**Project Name: FAM**

**Logo

Description automatically generated**

**Project members:**

**Faisal Minawi (**[**fmm34@mail.aub.edu**](mailto:fmm34@mail.aub.edu)**)**

**Mohammad El Kadri (**[**mae126@mail.aub.edu**](mailto:mae126@mail.aub.edu)**)**

**Ali Moussawi (**[**amm97@mail.aub.edu**](mailto:amm97@mail.aub.edu)**)**

**Firas Noun (**[**fbn02@mail.aub.edu**](mailto:fbn02@mail.aub.edu)**)**

**Project Description:**

This project aims at building a connect 4 game which has two different modes (Single Player and Multiplayer). This program also has 4 difficulties (Beginner, Easy, Medium, Hard).

**Project Implementation:**

1. int random\_move (int board [] [BOARD\_SIZE\_VERT], int computer\_num): This method is used to choose a random move when the player chooses to play against the beginner bot.
2. int player\_move (int board [] [BOARD\_SIZE\_VERT], int player\_num):

This method takes a coordinate from the player and updates the board if possible. The updated number is determined by the player\_num {1,2} and the coordinated depends on the column available.

1. int check\_winner (int board [] [BOARD\_SIZE\_VERT], int last\_move):

This function is a helper function for the check\_win\_tie in which it checks horizontally, vertically, and obliquely to help that function determine if there exists a winner by returning an integer that is the last “1” or “2” in the 4 connected. If the latter happened, the below function determines if there is a winner or not.

1. bool check\_win\_or\_tie (int board [] [BOARD\_SIZE\_VERT], int last\_move):

This function uses the previous function and announces the winner according to the last move. It returns true if there is a winner and false otherwise. Also, it prints the winner.

Note: used for multiplayer mode.

1. bool check\_win\_or\_tie\_single (int board [] [BOARD\_SIZE\_VERT], int last\_move):

This function is the same as the previous function. It checks the winner in the single player mode. The only difference is that the previous function displays the information directly. The winner {computer – player} of the single player mode is printed in the main function.

1. bool is\_column\_full (int board [] [BOARD\_SIZE\_VERT], int m):

This method returns a Boolean and checks whether a column in the board is full, or if it can accept another coordinate.

1. void update\_board (int board [] [BOARD\_SIZE\_VERT], int m, int player\_num): This method updates the board with each move made to it.
2. int best\_move(int board [] [BOARD\_SIZE\_VERT], int computer\_num, int depth): This function is the most important function in our program in which it identifies the best move that can be done. This function uses the Minimax algorithm.
3. int scoring(int score[][BOARD\_SIZE\_VERT]): Helper function for the Minimax function. Returns the score of the virtual created score board.
4. int calcRow(int board[][BOARD\_SIZE\_VERT], int col, int player): Helper function to the minimax algorithm that helps in calculation of the number of the row that is the last empty cell in the chosen column
5. void MinMax\_Recur(int MinMax, int player, int depth, int minmaxBoard[][BOARD\_SIZE\_VERT], int \*m, int \*score): Is a helper function for the best move method and it includes the minimax algorithm.

A minimax algorithm is a recursive software designed to determine the best strategy that maximizes winning chances while minimizing the likelihood of losing a game. Minimax can be shown as an investigation of a game tree's nodes to determine the optimal action to make. We can change the difficulty of the bot through changing the depth of the algorithm. The greater the depth, the harder the game becomes to get beaten.

1. void display\_board(int board[][BOARD\_SIZE\_VERT]): This function is used to the display the board the is constantly updated the coordinates given by the user or the bot.

**Main Function:**

In the main function, we start by asking the users which mode they would like to play single player or multiplayer.

If single player mode is chosen: The player is asked to choose the difficulty they would like to play with (Beginner, Easy, Medium, Hard).

Beginner: random moves will be selected by the computer to minimize the chance of it blocking or winning the game so that the beginner player learns how to play.

Easy – Medium – Hard: the computer uses the best move function that uses the selected depth. The easy has a lower depth of 2, medium of depth 5, and hard has the greatest depth of 8.

After that, a coin is tossed to determine whether the bot or the player will start.

If multiplayer mode is chosen, a coin is flipped to determine whether player 1 or player 2 starts playing. Then the players will start with their moves until a player wins.

In all options, if the user enters a wrong argument, they will be notified and asked to enter a valid input.

The game either ends in a win or in a tie!