PROGRAMMING CHALLENGE 20: CONNECT 4

Connect 4 is a game played by two players. In the figure shown, one player uses red tokens and the other uses yellow. Each player has 21 tokens. The game board is a vertical grid of 6 rows and 7 columns.

Columns get filled with tokens from the bottom. The players take turns to choose a column that is not full and drop a token into this column. The token will occupy the lowest empty position in the chosen column. The winner is the player who is the first to connect 4 of their own tokens in a horizontal, vertical or diagonal line. If all tokens have been used and neither player has connected 4 tokens, the game ends in a draw.

Your task is to write a program to play this game on a computer by following these specifications:

- Represent the game board using a **2D array**;
- Designate players using 'O' and 'X';
- Player 'O' **always** start first;
- Players take turn in placing their tokens;
- Display game board after every turn;
- Check for a winner after a token is placed;
- Winner is the player who is the first to connect 4 of their tokens horizontally or vertically.
- The game can also be won by connecting 4 tokens ***diagonally***, but you are **NOT REQUIRED** to write code for winning with diagonally connected tokens.

Use this top-level pseudocode with the given modules:

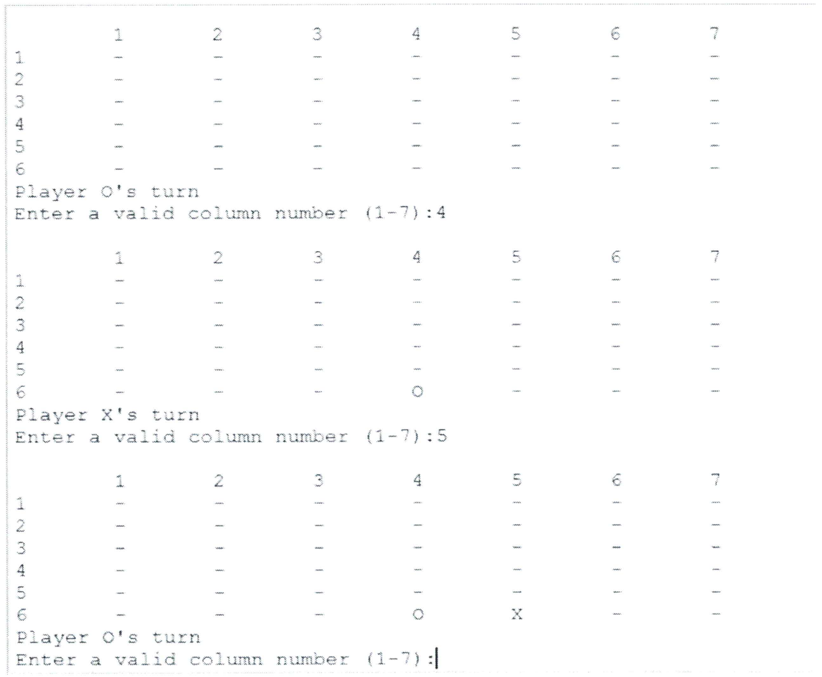
```
CALL InitialiseBoard
CALL SetUpGame
CALL OutputBoard
WHILE GameFinished = FALSE
    CALL ThisPlayerMakesMove
    CALL OutputBoard
    CALL CheckIfThisPlayerHasWon
    IF GameFinished = FALSE THEN
        CALL SwapThisPlayer
    ENDIF
ENDWHILE
```

The identifiers used in the pseudocode and explanations are given as follow:

Identifier	**Explanation**
`Board[1..6, 1..7]`	• 2D array to represent the board
`InitialiseBoard`	• Procedure to initialise the board to all blanks. • Use a suitable character to represent blank.
`SetUpGame`	• Procedure to set initial values for `GameFinished` and `ThisPlayer`
`GameFinished`	• `FALSE` if the game is not finished • `TRUE` if a player has won or board is full
`ThisPlayer`	• '`O`' when it is Player `O`'s turn • '`X`' when it is Player `X`'s turn
`OutputBoard`	• Procedure to output the current contents of the board
`ThisPlayerMakesMove`	• Procedure to get current player to input column number and place the token into the chosen board location. • Validation must be done on user input of column number.
`CheckIfThisPlayerHasWon`	• Procedure to check if the token just placed makes the current player a winner. • Checks should be made on whether the token just placed connected 4 tokens to form a horizontal or vertical line, and whether the game ends in a draw. • You **DO NOT** need to do diagonal check.
`SwapThisPlayer`	• Procedure to change player's turn

You **must use the above identifiers** and **other additional identifiers** of your own.

Row numbers and **column numbers** are displayed with the board's contents. Here is a ***sample screenshot*** of the first turns taken by player O and player X:



```
        1       2       3       4       5       6       7
1       -       -       -       -       -       -       -
2       -       -       -       -       -       -       -
3       -       -       -       -       -       -       -
4       -       -       -       -       -       -       -
5       -       -       -       -       -       -       -
6       -       -       -       -       -       -       -
Player O's turn
Enter a valid column number (1-7):4

        1       2       3       4       5       6       7
1       -       -       -       -       -       -       -
2       -       -       -       -       -       -       -
3       -       -       -       -       -       -       -
4       -       -       -       -       -       -       -
5       -       -       -       -       -       -       -
6       -       -       -       O       -       -       -
Player X's turn
Enter a valid column number (1-7):5

        1       2       3       4       5       6       7
1       -       -       -       -       -       -       -
2       -       -       -       -       -       -       -
3       -       -       -       -       -       -       -
4       -       -       -       -       -       -       -
5       -       -       -       -       -       -       -
6       -       -       -       O       X       -       -
Player O's turn
Enter a valid column number (1-7):
```

Task 1

Write program code for `InitialiseBoard`, `SetUpGame`, `OutputBoard`, and call these procedures. You may introduce **other additional identifiers** of your own, including parameter(s) and return value(s).

Evidence 1: Program code for `InitialiseBoard`, `SetUpGame`, `OutputBoard` and calling these procedures. Include a screenshot of running these procedures. **[8]**

Task 2

Write program code for `ThisPlayerMakesMove`. You may introduce **other additional identifiers** of your own, including parameter(s) and return value(s).

Evidence 2: Program code for `ThisPlayerMakesMove`. **[7]**

Task 3

Write program code for `CheckIfThisPlayerHasWon`. You may introduce **other additional identifiers** of your own, including parameter(s) and return value(s).

Evidence 3: Program code for `CheckIfThisPlayerHasWon`. **[15]**

Task 4

Write program code for `SwapThisPlayer`. You may introduce **other additional identifiers** of your own, including parameter(s) and return value(s).

Evidence 4: Program code for `SwapThisPlayer`. **[3]**

Task 5

Write program code for the top-level pseudocode (on page 1) that makes use of all the procedures from **Task 1 to 4**.

Evidence 5: Program code for the top-level pseudocode. **[4]**

Evidence 6: Run your program and produce screenshots for a game which ends in a draw and another game which player X wins. **[2]**

[Total: 39 marks]