Cairo University, Faculty of Artificial Intelligence

FACULTY OF COMPUTERS AND ARTIFICIAL INTELLIGENCE, CAIRO UNIVERSITY

CS213: Programming II
Year 2024-2025
First Semester

Assignment 2 – Version 3.0

Course Instructors:

Dr. Mohammad El-Ramly

Revision History

Version 1.0	By Dr Mohammed El-Ramly	26 Nov. 2022	Main Doc
Version 2.0	By Dr Mohammed El-Ramly	26 Nov. 2023	Revised
Version 3.0	By Dr Mohammed El-Ramly	6 Nov. 2024	Added new games
Version 4.0	By Dr Mohammad El-Ramly	16 Nov 2024	2 x 3 + 2 x 1 = 8 games per team

CS213: Object Oriented Programming
Assignment 2 (13 marks + 4 bonus) – Version 3.0

Cairo University, Faculty of Artificial Intelligence

Objectives

This assignment aims to (1) teach OOP concepts in C++, (2) using OOP to build systems with intermediate complexity, (3) Templates and STL, (4) Recursion and Backtracking, (5) Code Reuse.

Instructions

1. **Part1: Deadline is 15 Nov 2024. (Task1.1 - One problem from the sheet)**
2. **Part2: Deadline is 25 Nov 2024. (Task1.2 - Another problem from the sheet)**
3. **Part3: Deadline is 5 Dec 2024. (Tasks2,3,4,5 - The gaming application)**
4. **Weight is 13 marks** + 4 **bonus** marks.
5. Students will forms teams of **three** students **from the same lab/section.**
6. Please submit **only work that you did yourself**. If you copy work from your friend or book or the net **you will fail the course**. تسليم حلول منقولة من أى مصدر يؤدى إلى الرسوب فى هذا المقرر لا تغش الحل أو تنقله من أى مصدر و اسألنى فى أى شئ لا تفهمه لكن لا تنقل الحلول من النت أو من زملائك أو أى مكان

Task 0 (0 marks) - Individual / Team

1. Review OOP / STL / Templates / Recursion / Backtracking.
2. Review all code examples given in lecture and in classroom page.
3. Create a **private NOT public GitHub** repo for the project and **use it for development.**

Task 1 (4 marks) - Individual

1. Given sheet 2 of problems, OOP / STL / Templates / Recursion / Backtracking, each team member will solve 2 problems from the sheet. Smallest ID (20230900 > 20230746) will solve problems 1 and 4 and second one will solve 2 and 5 and the biggest ID will solve 3 and 6.

What to deliver?

- Working code in standard C++ and using STL not using third-party libraries.
- Name your file A2_SectionNum_ID_SheetPbXX.cpp (XX is pb num)
- Each problem will be loaded ONLY as .cpp file in a separate place in Classroom.

Task1 (8 marks) – Individual / Group - OOP Design and Development

You are given a generic implementation of a board game framework that can be used to easily and quickly develop board games; it shows the power of OOP in modularity and reusability. **You do not have to build every game from scratch. You can inherit from the framework classes** to make a new game.

It consists of the following generic classes (They are put in a header **`BoardGame_Classes.h`** because we cannot separate templates into a header file and implementation files.):

1- **`GameManager`** that owns a board and 2 players and runs the game and swap turns.

2- **`Player`** that represents a generic game player that has a name and symbol and does x, y moves

3- An abstract game **`Board`** with functions to be defined in children for rules of the game.

4- A **`RandomPlayer`** that returns a random x, y move.

CS213: Object Oriented Programming
Assignment 2 (13 marks + 4 bonus) – Version 3.0

Cairo University, Faculty of Artificial Intelligence

This framework is suitable for building multiple board games like X-O and similar ones. You are given an example X-O game implementation that includes the following classes:

5- **`X_O_Board`** represents the board of X-O and the rules for that game.
6- **`X_O_Player`** represents an X-O player that has a name and a symbol and does X_O moves.
7- **`X_O_Random_Player`** represents a random player that can return a random X_O move.

You are also given another demo that includes an AI player for those who want to do bonus part. The run method in the **`GameManager`** works as follows for any game:

```
function run():
    // Initialize variables to contain player moves
    x, y

    // Display the initial game board
    display_board(board)

    // While the game is not over
    while game_is_over(board) is false:

        // For each player (assumed two players, index 0 and 1)
        for each player in [0, 1]:

            // Get the player's move (x, y coordinates)
            get_move(player, x, y)

            // Ensure the move is valid by updating the board
            while update_board(board, x, y, player.symbol) is false:
                // If invalid, ask for a new move
                get_move(player, x, y)

            // Display the updated board after the move
            display_board(board)

            // Check if the current player has won
            if is_win(board) is true:
                print(player.name + " wins")
                return // End the game

            // Check if the game is a draw
            if is_draw(board) is true:
                print("Draw!")
                return // End the game
```

Team will develop 8 games using the framework and integrate them in one app with one menu. Team will add their IDs and get the smallest digit. **The smallest ID will do games 1 & 4, the next will do game 2 & 5 and the third will do game 3 & 6. If smallest digit is odd, the team together will do games 7 and 8. If it is even, the team together will do games 8 and 9. Ex: 20230001 + 20230002 + 20230003 = xxxx0006 = even. 20230001 will do gamea 1 & 4, 20230002 will do games 2 & 5 and 20230006 will do game 3 & 6 and all team will do games 8 and 9.** Team members are expected to help each other but not do work of others.

Constraints: **You are not allowed to alter the existing code.** Instead, you should create new classes that inherit from base classes to define the rules and behaviors of the new games. This exercise highlights the Open/Closed Principle, which states that software entities should be open for extension but closed for modification. Your task is to extend the functionality without changing the existing code.

For each game, you will need to inherit from the **`Board`** class and to create a special board for this game. You might also need to inherit from **`Player`** to create a special player for this game. You also need to create a random computer player for each game that chooses a **correct** random move.

See the given example of X_O game.

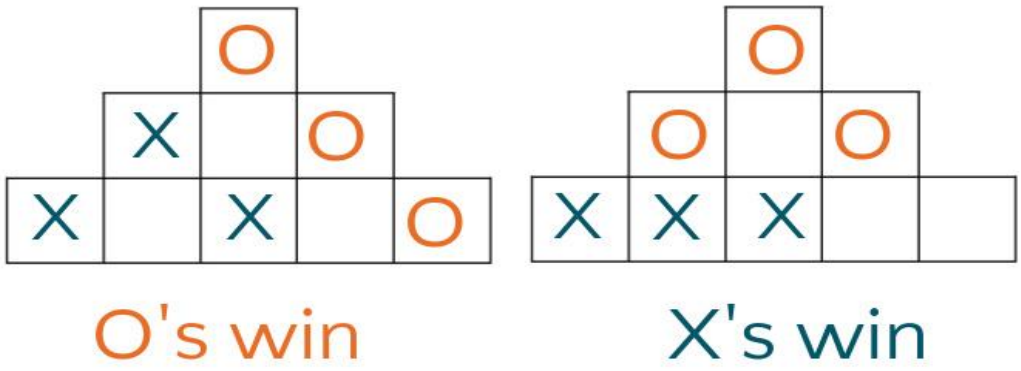
Game Descriptions

Game descriptions are taken from: https://www.whatdowedoallday.com/tic-tac-toe-variations/

1- Pyramic Tic-Tac-Toe

The game board is shaped like a pyramid. Five squares make the base, then three, then one. Players take turns marking Xs and Os as in traditional tic-tac-toe.

Winning: The first player to get three-in-a-row vertically, horizontally, or diagonally wins. See two examples of winning positions, below.



2- Four-in-a-row

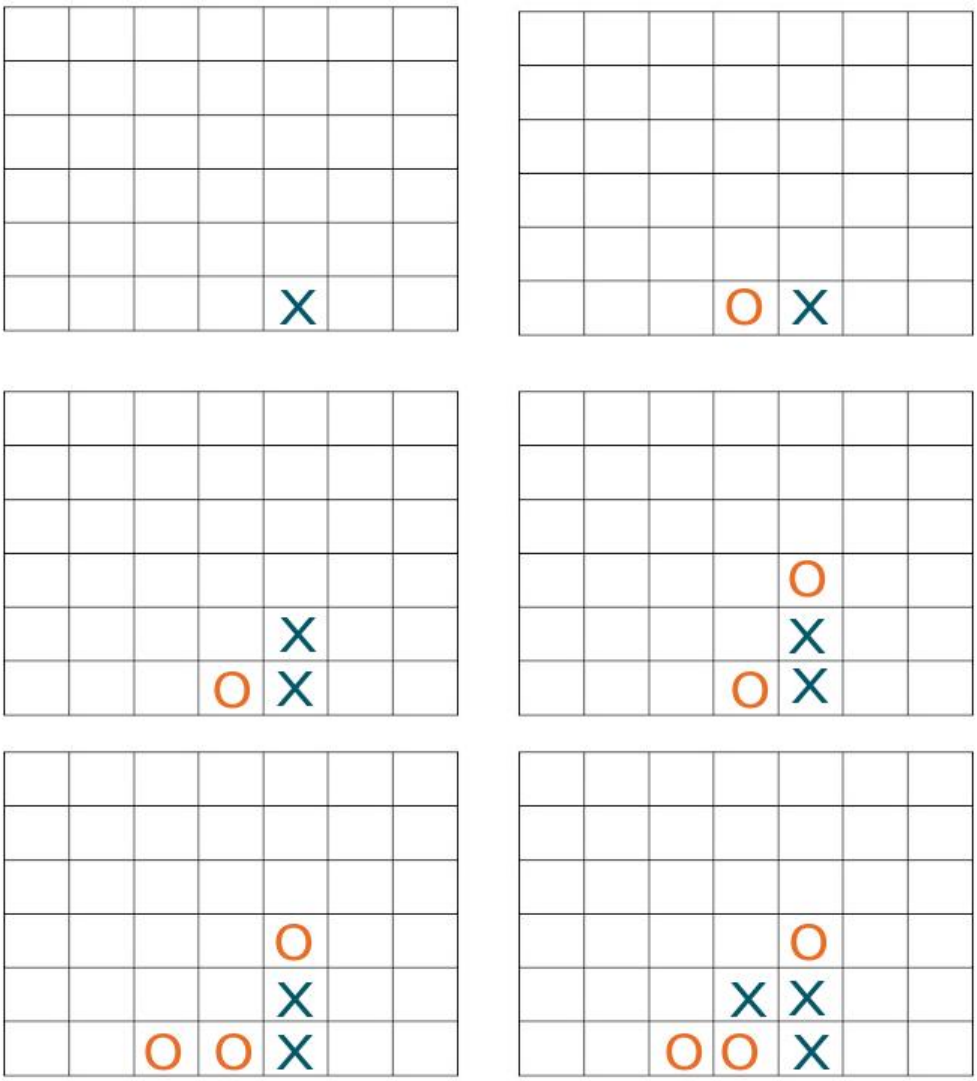
You will recognize four-in-a row as a two-dimensional version of the classic game, Connect Four. The game board consists of a 7 x 6 grid. Seven columns of six squares each. Instead of dropping counters as in Connect Four, players mark the grid with Xs and Os as in tic-tac-toe.

Rules: The first player places an X in the bottom square of any column. Taking turns, players make their mark in any column, as long as it is in the lowest square possible. See image below for an example of possible first six moves.

Winning: The first player to get four-in-a-row vertically, horizontally, or diagonally wins.

CS213: Object Oriented Programming
Assignment 2 (13 marks + 4 bonus) – Version 3.0

Cairo University, Faculty of Artificial Intelligence



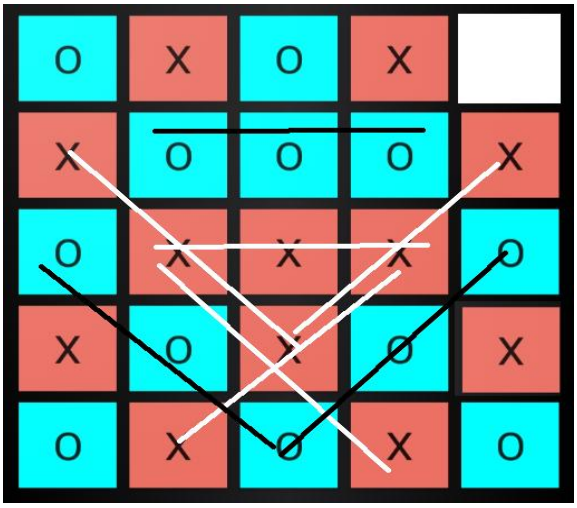
Example play for six moves of four-in-a-row

3- 5 x 5 Tic Tac Toe

This tic-tac-toe variation is played on a 5 x 5 grid. As in the traditional game, players are Xs or Os.

Rules: Players take turns placing an X or an O in one of the squares until all the squares except one are filled. (Each player has 12 turns for a total of 24 squares.)

Winning: Count the number of three-in-a-rows each player has. Sequences can be vertically, horizontally, or diagonally. Whoever has the most, wins. Can one mark be counted in more than one three-in-a-row sequence? Decide ahead of time, yes or no. It is easier in implementation to allow counting more than once.

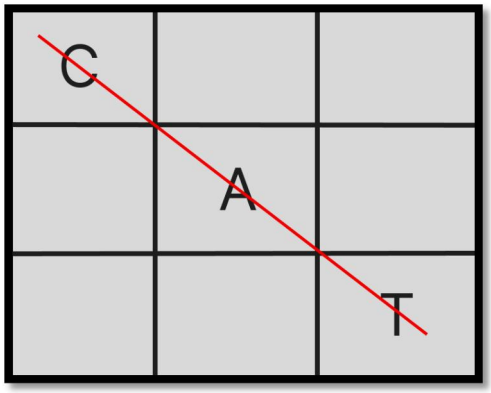


Example of finished 5 x 5 game and X wins with 5 times 3Xs and O loses with 3 times 3Os

CS213: Object Oriented Programming
Assignment 2 (13 marks + 4 bonus) – Version 3.0

Cairo University, Faculty of Artificial Intelligence

4- Word Tic-tac-toe



Word Tic-tac-toe is an innovative twist on the classic Tic-tac-toe game. Instead of using "X" or "O", players place letters on a 3x3 grid to form valid words. This version adds a linguistic challenge to the traditional game mechanics. Players aim to form a valid word with the letters they place on the board. Words can be formed horizontally, vertically, or diagonally.

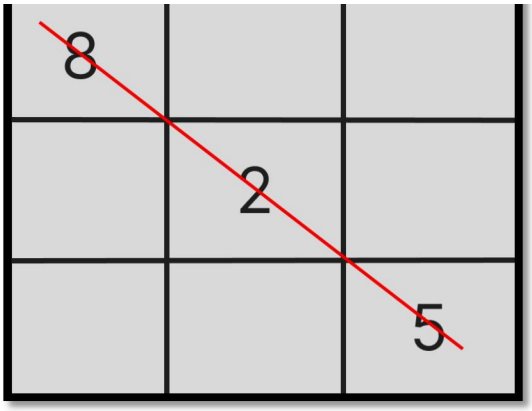
Rules: Each player takes turns placing one letter on the board. A player must try to form a valid word with each move. Players can build upon existing letters to form words, provided that the resulting sequence is a valid word.

Winning: The game is won by forming a valid word horizontally, vertically, or diagonally. If the board fills up without a valid word being formed, the game ends in a draw.

Additional Details:

- A file named dic.txt will be provided, containing all valid 3-letter words. Your win-checking function should **efficiently** search this file to determine if a valid word is formed after each move.

5- Numerical Tic-Tac-Toe



Numerical Tic-Tac-Toe offers a mathematical twist to the classic Tic-Tac-Toe game. Instead of the traditional "X" and "O", players use numbers to add an element of strategic calculation. The objective is to achieve a sum of 15 with three numbers in a row, column, or diagonal.

Rules: Player 1 typically starts and uses odd numbers (1, 3, 5, 7, 9), while Player 2 uses even numbers (2, 4, 6 8). Players alternate turns, placing one number in an empty cell on the board. Each number can only be used once.

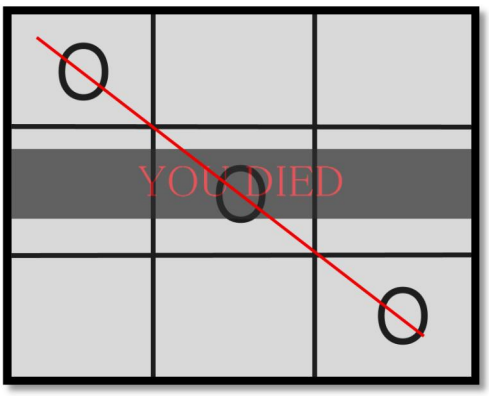
Winning: A player wins by placing three numbers in a row, column, or diagonal that add up to 15. If all cells are filled without achieving the sum of 15 in any line, the game ends in a draw.

CS213: Object Oriented Programming
Assignment 2 (13 marks + 4 bonus) – Version 3.0

Cairo University, Faculty of Artificial Intelligence

6- Misere Tic Tac Toe

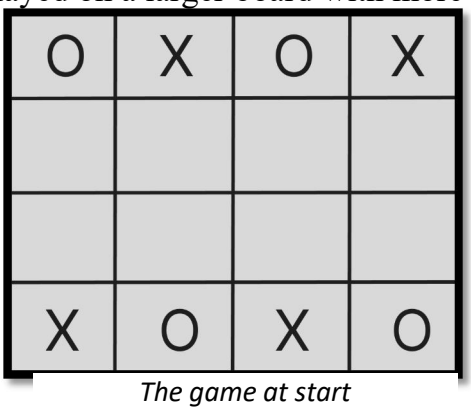
Misere Tic Tac Toe, also known as Inverse Tic Tac Toe or Toe Tac Tic, is a unique twist on the classic game. In this version, the objective is to avoid getting three marks in a row. The game flips the traditional win condition on its head, making every move a strategic decision to prevent losing.



Rules: The game is played on a standard 3x3 Tic-Tac-Toe grid. The goal is to avoid placing three of your marks in a row, column, or diagonal. The player who ends up with three marks in a row loses the game. If all cells are filled without either player aligning three marks in a row, the game ends in a draw.

7- 4 x 4 Tic-Tac-Toe

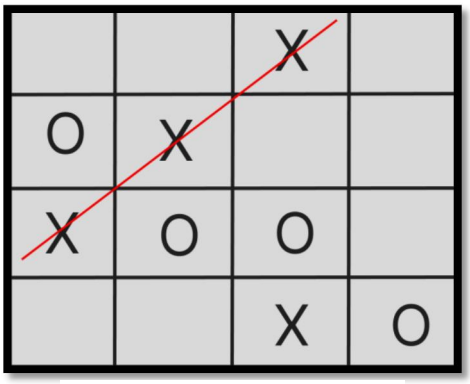
4 x 4 Tic-Tac-Toe is an extended version of the classic game, played on a larger board with more complex strategies. Each player has four tokens and aims to align three tokens in a row to win. This game introduces new movement rules and strategic depth to the traditional Tic-Tac-Toe. The game is played on a 4x4 grid. Each player has four tokens. Tokens are placed in specific starting positions: two tokens on opposite sides of the board for each player. (as shown in image).



The game at start

Rules: Players alternate turns, moving one of their tokens to an immediately adjacent open square. Tokens can be moved horizontally or vertically but not diagonally. Tokens may not jump over other tokens. The goal is to align three of your tokens in a row. This can be achieved horizontally, vertically, or diagonally.

Winning: The first player to get three tokens in a row wins the game. The alignment can be in any direction: horizontal, vertical, or diagonal.

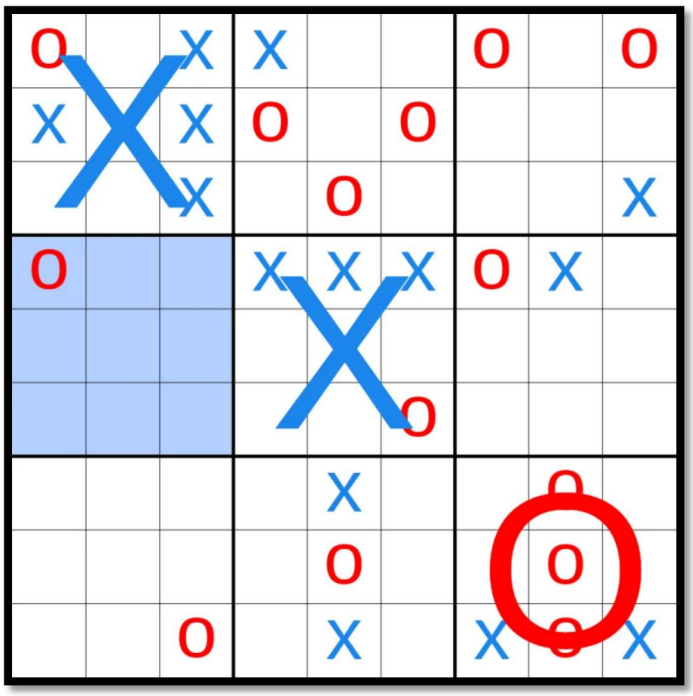


The game at end

CS213: Object Oriented Programming
Assignment 2 (13 marks + 4 bonus) – Version 3.0

Cairo University, Faculty of Artificial Intelligence

8- Ultimate Tic Tac Toe

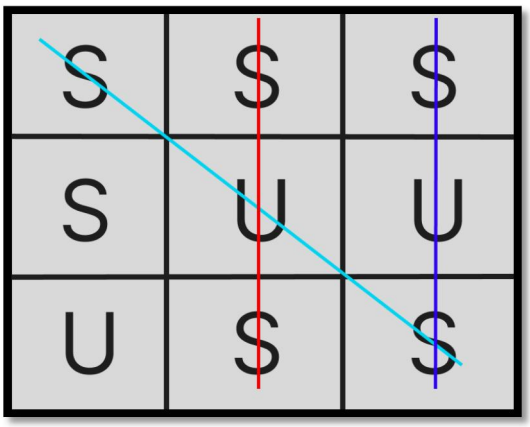


Ultimate Tic Tac Toe is an expansion of the classic game, where players engage in a meta-game of Tic Tac Toe within a 3x3 grid of smaller Tic Tac Toe boards. The goal is to win three smaller games in a row to claim victory on the main board, adding layers of complexity to the traditional game. The game is played on a large 3x3 grid, where each cell contains a smaller 3x3 Tic Tac Toe board.

Rules: Player 1 starts by choosing any of the nine smaller Tic Tac Toe boards to play on. Players alternate turns, playing Tic Tac Toe on the smaller boards. The winner of each smaller board claims that space on the main board. The winner of the smaller board replaces that board with their symbol (X or O) on the main board.

Winning: The first player to win three smaller boards in a row on the main 3x3 grid wins the Ultimate Tic Tac Toe game. The winning row on the main board can be horizontal, vertical, or diagonal.

9- SUS



The SUS game is a simple game played on a 3x3 grid. The objective is to form the sequence "S-U-S" by placing letters in the grid. Players must carefully plan their moves to create as many SUS sequences as possible while blocking their opponent from doing the same.

Rules: The goal is to create the sequence "S-U-S" in a straight line, which can be achieved diagonally, horizontally, or vertically. Players take turns placing either an "S" or a "U" in any empty square on the grid. A player must use the same letter for each turn. If a player successfully creates an "S-U-S" sequence, they take a point.

Winning: The game continues until all squares are filled or no more "S-U-S" sequences can be created. The player who creates the most "S-U-S" sequences wins the game.

CS213: Object Oriented Programming
Assignment 2 (13 marks + 4 bonus) – Version 3.0

Cairo University, Faculty of Artificial Intelligence

Group Task 3 (-1 mark if not done) App Integration & (1 mark) Code quality

-1 mark. The team will integrate their work and deliver **one single app that gives a menu of choices** with the games available and if the user chooses a game, it opens the board for them to play against another player or a random computer player.

1 mark. Code review is an essential part of software development to ensure its quality and readability and reduce the bugs. Each team member will read the code of the another member and will write in the report comments on errors / quality issues he found.

- Read more about code reviews and their tools also. See
  https://kinsta.com/blog/code-review-tools/
  https://github.com/joho/awesome-code-review
- Review the code using this checklist or similar ones **and write in your report any violations occur and which part file and line number(s).** https://www.codereviewchecklist.com/

Bonus Tasks 4,5 (2 + 2 marks) – GUI / AI Players

T4 - Two marks. For your game, inherit a smart computer player class from the given player class instead of the random player (See demo using AI). It should be able to choose the best movement given the current board status and play against human in order to win.

Read about min-max and backtracking algorithms and choose a suitable algorithm or try a suitable heuristic algorithm that chooses the best move for the computer. Try several game plays to ensure your algorithm works well.

Notice that the current X-O game demos: (1) one has a totally random computer player that chooses random moves and (2) the other has an AI-based player as an example.

T5 - Two marks. For developing a GUI for your game app that allows the players to choose the game and to play against a human, a random computer player or an AI player (if available)

What to deliver

1- **Written original non-cheated code in standard C++ not using third-party libraries** (Unless u do GUI) representing the whole gaming app that has your team games**.**
2- Name your file **A2_Section_Task2345_YourGroup_YourIDs.zip** (add 2 and 3 and/or 4, 5 depending on what you did). **NO EXE. Do not upload exe or object files.**
3- **A pdf** report, including:
   a. Cover page with names, emails, ID, logo, course name, prof name, section
   b. A description of the classes you created for each game and a UML class diagram.
   c. A work breakdown table saying **who did what.**
   d. A report of the **quality of the code** of the other members and what mistakes you found there and how you reviewed it.
   e. Image of a **private** project in **GitHub** to collaborate on code.
4- Code of the game app organized in files in one directory, and all app with bonus in another directory.
5- **All team members must understand the details of all programs** and be able to explain it.

CS213: Object Oriented Programming
Assignment 2 (13 marks + 4 bonus) – Version 3.0

Cairo University, Faculty of Artificial Intelligence

What to deliver for bonus:

1- Submit the original game code.
2- Submit the modified game code with bonus options (GUI / AI or both).
3- Add a link to a **public video** showing a demo of the GUI you made and the options in it.
4- **Explain in the report what you did for bonus, what tools you used, how it works, and who did what.**

Marking Criteria

1.	2 x 2	For a correct working solution for each **individual problem** (individual)
2.	**2 x 2**	For a working original **individual game** that plays correctly according to the rules.
3.	4	For correctly working original **group games.** (group)
4.	1	For **code reviews of all codes** (group)
5.	-1	For **not integrating** the games together in one program with a menu (group)
6.	-1	For not using GitHub. (group)
7.	-1	For not good quality report. (group)
8.	2	For a working smart **AI player** for your games with a suitable algorithm (individual)
9.	2	For a working **GUI** for the whole app and every game (group)
10.	-13	For cheating any part of the code, even 5 lines or generated code.