Hands-on Activity 5.1:	
Multidimensional Arrays	
Course Code: CPE007	**Program:** Computer Engineering
Course Title: Programming Logic and Design	**Date Performed:** 9/30/2025
Section: CPE11S1	**Date Submitted:** 9/30/2025
Name(s): Ramirez, Angel Mae C.	**Instructor:** Engr. Jimlord M. Quejado

6. Output

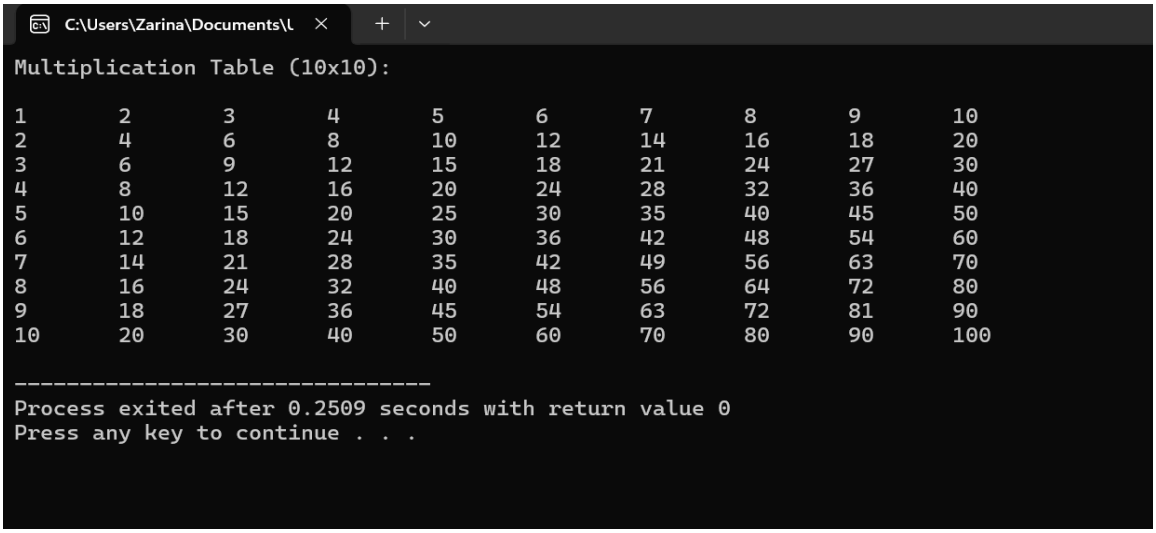
1. Write a program that creates a multiplication table using multidimensional array.

CODE:

Untitled1.cpp

```cpp
#include <iostream>
using namespace std;

int main() {
    const int rows = 10;
    const int cols = 10;
    int table[rows][cols];

    // Fill the multiplication table
    for (int i = 0; i < rows; ++i) {
        for (int j = 0; j < cols; ++j) {
            table[i][j] = (i + 1) * (j + 1);
        }
    }

    // Display the multiplication table
    cout << "Multiplication Table (10x10):\n\n";
    for (int i = 0; i < rows; ++i) {
        for (int j = 0; j < cols; ++j) {
            cout << table[i][j] << "\t";
        }
        cout << endl;
    }

    return 0;
}
```

OUTPUT:



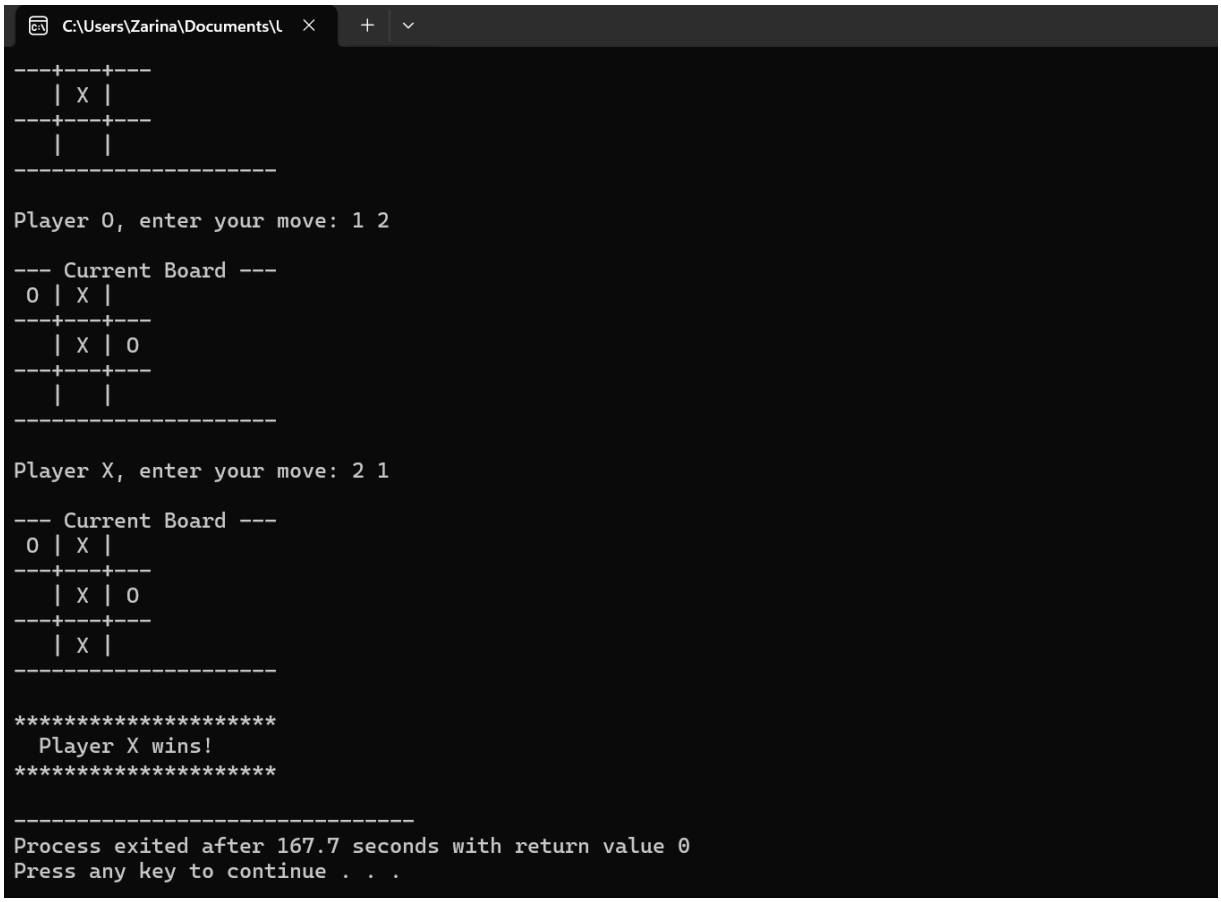
2. Write a program that creates a board with a tic-tac-toe moves.

CODE:

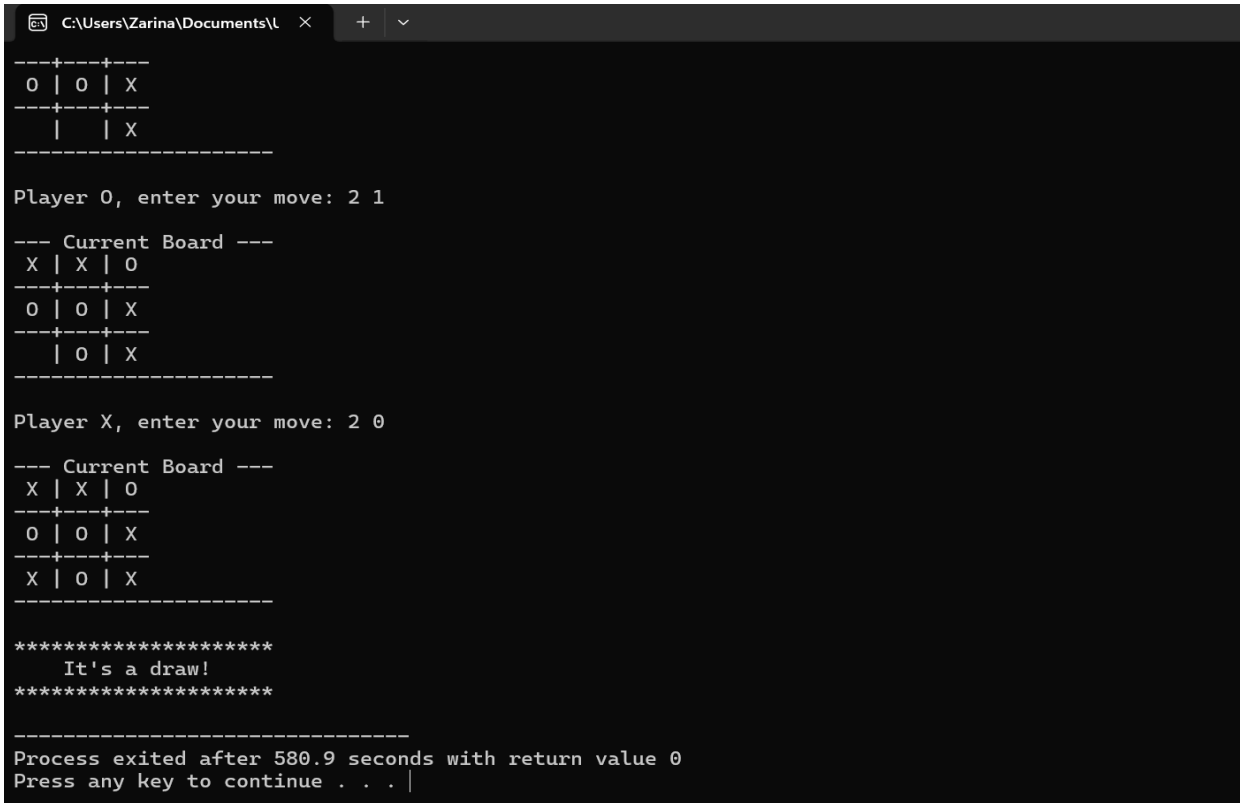
```
            return true;
        }
    }


    if (gameBoard[0][0] == symbol && gameBoard[1][1] == symbol && gameBoard[2][2] == symbol) {
        return true;
    }
    if (gameBoard[0][2] == symbol && gameBoard[1][1] == symbol && gameBoard[2][0] == symbol) {
        return true;
    }

    return false;
}

int main() {
    char board[3][3] = {
        {' ', ' ', ' '},
        {' ', ' ', ' '},
        {' ', ' ', ' '}
    };

    char player = 'X';
    int moveCount = 0;
    int row, column;

    cout << "--- Welcome to Tic-Tac-Toe! ---" << endl;
    cout << "Enter your move as 'row column' (e.g., '0 1')." << endl;

    while (moveCount < 9) {
        displayBoard(board);
        cout << "Player " << player << ", enter your move: ";
        cin >> row >> column;


        if (row < 0 || row > 2 || column < 0 || column > 2 || board[row][column] != ' ') {
            cout << "Invalid move! Try again." << endl;
            continue;
        }


        board[row][column] = player;
        moveCount++;


        if (hasWon(board, player)) {
            displayBoard(board);
            cout << "**********************" << endl;
            cout << "  Player " << player << " wins!  " << endl;
            cout << "**********************" << endl;
            return 0;
        }


        player = (player == 'X') ? 'O' : 'X';
    }


    displayBoard(board);
    cout << "**********************" << endl;
    cout << "    It's a draw!     " << endl;
    cout << "**********************" << endl;

    return 0;
}
```

OUTPUT:

PLAYER WINNING:



PLAYERS IN DRAW:



7. Supplementary Activity

1. ANALYSIS:

In this first activity I wanted a multiplication table that goes from 1 to 10. So I declared two constants rows and cols, both set to 10. Then, I created a 2D array called table rows cols to store the multiplication values. This array acts like a grid with 10 rows and 10 columns, where each cell will contain the product of (row index + 1) * (column index + 1). after it I used nested for-loops to fill in the values of the table. The outer loop goes through each row, and the inner loop goes through each column. Inside the inner loop I calculated the product of (i + 1) * (j + 1) because array indices start at 0, but multiplication tables start at 1. I stored each result in the corresponding position in the array. After filling in the values, I used another set of nested for loops to display the table. I remembered that sir thought me to use (\t) so that the numbers would align nicely in columns so i used it. Finally, I added return 0; to indicate that the program ended successfully.

2. ANALYSIS:

In this second activity the first thing i did is to create a function called displayBoard() that takes a 3x3 character array and prints it in the form of a tic tac toe board. I used a nested loop and formatting with | and ---+--- to make it look like a real game board. It takes the board and the current player's symbol ('X' or 'O') as parameters and returns true if the player has won, or false otherwise. I used loops and if-statements to check all 3 rows, 3 columns, and 2 diagonals. In the main() function, I started by creating the 3x3 board using a char array filled with spaces, so that it looks empty when the game starts. I also made some variables like player to know whose turn it is starting with X, moveCount to count how many turns have been made, and row and column to store the input from the player. The main part of the game is inside a while loop that keeps running until all 9 moves are used. Inside the loop, the board is displayed using the displayBoard() function so the players can see where they can put their move. Then it asks the current player to enter the position they want to play by typing a row and column. The program checks if the move is allowed, like if it's inside the board and not already taken. If it's not valid, it just tells the player to try again and skips the rest of the loop.

If the move is ok, then the player's symbol is placed on the board and the move count goes up. Then I used the hasWon() function to see if the player won by checking all rows, columns, and the diagonals. If there's a winner, it will show the board and say who wins, and then the game ends. If no one wins, the code just switches the turn from X to O or the other way. The loop keeps going until the board is full. If no one won after 9 moves, then it's a draw and it shows the final board and says it's a draw. This part of the code makes sure the game works whether someone wins or not that is all.

8. Conclusion

This activity has really taught me how to work with 2D arrays in a real program. I've seen few examples, but actually writing code to create a multiplication table and a simple game made it easier to understand how in practice multidimensional arrays work. I struggled a little thinking of how to proprly set up the rows and columns, particularly while using nested loops. However, after running the code a couple of times and fixing some minor things, I got better at how everything connected how the outer loop controls the rows and how the inner one controls the columns. Seeing the output of the multiplication table change rightly made me know I was doing it right, which feels quite good.

Working on that part of the tic-tac-toe game taught me more than just how arrays work. It helped me understand how a program should run step by step like how players take turns, how to stop them from putting their symbol in a spot that's already taken, and how to tell when someone wins or the game ends. I had to think about how each move affects the rest of the game, which made me pay more attention to the small details. I also had to do a lot of testing and fixing to catch little mistakes, like forgetting to switch players or not checking the win conditions the right way.