Assignment3

Faculty of Computers and Artificial Intelligence
Cairo University

CS213

Assignment 3

Name	ID	Game
Youssef Mohamed Abelsamea	20220407	3- 5 x 5 Tic Tac Toe
Omar Amin	20200341	1- Pyramic Tic-Tac-Toe
Youssef Amr	20220401	2- Four-in-a-row

Assignment3

- # Explain algorithm for Pyramic Tic-Tac-Toe:

1. X_O_Board Class:

Constructor (X_O_Board::X_O_Board()): Initializes a 3x5 board (n_rows = 3, n_cols = 5) with all elements set to 0.

update_board Method: Updates the board based on player moves. Checks if the given coordinates (x, y) are valid and if the corresponding board cell is empty (0).
The board has specific rules for allowing a move based on the values of x and y.

display_board Method: Displays the current state of the board in a formatted manner. Special formatting is applied to certain cells based on their positions.

is_winner Method: Checks for a winner by examining rows, columns, and diagonals. Uses bitwise AND (&) to check if three consecutive cells are marked by the same player.

is_draw Method: Checks if the game is a draw by verifying if the number of moves is 9 and there is no winner.

game_is_over Method: Checks if the maximum number of moves (9) has been reached.

2. GameManager Class:

Constructor (GameManager::GameManager()): Takes a pointer to a board and an array of two player pointers. Initializes the GameManager with the provided board and players.

run Method: Runs the game loop until it's over.
Players take turns making moves, and the board is displayed after each move. Checks for a winner or a draw after each move.

Assignment3

3. RandomPlayer Class:

Constructor (RandomPlayer::RandomPlayer()): Inherits from the Player class. Initializes a computer player with a random move generator.

get_move Method: Generates random moves for the computer player.

4. Player Class:

Constructors (Player::Player()): Constructor for a human player, and another for a computer player. Initializes player name and symbol.

get_move Method: For human players, prompts the user to enter their move (x and y coordinates).

to_string Method: Provides a string representation of the player (used for displaying the winner).

get_symbol Method: Returns the player's symbol.

Assignment3

- # Explain algorithm for Four-in-a-row:

1. X_O_Board Class:

Constructor (X_O_Board::X_O_Board()): Initializes a 6x7 board with all elements set to 0.

update_board Method: Updates the board based on player moves. Checks if the given coordinates (x, y) are valid, the corresponding board cell is empty (0), and the cell below is either occupied or it's the bottom row. Marks the cell with the player's symbol (converted to uppercase) and increments the move count.

display_board Method: Displays the current state of the board in a formatted manner. Shows the position of each cell and its content.

is_winner Method: Checks for a winner by examining rows, columns, and diagonals. The game is won if there are four consecutive 'X' or 'O' symbols in a row, column, or diagonal.

is_draw Method: Checks if the game is a draw by verifying if the number of moves is 42 (total number of cells) and there is no winner.

game_is_over Method: Checks if the maximum number of moves (42) has been reached.

2. GameManager Class:

Constructor (GameManager::GameManager()): Takes a pointer to a board and an array of two player pointers. Initializes the GameManager with the provided board and players.

run Method: Runs the game loop until it's over.
Players take turns making moves, and the board is displayed after each move. Checks for a winner or a draw after each move.

Assignment3

- # Explain algorithm for 5x5 Tic-Tac-Toe:

1. X_O_Board Class:

Constructor (X_O_Board::X_O_Board()): Initializes a 5x5 board with all elements set to 0.

update_board Method: Updates the board based on player moves. Checks if the given coordinates (x, y) are valid, the corresponding board cell is empty (0), and returns true if the move is invalid. If the move is valid, marks the cell with the player's symbol (converted to uppercase), increments the move count, and returns false.

display_board Method: Displays the current state of the board in a formatted manner. Shows the position of each cell and its content.

is_winner Method: Checks for a winner by examining rows, columns, and diagonals. Counts the number of winning combinations for 'X' and 'O' in rows, columns, and diagonals. Outputs the result based on the counts and declares the winner or a draw.

game_not_over Method: Checks if the game is not over based on the number of moves (limited to 24).

2. EasyPlayer Class: Inherits from the Player class.

Constructor (EasyPlayer::EasyPlayer()): Initializes an Easy Player with the symbol 'o'.

get_move Method: Generates random moves for the Easy Player.

4. GameManager Class:

Constructor (GameManager::GameManager()): Takes a pointer to a board and an array of two player pointers. Initializes the GameManager with the provided board and players.

run Method: Runs the game loop until it's over.
Players take turns making moves, and the board is displayed after each move. Checks for a winner or draw after each move.

Game Rules: The game is played on a 5x5 board.
Two players, one human and one computer, take turns making moves.
The game continues until there's a winner, a draw (after 24 moves), or the user chooses to exit.

Note: There's a mistake in the is_winner method where the comparison operator & should be replaced with &&. The SmartPlayer class currently does not demonstrate any smart strategy; it simply makes a move at position (0, 0). This code provides a basic structure for a two-player game with a human player and a computer player, where the computer player's strategy is simple, and the board is displayed after each move.

- **Explain algorithm for Task 3:**

Collect functions on each game and connect by char call “Game”.
Because each game has different algorithm.