



Faculty of Computers and Artificial Intelligence

Cairo University



CS213: Object Oriented Programming
Under Supervision of Dr. Mohammed El-Ramly

Assignment: 2

Section Group: 7

Task: 2 & 3 & 5

Board Games – Assignment 2 Report

Name	Malak Amr Ahmed	Abdelrhman Yasser Ali	Yassin Ahmed Ali
ID's	20230416	20230220	20230465
Emails	malakamrahmdd@gmail.com	abdelrahmany05@gmail.com	yassinsawy@outlook.com
Phones	01156903124	01288217119	01033822107

Classes Description

4x4_XO.h

Class: FourByFour_XO_Board

Description: Represents a 4x4 tic-tac-toe board. Manages board updates, move validation, and win/draw conditions.

Attributes:

- int rows
- int columns
- T** board
- int n_moves

Methods:

- FourByFour_XO_Board()
- ~FourByFour_XO_Board()
- bool validate_token(int& x, int& y, T symbol)
- bool validate_move(int x, int y, T symbol)
- bool update_board(int x, int y, T symbol) override
- void delete_old(int x, int y)
- void display_board() override
- bool is_win() override
- bool is_draw() override
- bool game_is_over() override

Class: FourByFour_XO_Player

Description: Represents a player who makes moves in the 4x4 tic-tac-toe game.

Attributes:

- string name
- T symbol
- Board<T>* boardPtr

Methods:

- void getmove(int& x, int& y) override

Class: FourByFour_XO_Random_Player

Description: Represents a random AI player for the 4x4 tic-tac-toe game.

Attributes:

- string name
- T symbol
- Board<T>* boardPtr

Methods:

- void getmove(int& x, int& y) override

5x5_XO.h

Class: FiveByFiveBoard

Description: Represents a 5x5 tic-tac-toe board. Counts sequences of three consecutive symbols and checks for win/draw conditions.

Attributes:

- int rows
- int columns
- T** board
- int n_moves
- int win

Methods:

- FiveByFiveBoard()
- ~FiveByFiveBoard()
- bool update_board(int x, int y, T symbol) override
- void display_board() override
- int countSequences(T player)
- int getMoves()
- void assign_move(int &x, int &y)
- bool is_win() override
- bool is_draw() override
- bool game_is_over() override

Class: FiveByFivePlayer

Description: Represents a player in the 5x5 tic-tac-toe game.

Attributes:

- string name
- T symbol
- Board<T>* boardPtr

Methods:

- void getmove(int& x, int& y) override

Class: FiveByFiveRandomPlayer

Description: Represents a random AI player in the 5x5 tic-tac-toe game.

Attributes:

- string name
- T symbol

Methods:

- void getmove(int& x, int& y) override

FourInRow.h

Class: FourInRow

Description: Represents a Connect Four game board with 6 rows and 7 columns.

Attributes:

- int rows
- int columns
- T** board
- int n_moves

Methods:

- FourInRow()
- ~FourInRow()
- bool update_board(int x, int y, T symbol) override
- void display_board() override
- bool is_win() override
- bool is_draw() override
- bool game_is_over() override
- void computer_move(T symbol)

Class: FourInRow_Player

Description: Represents a player in the Connect Four game.

Attributes:

- string name
- T symbol

Methods:

- void getmove(int& x, int& y) override

Class: FourInRow_Random_Player

Description: Represents a random AI player for the Connect Four game.

Attributes:

- string name
- T symbol

Methods:

- void getmove(int& x, int& y) override

Misere_XO.h

Class: Misere_XO

Description: Represents a 3x3 tic-tac-toe game where the objective is to force the opponent to complete a line.

Attributes:

- int rows
- int columns
- T** board
- int n_moves
- int win

Methods:

- Misere_XO()
- ~Misere_XO()
- bool update_board(int x, int y, T symbol) override
- void display_board() override
- bool is_win() override
- bool is_draw() override
- bool game_is_over() override
- bool getWin()

- int getMoves()
- void assign_move(int& x, int& y)

Class: Misere_XO_Player

Description: Represents a player in the Misere tic-tac-toe game.

Attributes:

- string name
- T symbol
- Board<T>* boardPtr

Methods:

- void getmove(int& x, int& y) override

Class: Misere_XO_Random_Player

Description: Represents a random AI player for the Misere tic-tac-toe game.

Attributes:

- string name
- T symbol

Methods:

- void getmove(int& x, int& y) override

Numerical_Tic.h

Class: NumericalTicTacToe

Description: Represents a numerical tic-tac-toe game where players use numbers to sum up to 15.

Attributes:

- int rows
- int columns
- T** board
- int n_moves

Methods:

- NumericalTicTacToe()
- ~NumericalTicTacToe()
- bool update_board(int x, int y, T symbol) override
- void display_board() override
- bool is_win() override
- bool is_draw() override
- bool game_is_over() override

- bool validate_move(int x, int y, T symbol)

Class: NumericalTicToe_Player

Description: Represents a player who makes moves in the numerical tic-tac-toe game.

Attributes:

- string name
- T symbol
- set<int> used

Methods:

- void get_value(T &symbol)
- void getmove(int& x, int& y) override

Class: NumericalTicToe_Random_Player

Description: Represents a random AI player in the numerical tic-tac-toe game.

Attributes:

- string name
- T symbol
- set<int> used
- Board<T>* boardPtr

Methods:

- void getmove(int& x, int& y) override

pyramid_X_O.h

Class: Pyramid_X_O_Board

Description: Represents a pyramid-shaped tic-tac-toe board with 3 rows and a varying number of columns.

Attributes:

- int rows
- int columns
- T** board
- int n_moves

Methods:

- Pyramid_X_O_Board()
- ~Pyramid_X_O_Board()
- bool update_board(int x, int y, T symbol) override
- void display_board() override

- bool is_win() override
- bool is_draw() override
- bool game_is_over() override

Class: Pyramid_X_O_Player

Description: Represents a player in the pyramid tic-tac-toe game.

Attributes:

- string name
- T symbol

Methods:

- void getmove(int& x, int& y) override

Class: Pyramid_X_O_Random_Player

Description: Represents a random AI player in the pyramid tic-tac-toe game.

Attributes:

- string name
- T symbol

Methods:

- void getmove(int& x, int& y) override

Ultimate_XO.h

Class: Ultimate_XO_Board

Description: Represents a 9x9 ultimate tic-tac-toe board consisting of smaller 3x3 boards.

Attributes:

- int rows
- int columns
- char** board
- int n_moves
- char small_board[3][3]

Methods:

- Ultimate_XO_Board()
- ~Ultimate_XO_Board()
- bool validate_move(int x, int y, char symbol)
- bool update_board(int x, int y, char symbol) override
- void display_board() override
- char check_win(int r, int c)

- bool is_win() override
- bool is_draw() override
- bool game_is_over() override

Class: Ultimate_XO_Player

Description: Represents a player in the ultimate tic-tac-toe game.

Attributes:

- string name
- T symbol

Methods:

- void getmove(int& x, int& y) override

Class: Ultimate_XO_Random_Player

Description: Represents a random AI player in the ultimate tic-tac-toe game.

Attributes:

- string name
- T symbol
- Board<T>* boardPtr

Methods:

- void getmove(int& x, int& y) override

Word_TicTac.h

Class: Word_TicTac

Description: Represents a word-based 3x3 tic-tac-toe game where players form words from a dictionary.

Attributes:

- int rows
- int columns
- T** board
- int n_moves
- set<string> dic

Methods:

- Word_TicTac()
- ~Word_TicTac()
- bool update_board(int x, int y, T symbol) override
- void display_board() override

- bool is_win() override
- bool is_draw() override
- bool game_is_over() override

Class: Word_TicTac_Player

Description: Represents a player in the word-based tic-tac-toe game.

Attributes:

- string name
- T symbol

Methods:

- void get_value(T &symbol)
- void getmove(int& x, int& y) override

Class: Word_TicTac_Random_Player

Description: Represents a random AI player in the word-based tic-tac-toe game.

Attributes:

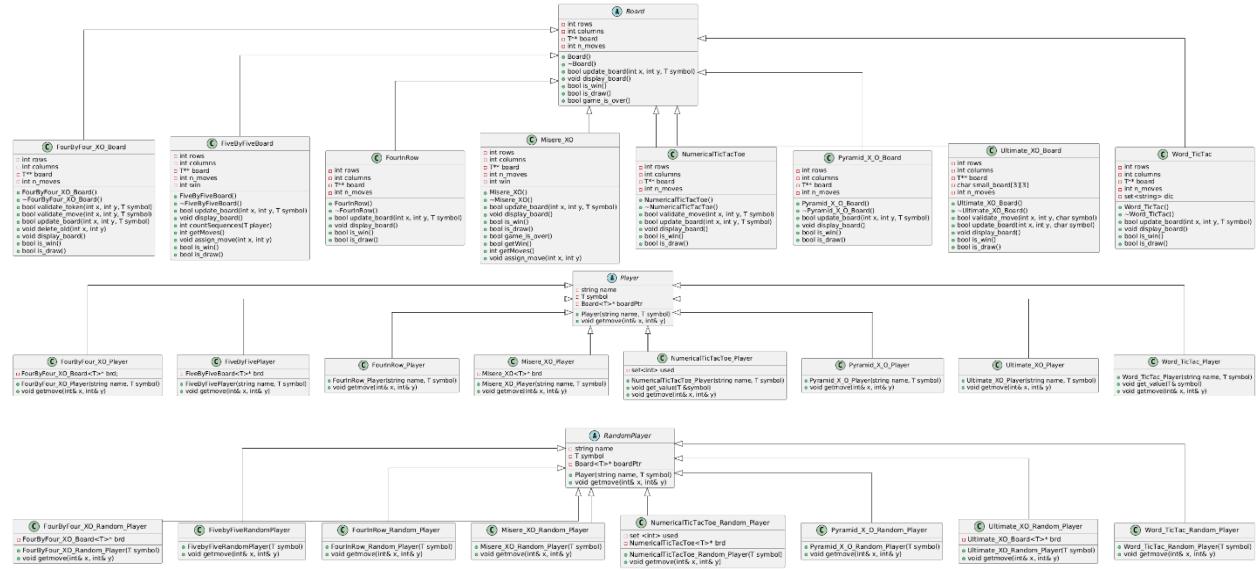
- string name
- T symbol

Methods:

- void getmove(int& x, int& y) override

Classes UML Diagram

This is a generic class diagram for the games



The detailed diagram for each game is provided here:

<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center; padding: 5px;">A Board</th></tr> </thead> <tbody> <tr> <td style="padding: 5px;"> <ul style="list-style-type: none"> □ int rows □ int columns □ T** board □ int n_moves ● Board() ● ~Board() ● bool update_board(int x, int y, T symbol) ● void display_board() ● bool is_win() ● bool is_draw() ● bool game_is_over() </td></tr> </tbody> </table>	A Board	<ul style="list-style-type: none"> □ int rows □ int columns □ T** board □ int n_moves ● Board() ● ~Board() ● bool update_board(int x, int y, T symbol) ● void display_board() ● bool is_win() ● bool is_draw() ● bool game_is_over() 	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center; padding: 5px;">C FourByFour_XO_Board</th></tr> </thead> <tbody> <tr> <td style="padding: 5px;"> <ul style="list-style-type: none"> □ int rows □ int columns □ T** board □ int n_moves ● FourByFour_XO_Board() ● ~FourByFour_XO_Board() ● bool validate_token(int x, int y, T symbol) ● bool validate_move(int x, int y, T symbol) ● bool update_board(int x, int y, T symbol) ● void delete_old(int x, int y) ● void display_board() ● bool is_win() ● bool is_draw() </td></tr> </tbody> </table>	C FourByFour_XO_Board	<ul style="list-style-type: none"> □ int rows □ int columns □ T** board □ int n_moves ● FourByFour_XO_Board() ● ~FourByFour_XO_Board() ● bool validate_token(int x, int y, T symbol) ● bool validate_move(int x, int y, T symbol) ● bool update_board(int x, int y, T symbol) ● void delete_old(int x, int y) ● void display_board() ● bool is_win() ● bool is_draw() 	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center; padding: 5px;">C FiveByFiveBoard</th></tr> </thead> <tbody> <tr> <td style="padding: 5px;"> <ul style="list-style-type: none"> □ int rows □ int columns □ T** board □ int n_moves □ int win ● FiveByFiveBoard() ● ~FiveByFiveBoard() ● bool update_board(int x, int y, T symbol) ● void display_board() ● int countSequences(T player) ● int getMoves() ● void assign_move(int x, int y) ● bool is_win() ● bool is_draw() </td></tr> </tbody> </table>	C FiveByFiveBoard	<ul style="list-style-type: none"> □ int rows □ int columns □ T** board □ int n_moves □ int win ● FiveByFiveBoard() ● ~FiveByFiveBoard() ● bool update_board(int x, int y, T symbol) ● void display_board() ● int countSequences(T player) ● int getMoves() ● void assign_move(int x, int y) ● bool is_win() ● bool is_draw()
A Board								
<ul style="list-style-type: none"> □ int rows □ int columns □ T** board □ int n_moves ● Board() ● ~Board() ● bool update_board(int x, int y, T symbol) ● void display_board() ● bool is_win() ● bool is_draw() ● bool game_is_over() 								
C FourByFour_XO_Board								
<ul style="list-style-type: none"> □ int rows □ int columns □ T** board □ int n_moves ● FourByFour_XO_Board() ● ~FourByFour_XO_Board() ● bool validate_token(int x, int y, T symbol) ● bool validate_move(int x, int y, T symbol) ● bool update_board(int x, int y, T symbol) ● void delete_old(int x, int y) ● void display_board() ● bool is_win() ● bool is_draw() 								
C FiveByFiveBoard								
<ul style="list-style-type: none"> □ int rows □ int columns □ T** board □ int n_moves □ int win ● FiveByFiveBoard() ● ~FiveByFiveBoard() ● bool update_board(int x, int y, T symbol) ● void display_board() ● int countSequences(T player) ● int getMoves() ● void assign_move(int x, int y) ● bool is_win() ● bool is_draw() 								
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center; padding: 5px;">C FourInRow</th></tr> </thead> <tbody> <tr> <td style="padding: 5px;"> <ul style="list-style-type: none"> □ int rows □ int columns □ T** board □ int n_moves ● FourInRow() ● ~FourInRow() ● bool update_board(int x, int y, T symbol) ● void display_board() ● bool is_win() ● bool is_draw() </td></tr> </tbody> </table>	C FourInRow	<ul style="list-style-type: none"> □ int rows □ int columns □ T** board □ int n_moves ● FourInRow() ● ~FourInRow() ● bool update_board(int x, int y, T symbol) ● void display_board() ● bool is_win() ● bool is_draw() 	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center; padding: 5px;">C Misere_XO</th></tr> </thead> <tbody> <tr> <td style="padding: 5px;"> <ul style="list-style-type: none"> □ int rows □ int columns □ T** board □ int n_moves □ int win ● Misere_XO() ● ~Misere_XO() ● bool update_board(int x, int y, T symbol) ● void display_board() ● bool is_win() ● bool is_draw() ● bool game_is_over() ● bool getWin() ● int getMoves() ● void assign_move(int x, int y) </td></tr> </tbody> </table>	C Misere_XO	<ul style="list-style-type: none"> □ int rows □ int columns □ T** board □ int n_moves □ int win ● Misere_XO() ● ~Misere_XO() ● bool update_board(int x, int y, T symbol) ● void display_board() ● bool is_win() ● bool is_draw() ● bool game_is_over() ● bool getWin() ● int getMoves() ● void assign_move(int x, int y) 	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center; padding: 5px;">C NumericalTicTacToe</th></tr> </thead> <tbody> <tr> <td style="padding: 5px;"> <ul style="list-style-type: none"> □ int rows □ int columns □ T** board □ int n_moves ● NumericalTicTacToe() ● ~NumericalTicTacToe() ● bool validate_move(int x, int y, T symbol) ● bool update_board(int x, int y, T symbol) ● void display_board() ● bool is_win() ● bool is_draw() </td></tr> </tbody> </table>	C NumericalTicTacToe	<ul style="list-style-type: none"> □ int rows □ int columns □ T** board □ int n_moves ● NumericalTicTacToe() ● ~NumericalTicTacToe() ● bool validate_move(int x, int y, T symbol) ● bool update_board(int x, int y, T symbol) ● void display_board() ● bool is_win() ● bool is_draw()
C FourInRow								
<ul style="list-style-type: none"> □ int rows □ int columns □ T** board □ int n_moves ● FourInRow() ● ~FourInRow() ● bool update_board(int x, int y, T symbol) ● void display_board() ● bool is_win() ● bool is_draw() 								
C Misere_XO								
<ul style="list-style-type: none"> □ int rows □ int columns □ T** board □ int n_moves □ int win ● Misere_XO() ● ~Misere_XO() ● bool update_board(int x, int y, T symbol) ● void display_board() ● bool is_win() ● bool is_draw() ● bool game_is_over() ● bool getWin() ● int getMoves() ● void assign_move(int x, int y) 								
C NumericalTicTacToe								
<ul style="list-style-type: none"> □ int rows □ int columns □ T** board □ int n_moves ● NumericalTicTacToe() ● ~NumericalTicTacToe() ● bool validate_move(int x, int y, T symbol) ● bool update_board(int x, int y, T symbol) ● void display_board() ● bool is_win() ● bool is_draw() 								
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center; padding: 5px;">C Pyramid_X_O_Board</th></tr> </thead> <tbody> <tr> <td style="padding: 5px;"> <ul style="list-style-type: none"> □ int rows □ int columns □ T** board □ int n_moves ● Pyramid_X_O_Board() ● ~Pyramid_X_O_Board() ● bool update_board(int x, int y, T symbol) ● void display_board() ● bool is_win() ● bool is_draw() </td></tr> </tbody> </table>	C Pyramid_X_O_Board	<ul style="list-style-type: none"> □ int rows □ int columns □ T** board □ int n_moves ● Pyramid_X_O_Board() ● ~Pyramid_X_O_Board() ● bool update_board(int x, int y, T symbol) ● void display_board() ● bool is_win() ● bool is_draw() 	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center; padding: 5px;">C Ultimate_XO_Board</th></tr> </thead> <tbody> <tr> <td style="padding: 5px;"> <ul style="list-style-type: none"> □ int rows □ int columns □ T** board □ char small_board[3][3] □ int n_moves ● Ultimate_XO_Board() ● ~Ultimate_XO_Board() ● bool validate_move(int x, int y, char symbol) ● bool update_board(int x, int y, char symbol) ● void display_board() ● bool is_win() ● bool is_draw() </td></tr> </tbody> </table>	C Ultimate_XO_Board	<ul style="list-style-type: none"> □ int rows □ int columns □ T** board □ char small_board[3][3] □ int n_moves ● Ultimate_XO_Board() ● ~Ultimate_XO_Board() ● bool validate_move(int x, int y, char symbol) ● bool update_board(int x, int y, char symbol) ● void display_board() ● bool is_win() ● bool is_draw() 	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center; padding: 5px;">C Word_TicTac</th></tr> </thead> <tbody> <tr> <td style="padding: 5px;"> <ul style="list-style-type: none"> □ int rows □ int columns □ T** board □ int n_moves □ set<string> dic ● Word_TicTac() ● ~Word_TicTac() ● bool update_board(int x, int y, T symbol) ● void display_board() ● bool is_win() ● bool is_draw() </td></tr> </tbody> </table>	C Word_TicTac	<ul style="list-style-type: none"> □ int rows □ int columns □ T** board □ int n_moves □ set<string> dic ● Word_TicTac() ● ~Word_TicTac() ● bool update_board(int x, int y, T symbol) ● void display_board() ● bool is_win() ● bool is_draw()
C Pyramid_X_O_Board								
<ul style="list-style-type: none"> □ int rows □ int columns □ T** board □ int n_moves ● Pyramid_X_O_Board() ● ~Pyramid_X_O_Board() ● bool update_board(int x, int y, T symbol) ● void display_board() ● bool is_win() ● bool is_draw() 								
C Ultimate_XO_Board								
<ul style="list-style-type: none"> □ int rows □ int columns □ T** board □ char small_board[3][3] □ int n_moves ● Ultimate_XO_Board() ● ~Ultimate_XO_Board() ● bool validate_move(int x, int y, char symbol) ● bool update_board(int x, int y, char symbol) ● void display_board() ● bool is_win() ● bool is_draw() 								
C Word_TicTac								
<ul style="list-style-type: none"> □ int rows □ int columns □ T** board □ int n_moves □ set<string> dic ● Word_TicTac() ● ~Word_TicTac() ● bool update_board(int x, int y, T symbol) ● void display_board() ● bool is_win() ● bool is_draw() 								
<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center; padding: 5px;">C Pyramid_X_O_Board</th></tr> </thead> <tbody> <tr> <td style="padding: 5px;"> <ul style="list-style-type: none"> □ int rows □ int columns □ T** board □ int n_moves ● Pyramid_X_O_Board() ● ~Pyramid_X_O_Board() ● bool update_board(int x, int y, T symbol) ● void display_board() ● bool is_win() ● bool is_draw() </td></tr> </tbody> </table>	C Pyramid_X_O_Board	<ul style="list-style-type: none"> □ int rows □ int columns □ T** board □ int n_moves ● Pyramid_X_O_Board() ● ~Pyramid_X_O_Board() ● bool update_board(int x, int y, T symbol) ● void display_board() ● bool is_win() ● bool is_draw() 	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center; padding: 5px;">C Ultimate_XO_Board</th></tr> </thead> <tbody> <tr> <td style="padding: 5px;"> <ul style="list-style-type: none"> □ int rows □ int columns □ T** board □ char small_board[3][3] □ int n_moves ● Ultimate_XO_Board() ● ~Ultimate_XO_Board() ● bool validate_move(int x, int y, char symbol) ● bool update_board(int x, int y, char symbol) ● void display_board() ● bool is_win() ● bool is_draw() </td></tr> </tbody> </table>	C Ultimate_XO_Board	<ul style="list-style-type: none"> □ int rows □ int columns □ T** board □ char small_board[3][3] □ int n_moves ● Ultimate_XO_Board() ● ~Ultimate_XO_Board() ● bool validate_move(int x, int y, char symbol) ● bool update_board(int x, int y, char symbol) ● void display_board() ● bool is_win() ● bool is_draw() 	<table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th style="text-align: center; padding: 5px;">C Word_TicTac</th></tr> </thead> <tbody> <tr> <td style="padding: 5px;"> <ul style="list-style-type: none"> □ int rows □ int columns □ T** board □ int n_moves □ set<string> dic ● Word_TicTac() ● ~Word_TicTac() ● bool update_board(int x, int y, T symbol) ● void display_board() ● bool is_win() ● bool is_draw() </td></tr> </tbody> </table>	C Word_TicTac	<ul style="list-style-type: none"> □ int rows □ int columns □ T** board □ int n_moves □ set<string> dic ● Word_TicTac() ● ~Word_TicTac() ● bool update_board(int x, int y, T symbol) ● void display_board() ● bool is_win() ● bool is_draw()
C Pyramid_X_O_Board								
<ul style="list-style-type: none"> □ int rows □ int columns □ T** board □ int n_moves ● Pyramid_X_O_Board() ● ~Pyramid_X_O_Board() ● bool update_board(int x, int y, T symbol) ● void display_board() ● bool is_win() ● bool is_draw() 								
C Ultimate_XO_Board								
<ul style="list-style-type: none"> □ int rows □ int columns □ T** board □ char small_board[3][3] □ int n_moves ● Ultimate_XO_Board() ● ~Ultimate_XO_Board() ● bool validate_move(int x, int y, char symbol) ● bool update_board(int x, int y, char symbol) ● void display_board() ● bool is_win() ● bool is_draw() 								
C Word_TicTac								
<ul style="list-style-type: none"> □ int rows □ int columns □ T** board □ int n_moves □ set<string> dic ● Word_TicTac() ● ~Word_TicTac() ● bool update_board(int x, int y, T symbol) ● void display_board() ● bool is_win() ● bool is_draw() 								

A Player	C FourByFour_XO_Player	C FiveByFivePlayer
<ul style="list-style-type: none"> □ string name □ T symbol □ Board<T>* boardPtr 	<ul style="list-style-type: none"> □ FourByFour_XO_Board<T>* brd; ● FourByFour_XO_Player(string name, T symbol) ● void getmove(int& x, int& y) 	<ul style="list-style-type: none"> □ FiveByFiveBoard<T>* brd ● FiveByFivePlayer(string name, T symbol) ● void getmove(int& x, int& y)
C FourInRow_Player	C Misere_XO_Player	C NumericalTicTacToe_Player
<ul style="list-style-type: none"> ● FourInRow_Player(string name, T symbol) ● void getmove(int& x, int& y) 	<ul style="list-style-type: none"> □ Misere_XO<T>* brd ● Misere_XO_Player(string name, T symbol) ● void getmove(int& x, int& y) 	<ul style="list-style-type: none"> □ set<int> used ● NumericalTicTacToe_Player(string name, T symbol) ● void get_value(T &symbol) ● void getmove(int& x, int& y)
C Pyramid_X_O_Player	C Ultimate_XO_Player	C Word_TicTac_Player
<ul style="list-style-type: none"> ● Pyramid_X_O_Player(string name, T symbol) ● void getmove(int& x, int& y) 	<ul style="list-style-type: none"> ● Ultimate_XO_Player(string name, T symbol) ● void getmove(int& x, int& y) 	<ul style="list-style-type: none"> ● Word_TicTac_Player(string name, T symbol) ● void get_value(T &symbol) ● void getmove(int& x, int& y)
A RandomPlayer	C FourByFour_XO_Random_Player	C FivebyFiveRandomPlayer
<ul style="list-style-type: none"> □ string name □ T symbol □ Board<T>* boardPtr 	<ul style="list-style-type: none"> □ FourByFour_XO_Board<T>* brd ● FourByFour_XO_Random_Player(T symbol) ● void getmove(int& x, int& y) 	<ul style="list-style-type: none"> ● FivebyFiveRandomPlayer(T symbol) ● void getmove(int& x, int& y)
C FourInRow_Random_Player	C Misere_XO_Random_Player	C NumericalTicTacToe_Random_Player
<ul style="list-style-type: none"> ● FourInRow_Random_Player(T symbol) ● void getmove(int& x, int& y) 	<ul style="list-style-type: none"> ● Misere_XO_Random_Player(T symbol) ● void getmove(int& x, int& y) 	<ul style="list-style-type: none"> □ set <int> used □ NumericalTicTacToe<T>* brd ● NumericalTicTacToe_Random_Player(T symbol) ● void getmove(int& x, int& y)
C Pyramid_X_O_Random_Player	C Ultimate_XO_Random_Player	C Word_TicTac_Random_Player
<ul style="list-style-type: none"> ● Pyramid_X_O_Random_Player(T symbol) ● void getmove(int& x, int& y) 	<ul style="list-style-type: none"> □ Ultimate_XO_Board<T>* brd ● Ultimate_XO_Random_Player(T symbol) ● void getmove(int& x, int& y) 	<ul style="list-style-type: none"> ● Word_TicTac_Random_Player(T symbol) ● void getmove(int& x, int& y)

Work Breakdown

Basic Assignment

- Abdelrahman: Games 1 and 4
- Malak: Games 2 and 5
- Yassin: Games 3 and 6
- Teamwork: Games 7 and 8 and menu

Code Reviews

Yassin

- Found that *Word Tic-Tac-Toe* wasn't searching for valid words in normal vector efficiently so we used `set()` and `count()` that does binary search on the valid words.
- Used `srand()` to seed time so the random pattern becomes unpredictable in random player.

Abdelrahman

- Adjusted `win` attribute and `is_win()` functions in *5x5* and *Misere Tic-Tac-Toe* to make sure that the right player wins the game
- Added some checking in *4x4*'s *RandomPlayer* so that it makes a valid moves and chooses a valid token.

Malak

- In *Numerical Tic-Tac-Toe* the game lets a player win even if he only got two numbers with sum of 15 instead of 3, so I added a condition to check if an empty cell is in the winning sequence in `is_win()` function.

Bonus Description

- We implemented a GUI application that shows the group games *4x4 Tic-Tac-Toe* and *Ultimate Tic-Tac-Toe*
- Qt library was used for implementation
- Work breakdown:
 - Abdelrahman: Designed the UI
 - Yassin: Implemented the UI and enhanced its functionality
 - Malak: Implemented the logic of the games

GitHub Repo

The screenshot shows a GitHub repository page for a project named 'A2_S7_Task23_B_20230465_20230220_20230416'. The repository has 28 commits. The commit history includes:

- .idea (V3) - 2 weeks ago
- cmake-build-debug (menu added) - yesterday
- .gitattributes (Initial commit) - 3 weeks ago
- .gitignore (Create FourInRow) - 2 weeks ago
- 3x3X_O.h (Update) - 2 weeks ago
- 4x4_XO.h (Merge branch 'main' of https://github.com/yassinelsawy/A...) - 5 hours ago
- 5x5_XO.h (Merge branch 'main' of https://github.com/yassinelsawy/A...) - 5 hours ago
- Board Game Classes.html (base class files added) - 2 weeks ago
- BoardGame_Classes.h (Update) - yesterday
- CMakeLists.txt (Update) - yesterday
- FourInRow.h (Testing and debugging) - 5 hours ago
- Misere_XO.h (Testing and debugging) - 5 hours ago

The repository has 0 stars, 1 watching, and 0 forks. It has no releases published and no packages published. Contributors include abdelrahmany, yassinelsawy, and MalakAmrAhmdd.

GUI Video Link

<https://youtu.be/MM9XJIXEb50>