed : 0 hours : 0 minutes : 38 seconds
Player 1 wins!!
Please enter your name:
AHMED
Leaderboard:
  1- ranime 6
  2- youssef 6
  3- AHMED 6
Press 1 if you want to return to main menu?
```

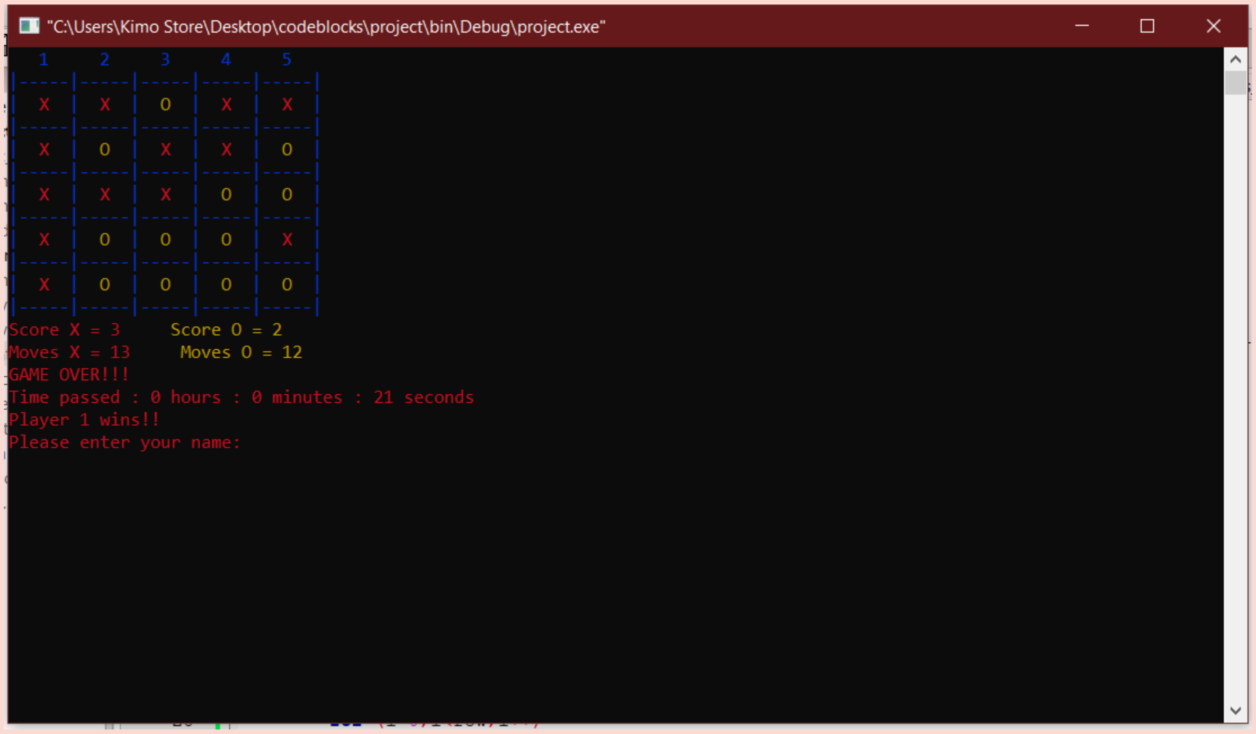
⇒ The leaderboard:



```
"C:\Users\Kimo Store\Desktop\codeblocks\project\bin\Debug\project.exe"

                                        Leadboard:
  1- ranime 6
  2- youssef 6
  3- Tarek 3
Press 1 to return to main menu
```

⇒ A “vs. computer” game:



```
"C:\Users\Kimo Store\Desktop\codeblocks\project\bin\Debug\project.exe"
   1     2     3     4     5
|-----|-----|-----|-----|-----|
|  X  |  X  |  O  |  X  |  X  |
|-----|-----|-----|-----|-----|
|  X  |  O  |  X  |  X  |  O  |
|-----|-----|-----|-----|-----|
|  X  |  X  |  X  |  O  |  O  |
|-----|-----|-----|-----|-----|
|  X  |  O  |  O  |  O  |  X  |
|-----|-----|-----|-----|-----|
|  X  |  O  |  O  |  O  |  O  |
|-----|-----|-----|-----|-----|
Score X = 3      Score O = 2
Moves X = 13      Moves O = 12
GAME OVER!!!
Time passed : 0 hours : 0 minutes : 21 seconds
Player 1 wins!!
Please enter your name:
```

*****

GITHUB’S LINK:

https://github.com/ranimeshehata2001/Connect-four

*****