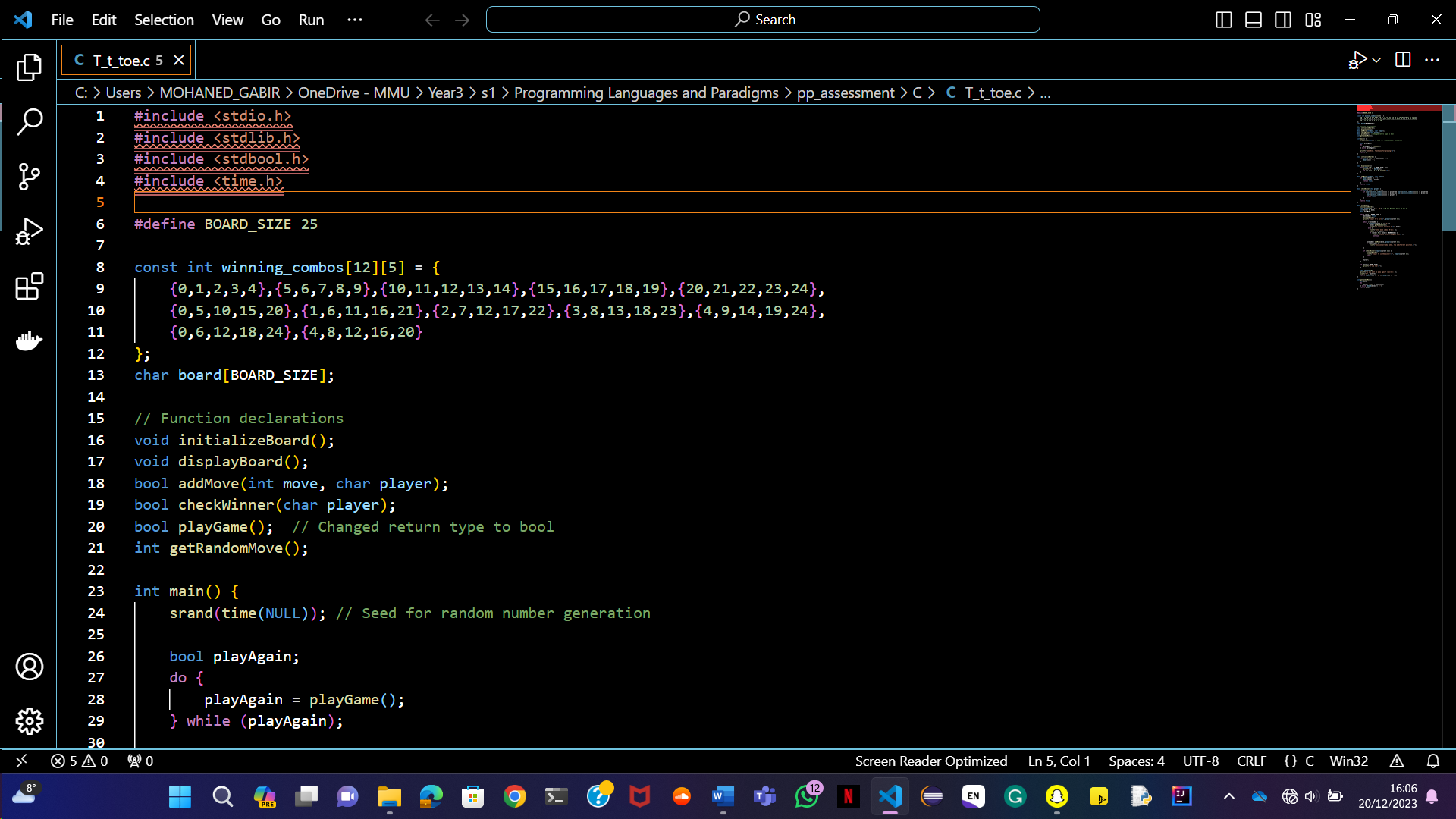
**Tic\_Tac\_Toe game in c**

This is my **C** version code for a player board game featuring a human player, **MG**, and a computer player, **AI**. The game uses a 5x5 grid, with each cell initially empty. Here is the design and the explanation of the code.

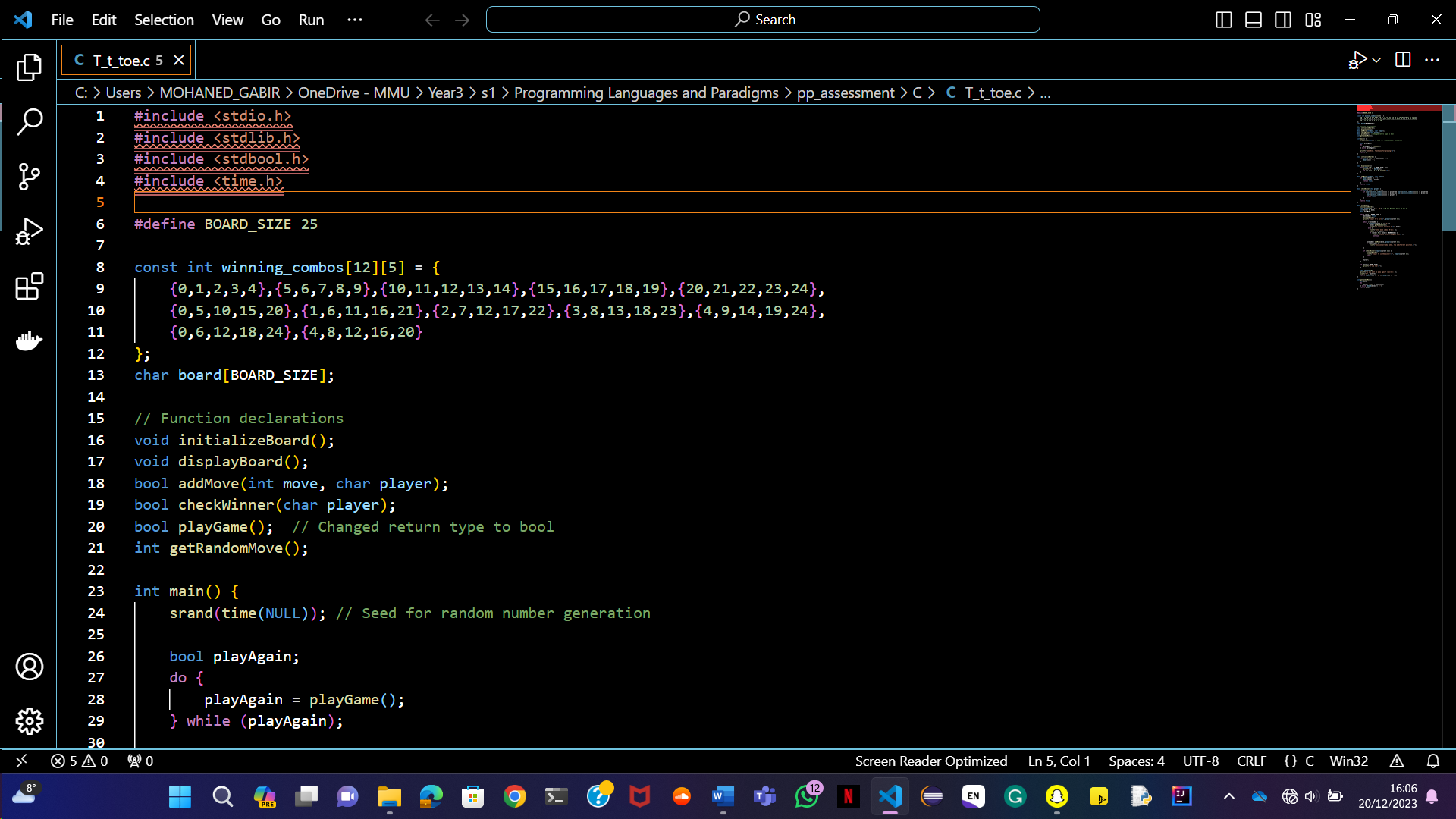
**Header Files**



Picture 1: headers files.

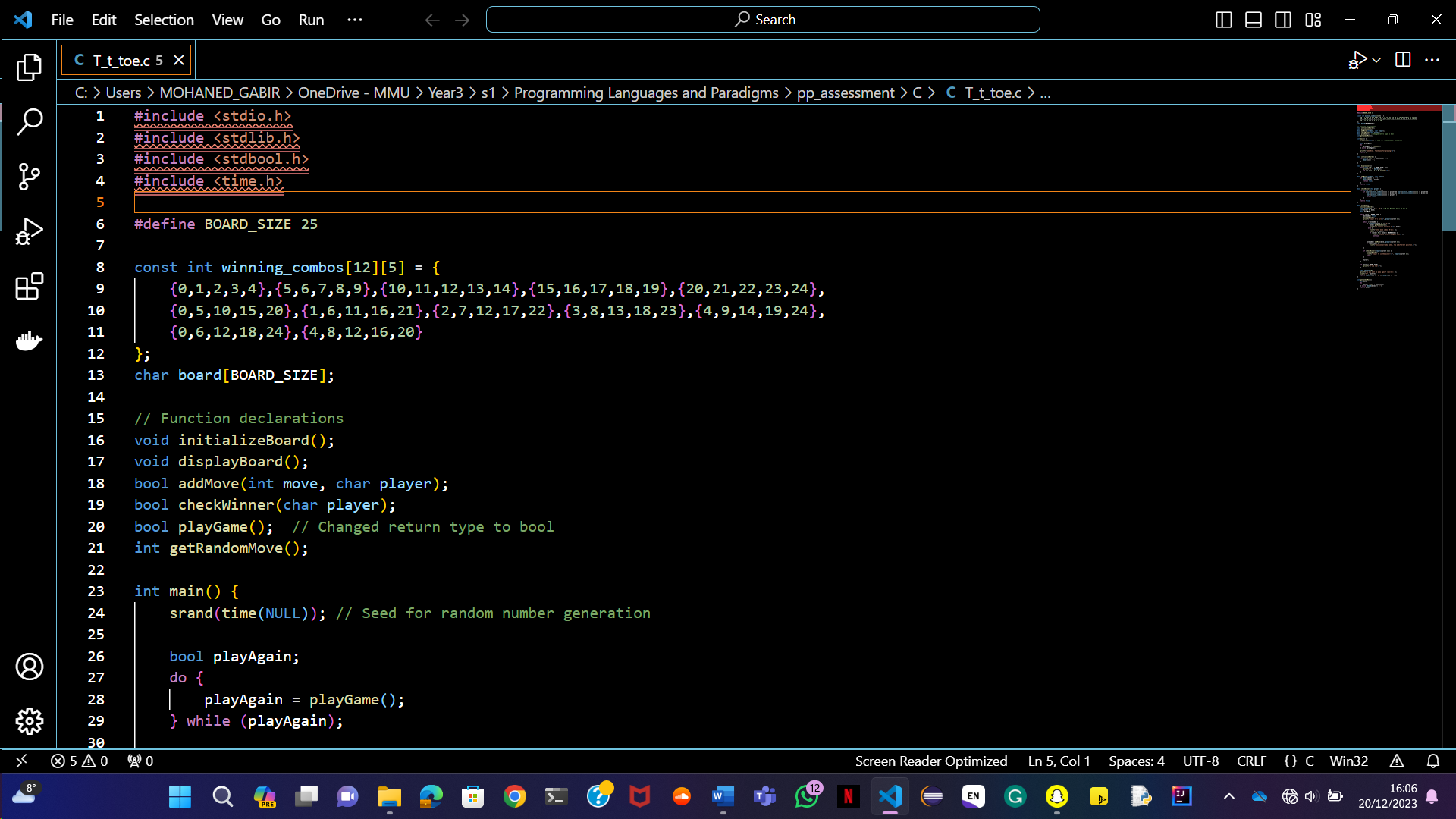
This line includes standard C libraries. **Stdio.h** for input and output functions, **stdlib.h** for general utility functions, **stdbool.h** for Boolean types, and **time.h** for time-related functions.

**Constants and Global Variables**



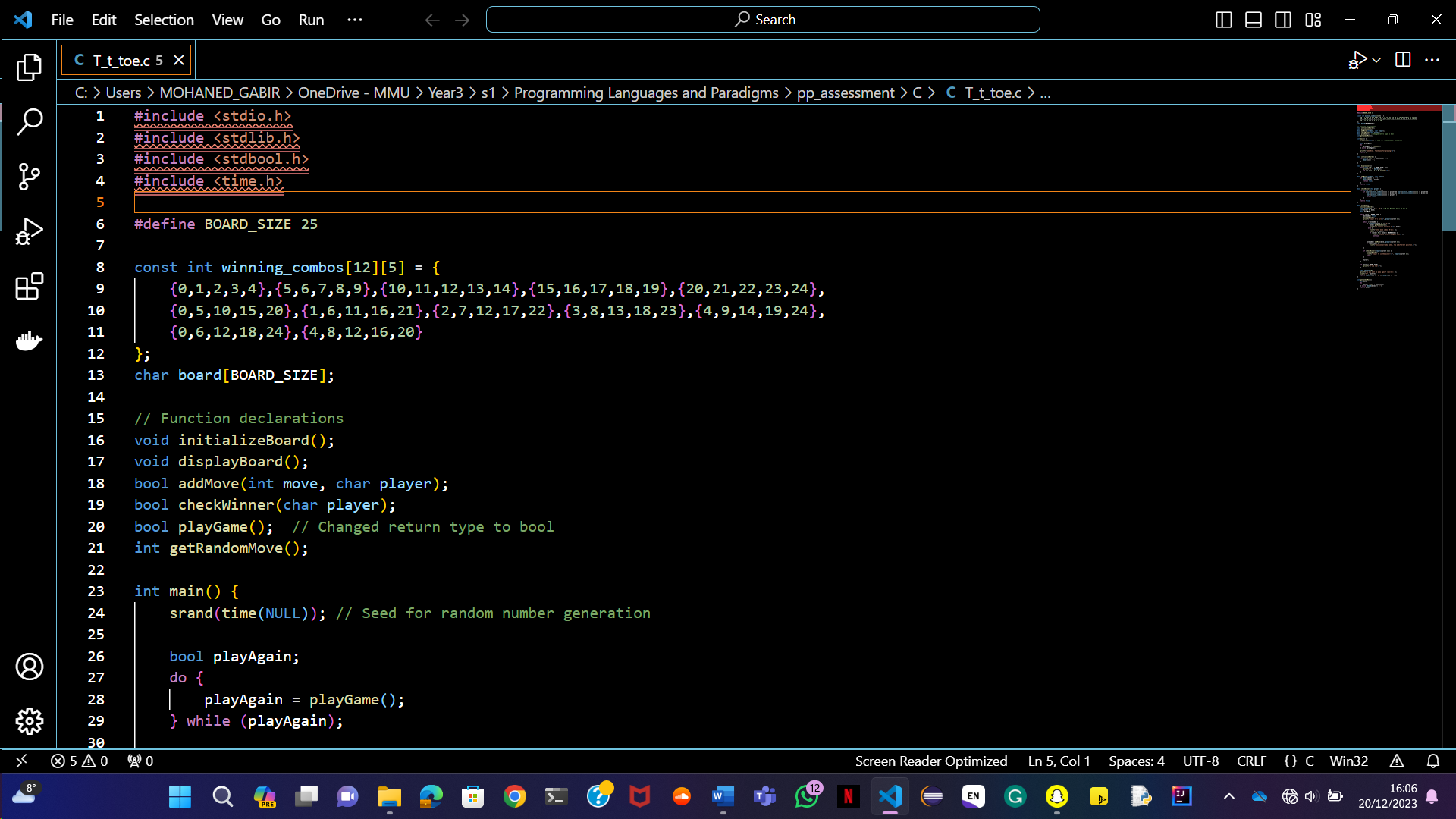
Picture 2: game size.

* Defines a constant BOARD\_SIZE for the size of the game board.



Picture 3: game-winning combination.

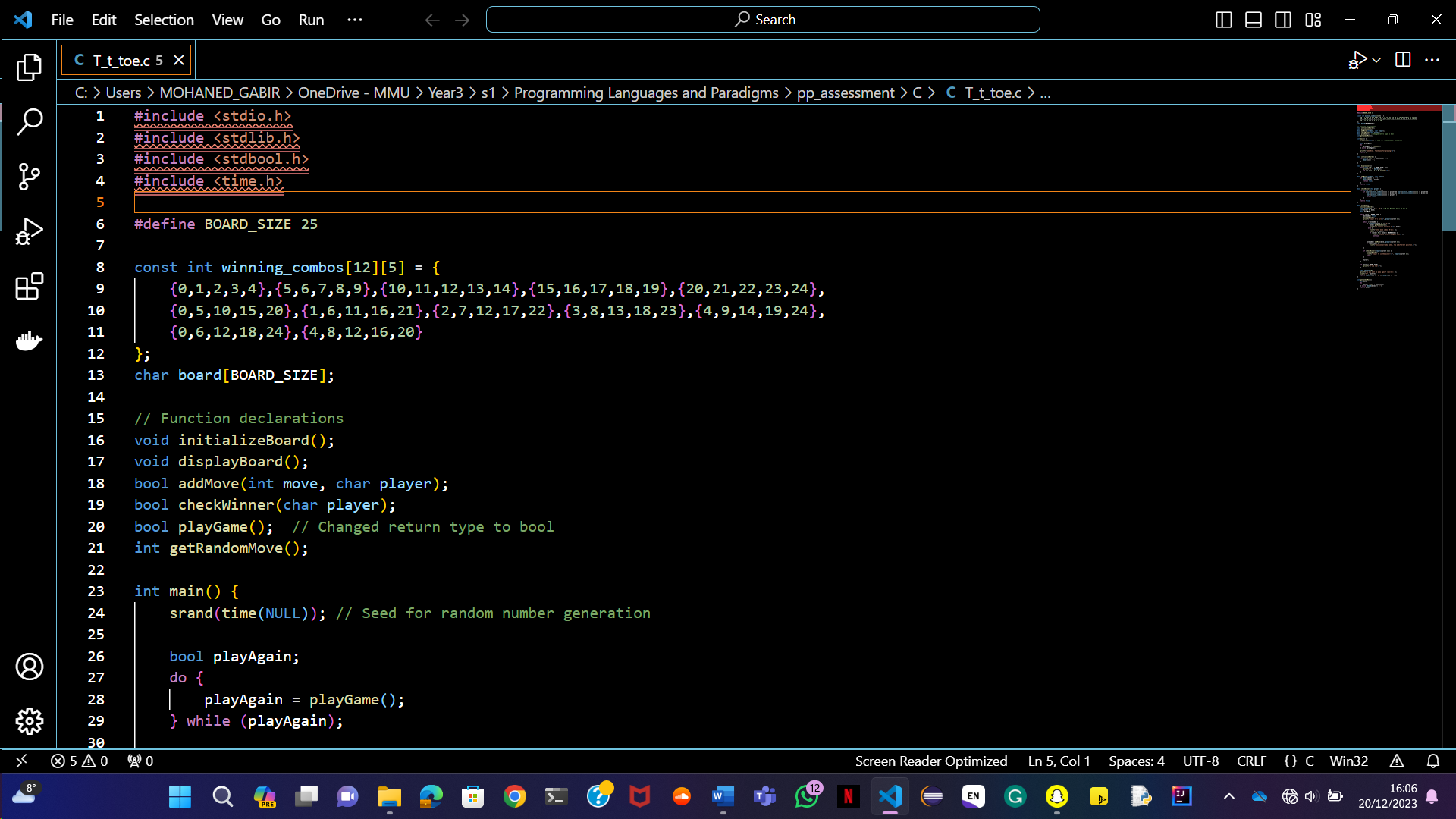
* Defines an array of winning combinations for the game.



Picture 4: game board array.

* Declares a global array board representing the game board.

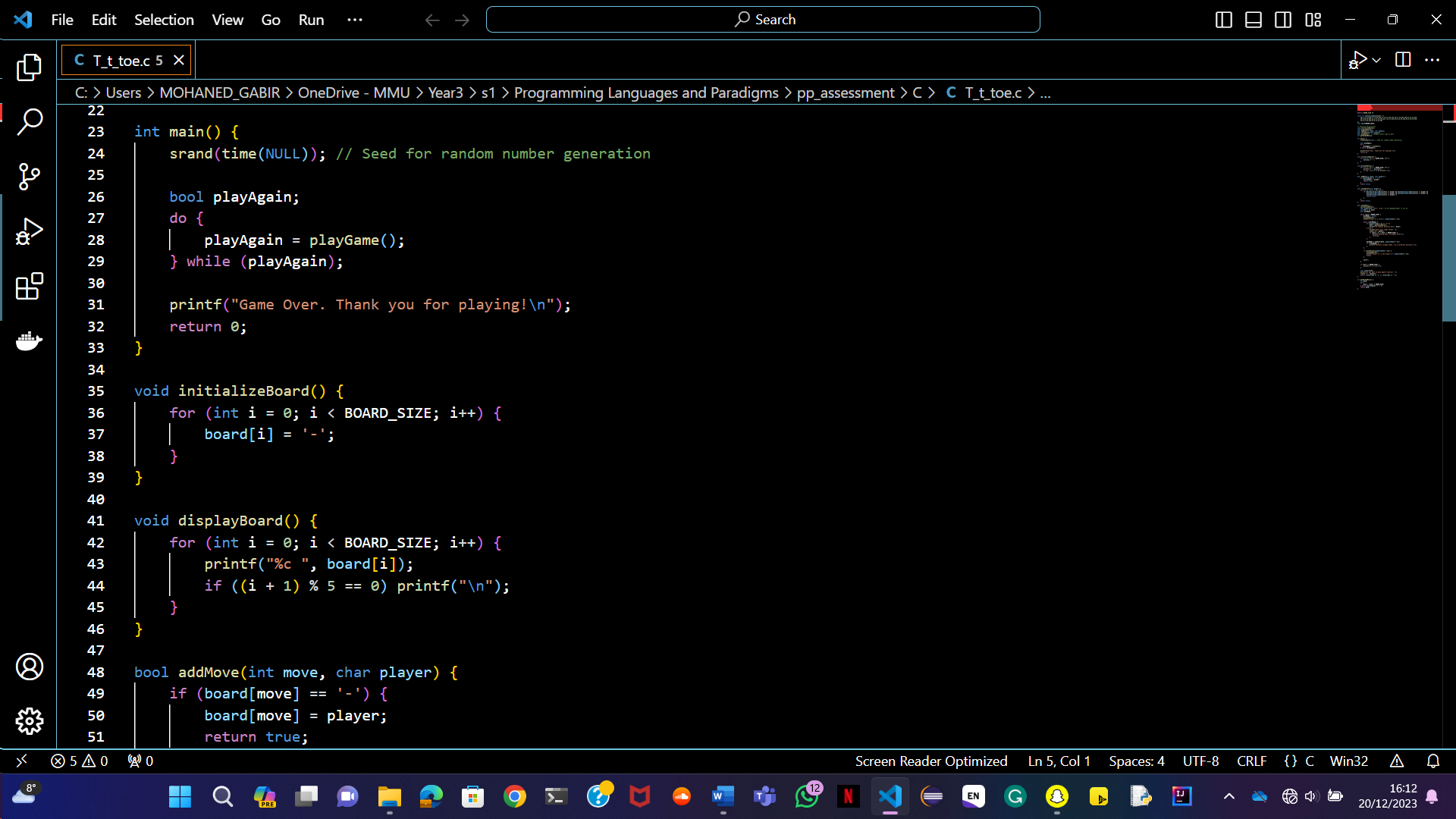
**Function Declarations**



Picture 5: functions declarations

* These lines declare the functions used in the program, specifying their return types and parameters.

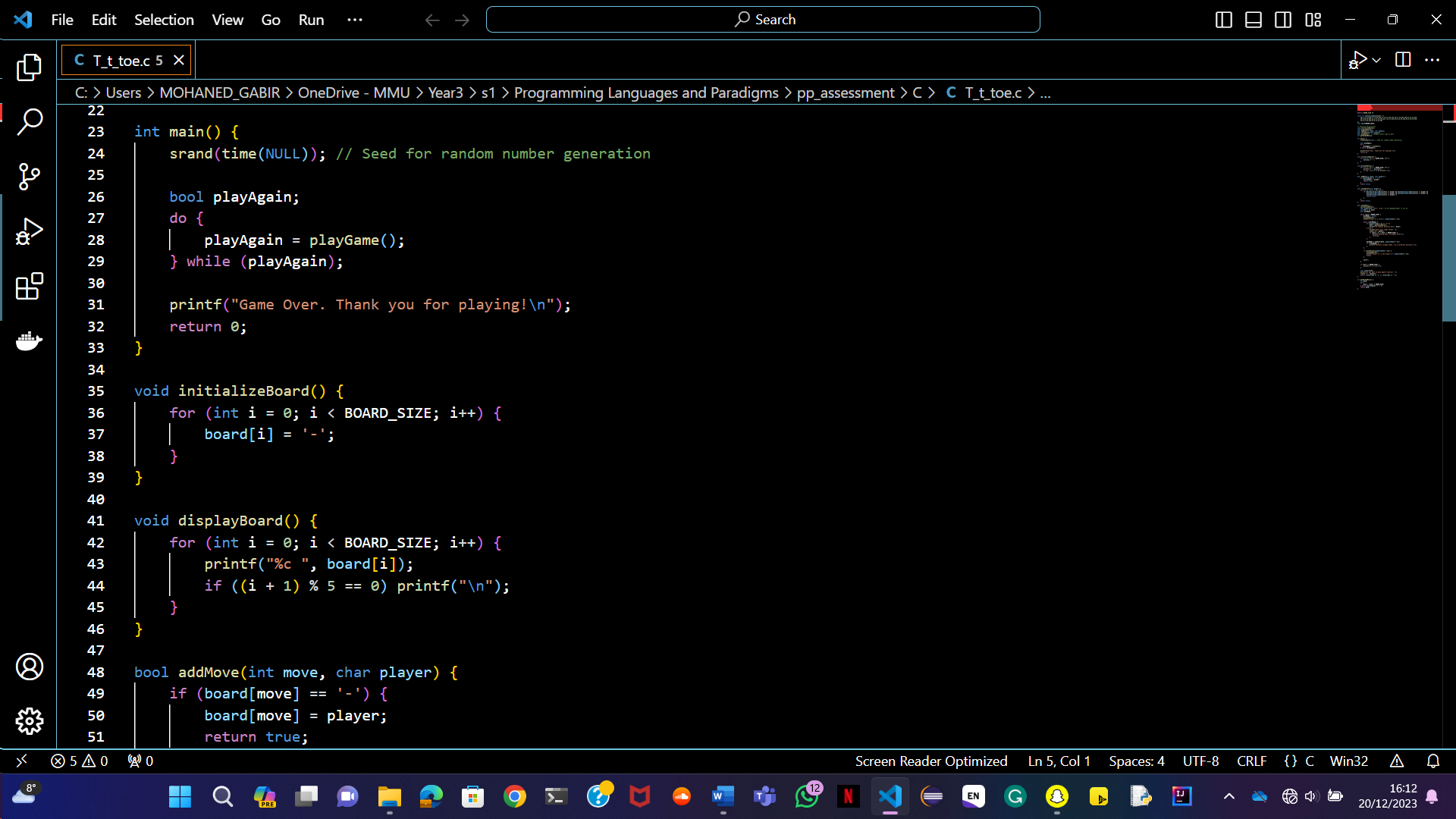
**Man Function**



Picture 6: primary function.

* Main is the entry point of the program.
* **Srand(time(NULL))** initialises the random number generator.
* A loop allows the game to be played multiple times based on the player’s choice.
* The program prints a game-over message before exiting.

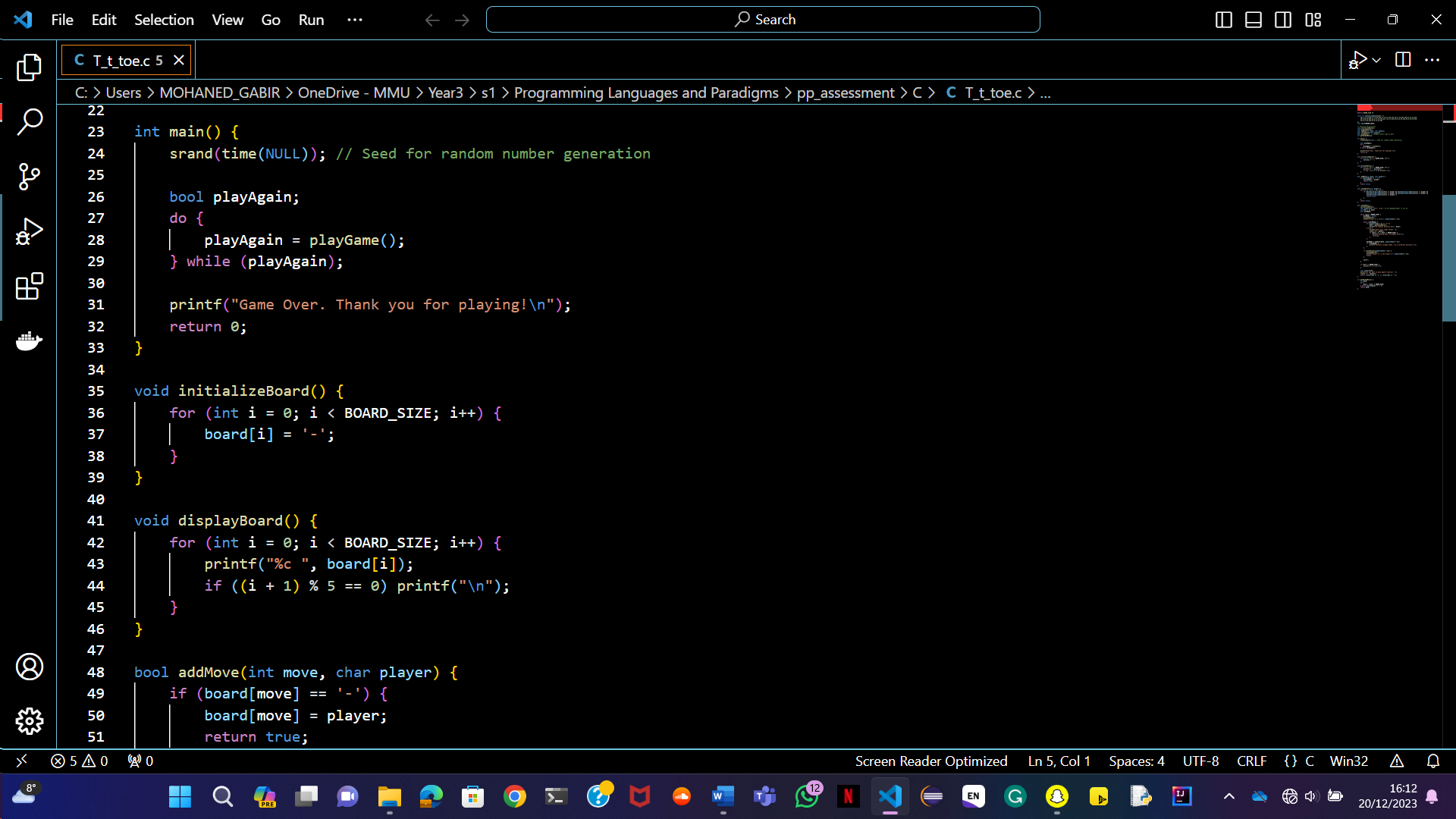
**InitializeBoard Function**



Picture 7: initialising board function.

* Initialises the game board by filling the board array with ‘-‘.

**DisplayBoard Function**



Picture 8: displaying board function.

* Prints the current state of the game board to the console.

**addMove Function**



Picture 9: adding a move to players if a spot is taken.

* Tries to add a player’s move to the board. If the spot is already taken, it returns false.

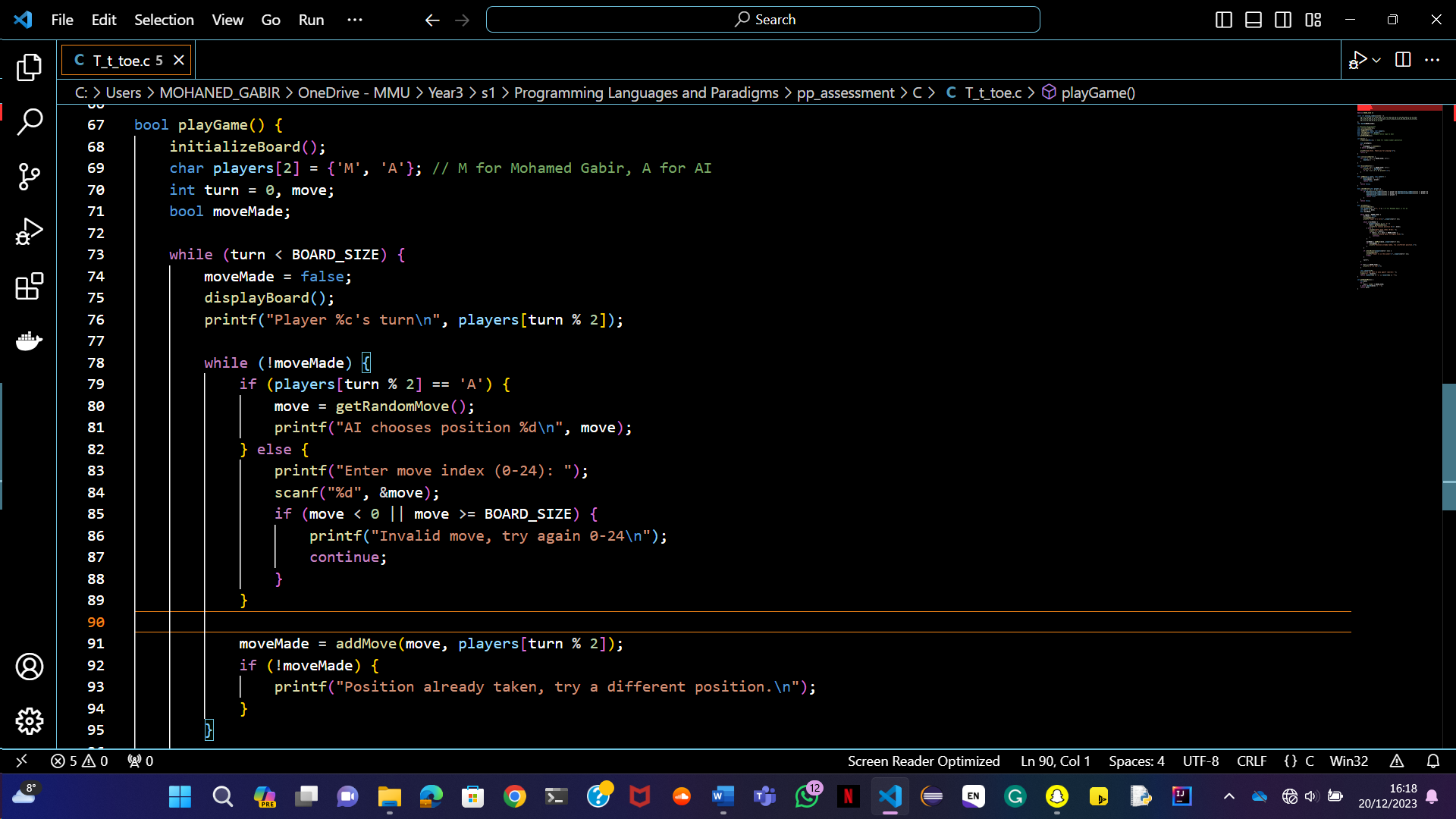
**CheckWinner Function**

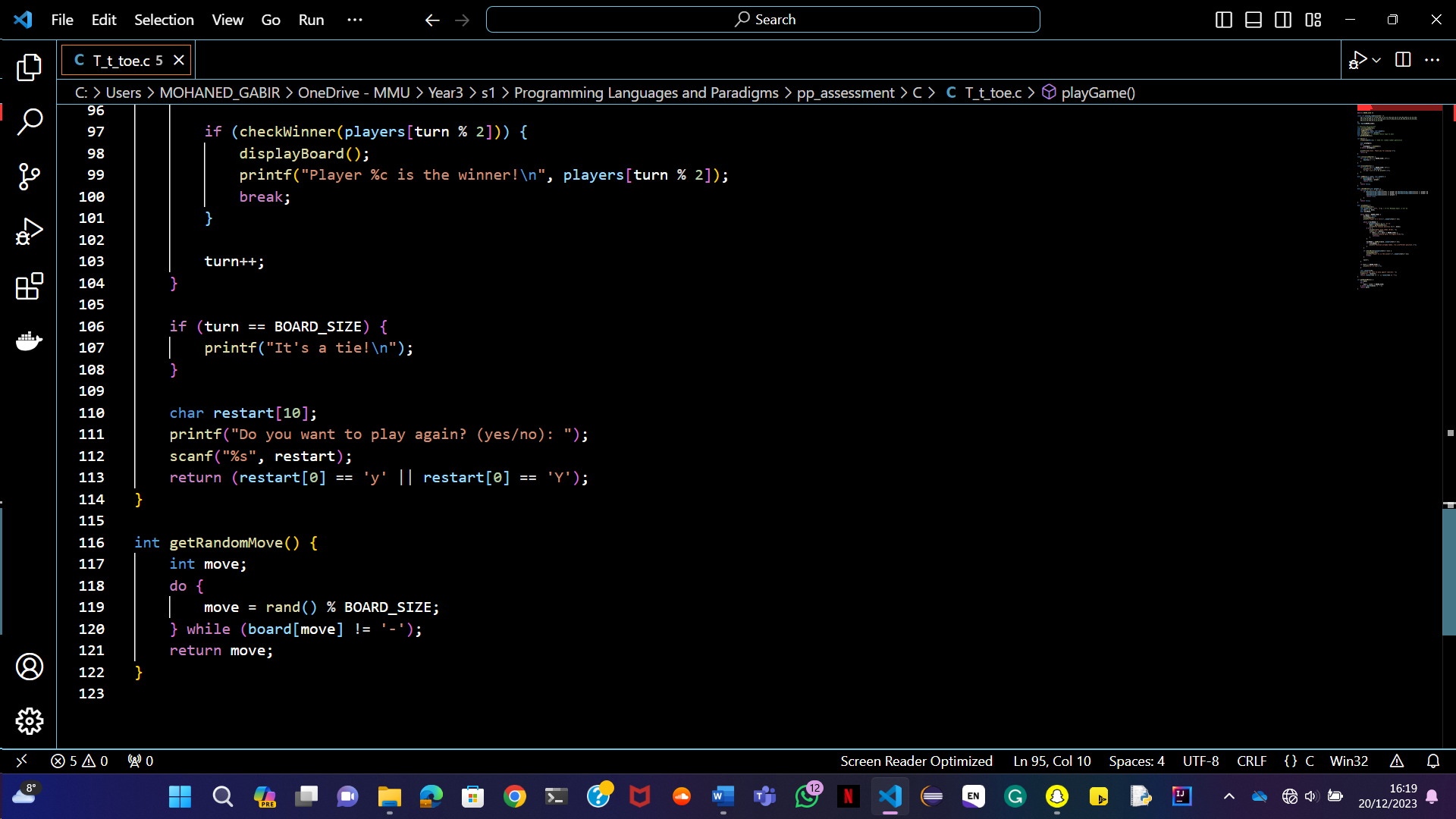


Picture 10: checking for player winner.

* The check winning function checks if the given player has won by matching any winning combination in the game.

**playGame function**

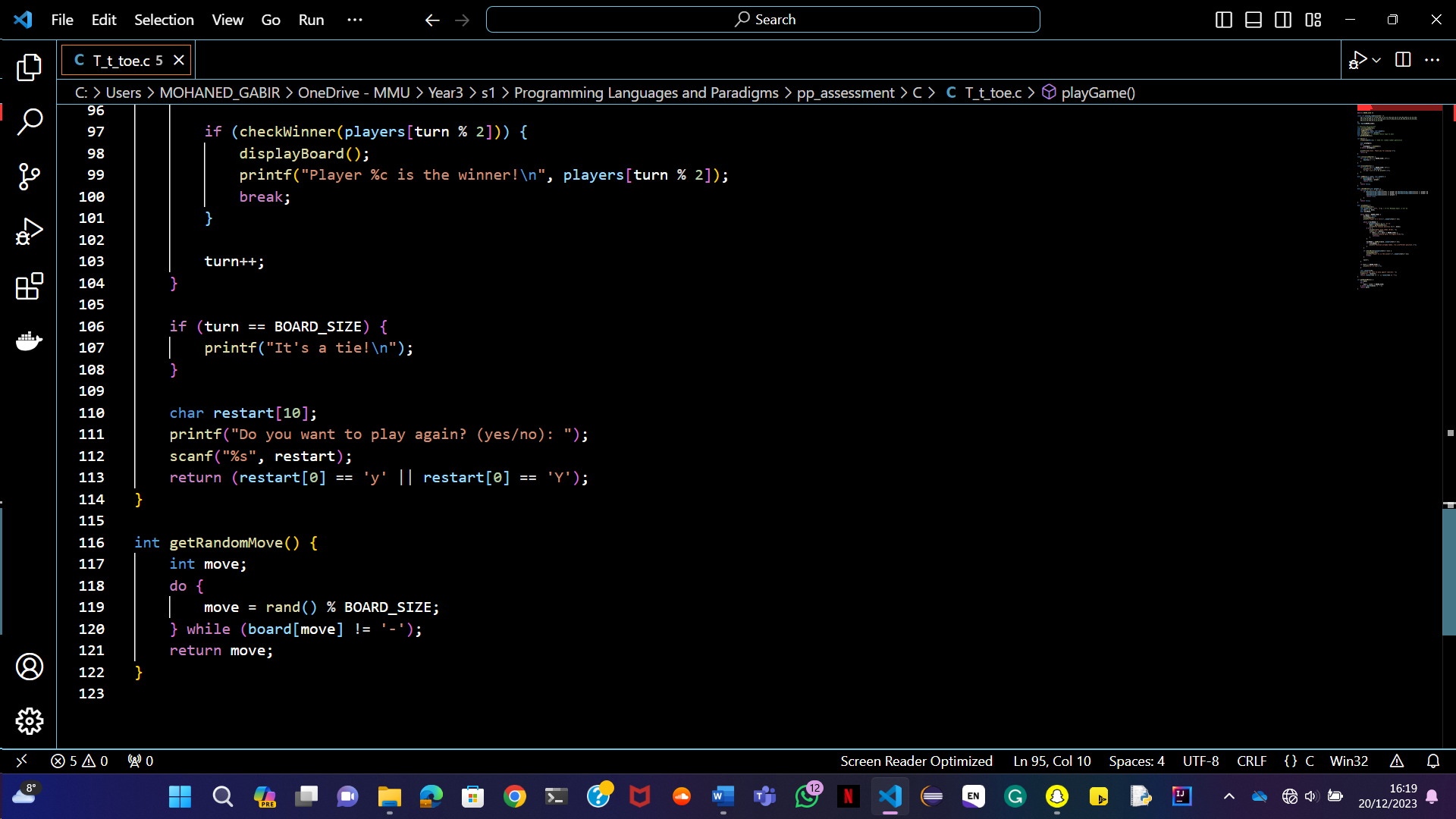




Picture 11: play game function.

* Orchestrates the game, initialises the board, and controls the game loop.
* Asks the player if they want to play again at the end.

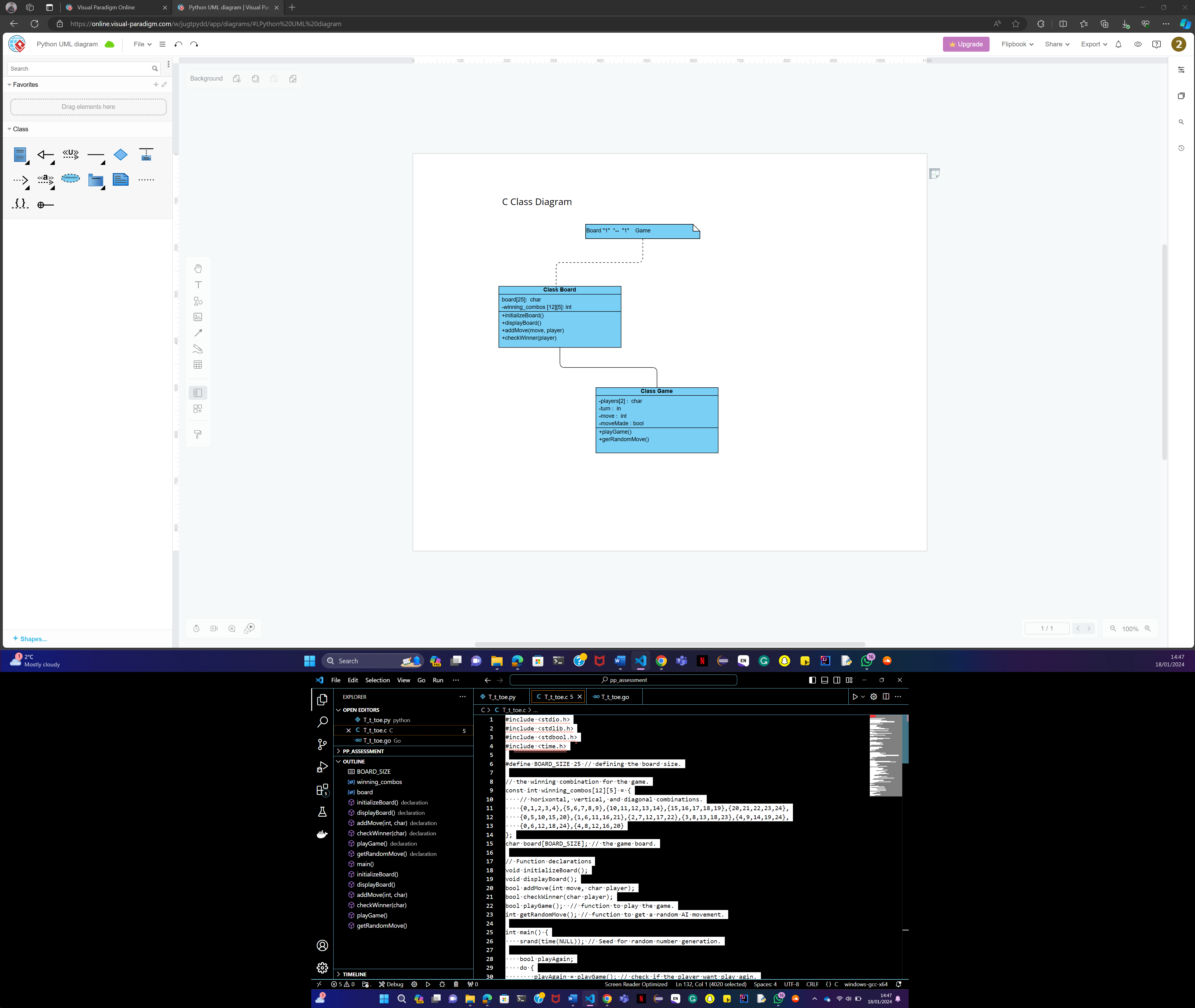
**getRandomMove function**



Picture 12: random move function.

* Generates a random move for the AI player, ensuring the spot is not already taken.

**C UML Diagram**



Picture 13: The drawing of class diagram.