Hands-on Activity 5.1	
Multidimensional Arrays	
Course Code: CPE007	**Program:** Computer Engineering
Course Title: Programming Logic and Design	**Date Performed:** September 25, 2025
Section: CPE11S1	**Date Submitted:** September 29, 2025
Name(s): Mendoza, Nathaniel B.	**Instructor: Engr. Jimlord M. Quejado**

6. Output

1. **Code:**

MultiplicationTableMENDOZA.cpp | TicTacToeMENDOZA.cpp

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

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

    // Filling the table
    for (int i = 0; i < size; i++) {
        for (int j = 0; j < size; j++) {
            table[i][j] = (i + 1) * (j + 1);
        }
    }

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

    return 0;
}
```

Code Output:

```
C:\Users\NathanieL\Documents\MultiplicationTableMENDOZA.exe
Multiplication Table (10x10):
1       2       3       4       5       6       7       8       9       10
2       4       6       8       10      12      14      16      18      20
3       6       9       12      15      18      21      24      27      30
4       8       12      16      20      24      28      32      36      40
5       10      15      20      25      30      35      40      45      50
6       12      18      24      30      36      42      48      54      60
7       14      21      28      35      42      49      56      63      70
8       16      24      32      40      48      56      64      72      80
9       18      27      36      45      54      63      72      81      90
10      20      30      40      50      60      70      80      90      100

--------------------------------
Process exited after 0.1323 seconds with return value 0
Press any key to continue . . .
```

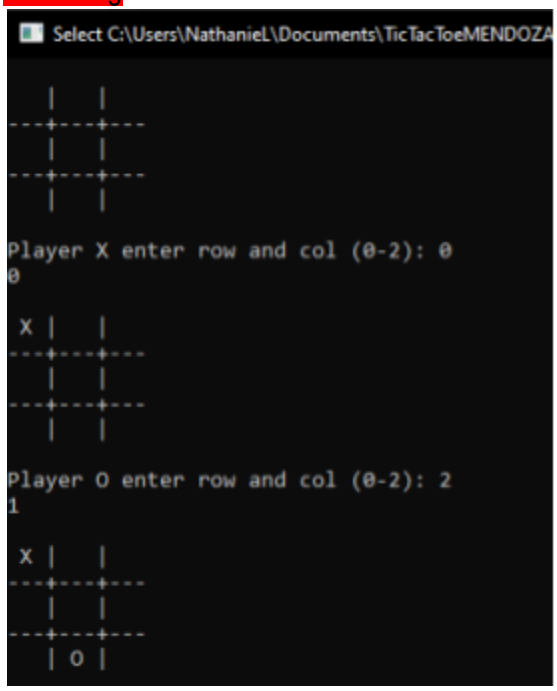
2. Code:

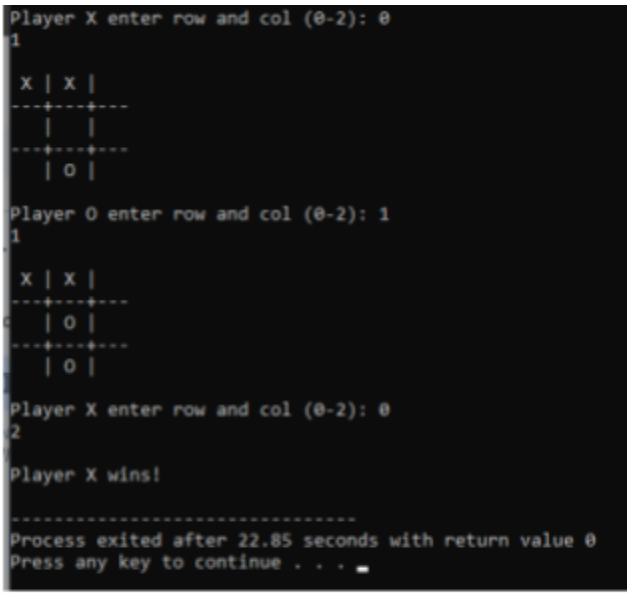
MultiplicationTableMENDOZA.cpp | TicTacToeMENDOZA.cpp

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

int main() {
    char board[3][3] = {{' ', ' ', ' '},
                        {' ', ' ', ' '},
                        {' ', ' ', ' '}};
    char player = 'X';
    int row, col, moves = 0;

    while (true) {
        // Print board
        cout << endl;
        for (int i = 0; i < 3; i++) {
            cout << " ";
            for (int j = 0; j < 3; j++) {
                cout << board[i][j];
                if (j < 2) cout << " | ";
            }
            cout << endl;
            if (i < 2) cout << "---+---+---" <<endl;
        }

        // Player move
        cout << "\nPlayer " << player << " enter row and col (0-2): "; //Can only in put from 0 to 2 only then anothe
        cin >> row >> col;

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

        board[row][col] = player;
        moves++;

        // Check win (Applying Logical Operators)
        if ((board[0][0]==player && board[1][1]==player && board[2][2]==player) ||
            (board[0][2]==player && board[1][1]==player && board[2][0]==player) ||
            (board[row][0]==player && board[row][1]==player && board[row][2]==player) ||
            (board[0][col]==player && board[1][col]==player && board[2][col]==player)) {
            cout << "\nPlayer " << player << " wins!\n";
            break;
        }

        if (moves == 9) {
            cout << "\nIt's a draw!\n";
            break;
        }

        // Switch player
        player = (player == 'X') ? 'O' : 'X';
    }
}
```

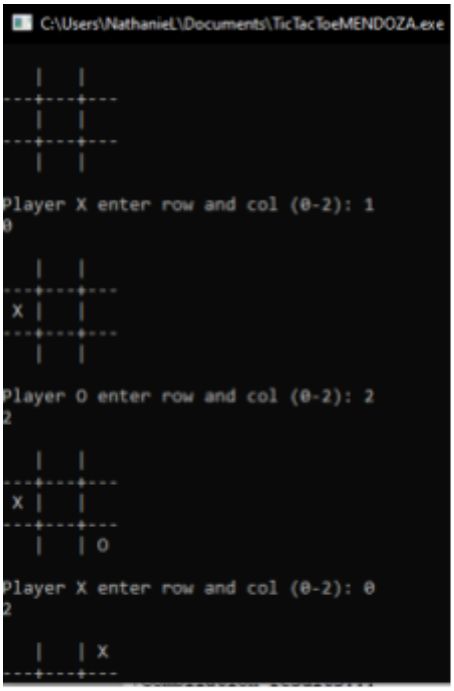
Code Output:

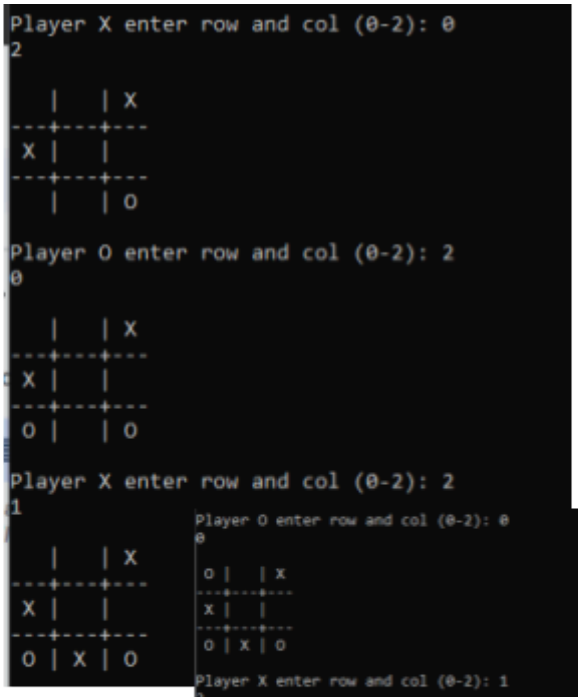
X Winning

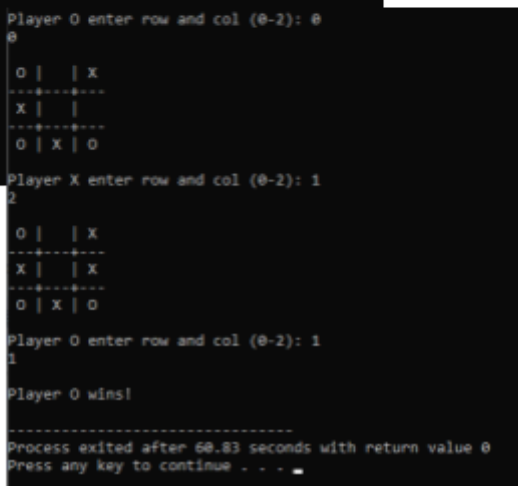




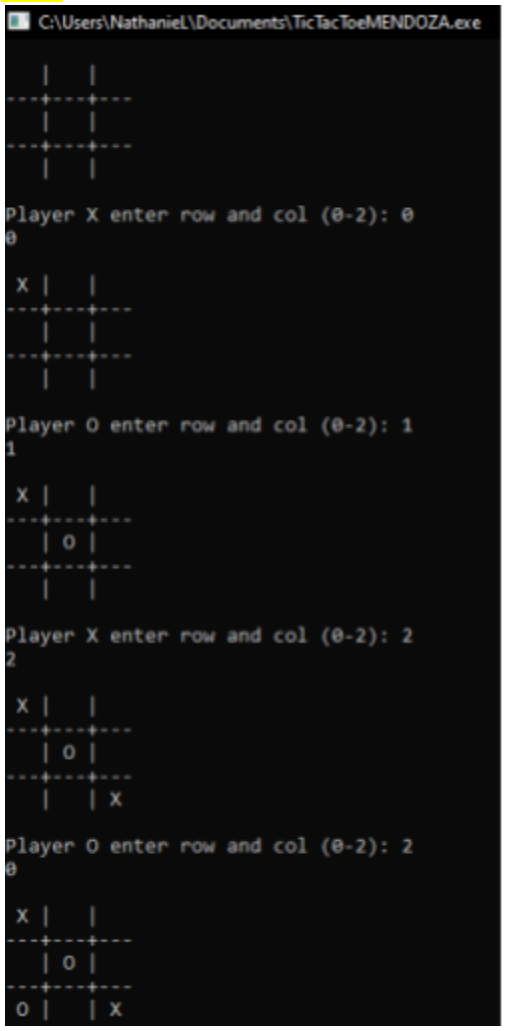
O Winning

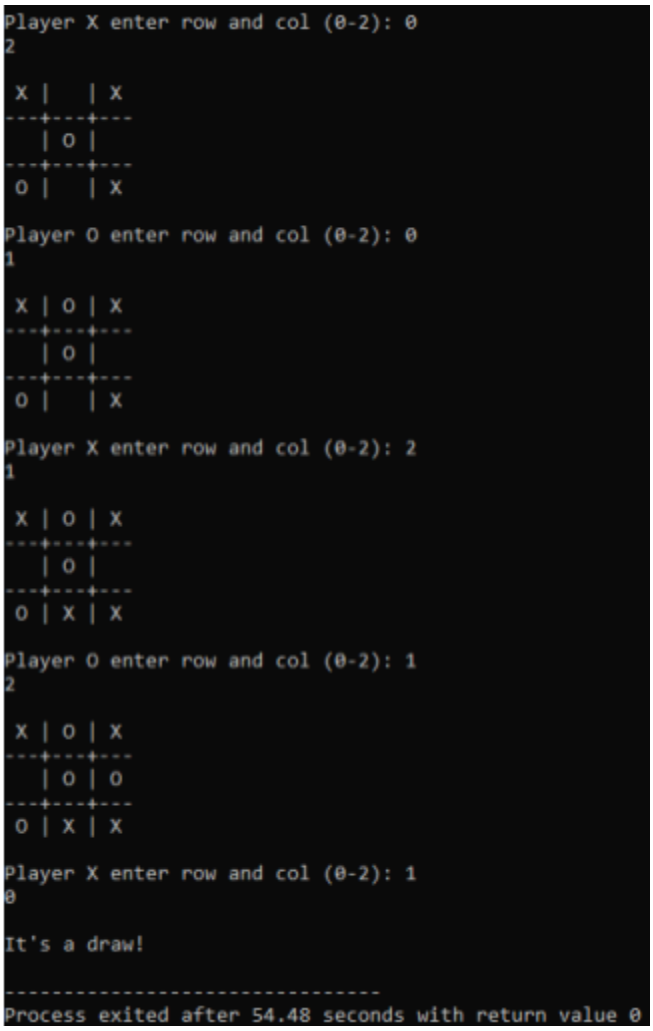






Draw





7. Supplementary Activity

1. Code Analysis:

I started the program by implementing #include <iostream> known as Input and Output Stream so that I would be able to apply cout for my program. Next is that I used using namespace std; therefore I wouldn't use std:: for every time I use cout. Then I implemented a multidimensional array in which I used int as my element data type then table as my arrayName as it was being asked for the problem then my arraysize consists of [10][10] so that it would consist a total of 100 outputs for my multiplication table from 1 to 10. After that I just implemented a multiplication table of 1 to 10 each row. Next, I utilized a nested for loop in which both stops after it being less than 10 and while it is less than 10 it would keep incrementing it or +1 until it reaches 9. Next, I printed the multiplication table and added a tab so that it would keep the spaces clean and then endl;. Finally, I ended the program with return 0.

2. Code Analysis:

I started the program by implementing #include <iostream> known as Input and Output Stream so that I would be able to apply cout and cin for my program. Next is that I used using namespace std; therefore I wouldn't use std:: for every time I use cout or cin. Then I implemented a multidimensional array in which I used char as my element data type then board as my arrayName which is a 3 by 3 array since the game Tic-Tac-Toe only needs 9 spaces. I also declared variables such as player initialized with 'X', row and col for storing player's input coordinates, and

moves initialized as 0 to count how many turns have been played. After that I implemented a while(true) loop so the game keeps running until a winner or draw is detected. Inside the loop I first printed the board using nested for loops which display the current state of the 3x3 grid along with dividers (| and ---+---+---) so it looks like a real Tic-Tac-Toe board. Next, I asked the current player to input their row and column from 0–2. I also added a condition to check if the input is valid (not out of bounds and not already taken). If it is invalid, it displays “Invalid! Try again.” and continues the loop. If the move is valid, the chosen position is filled with the current player’s mark (X or O) and the moves counter is incremented. Then I applied logical operators to check the winning conditions diagonals, the current row, and the current column. If the player meets a winning condition, it prints out that the current player has won and then breaks the loop. If no one wins but the total moves reach 9, then it displays “It’s a draw!” and breaks the loop as well. Finally, if the game is not over, the program switches the current player using a ternary operator: if the player was X, it changes to O, and if it was O, it changes to X. This continues until a win or a draw ends the game.

8. Conclusion

Throughout this activity, while making these two programs, I realized how useful multidimensional arrays are not just in solving math problems but also in creating interactive applications like games specially for the current challenge I faced today. At first, I thought arrays were only for storing numbers, and data types in which I could also utilize pointers and dereferencing them. But when I applied them in the Tic-Tac-Toe program, I saw how they could represent an actual game board where each move is tracked and updated. This made me appreciate how programming concepts connect to real-life scenarios. In doing these activities, I noticed a big difference between the two programs. The multiplication table program was easier to do since it only required storing values in a 2D array and printing them using nested loops. It was more straightforward and did not give me too many errors. On the other hand, the Tic-Tac-Toe program was more challenging because it required checking different winning conditions, switching players, and validating moves. I faced a lot of errors while trying to make it work properly, and I really had to look for examples to guide me in fixing the mistakes. Through this process, I realized that coding is never easy nor smooth and simple, but solving those errors step by step taught me patience and persistence. It also showed me that using references and learning from sources is part of the process of becoming better at programming. Overall, this activity made me appreciate how arrays can be applied in both simple computations and more complex, interactive programs. I also learned the importance of breaking down problems into smaller steps, like filling up the multiplication table with loops or checking different winning conditions in Tic-Tac-Toe. It was challenging at times, but I felt accomplished when the programs finally worked as expected. Overall, this activity stressed me a lot, especially on making a tic tac toe that moved, but it made me more comfortable with arrays, and gave me confidence that I can apply coding not just for academics but also for creativity and fun.