**Describing the language features that you have used to solve the solution, and explaining how the solution conforms to the stated paradigm.**

Many language features include data types and type synonyms, list comprehensions, guards, module imports, and recursion used to solve this solution. The program uses the data types and type synonyms to create a player with two constructors "R" and "W". Utilize the list comprehensions to create a 3x4 girds game board. Also, the program has used module imports to "maybe" functional in the program to allow the game board set up maybe player to make the player only input the player type data (R and W) in the board. For the check win and draw part, the solution has applied to use the guards instead of using the conditional statement (If statement) to check if the player wins or draws. Also, guards were used to check the players if they made a valid move in the game. Instead of using loops to loop the main game for each round, this program has used the recursion function to check each round of the game to see if the game wins and draws. Additionally, the program also uses the higher-order function (map and intercalate) to design the game board and use the list to create winning combos.

The solution conforms to the functional paradigm, as evidenced by the following points: Firstly, the whole program was designed by the pure function. For example, the “checkWinner” function takes a board and a player as input and returns a Boolean output without relying on or modifying an external state. Secondly, the program embraces immutability as a fundamental principle. Notably, a function like “makeMove”, whenever running this function, the program creates a new board instead of modifying the existing board. Lastly, the program has used the recursion for the main game loop in the “game” function. The recursion is frequently utilized in the functional paradigm.