Activity 5.1	
Multidimensional Arrays	
Course Code: CPE 007	**Program:** Computer Engineering
Course Title: Programming Logic and Design	**Date Performed: September 25, 2025**
Section: CPE11S1	**Date Submitted: September 26, 2025**
Name(s): Jaime Luis M. Demain	**Instructor: Engr. Jimlord M. Quejado**

6. Output

1.

Code:

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

int main()
{
   int col, i, j, num;
   cout<<"Multiplication Table from 1 to 10:\n\n";
   for(i=1; i<=10; i++)
   {
      col = i;
      for(j=1; j<=10; j++)
      {
         num = col*j;
         cout<<num<<"  ";
      }
      cout<<endl;
   }
   cout<<endl;
   return 0;
}
```

Output:

```
Multiplication Table from 1 to 10:

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 0.02776 seconds with return value 0
Press any key to continue . . .
```

2.

Code:

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

int main() {
   char board[3][3];
   int count = '1';

   // Initialize board
   for (int i = 0; i < 3; i++)
       for (int j = 0; j < 3; j++)
           board[i][j] = count++;

   char player = ' ';
   char currentPlayer = 'X';

   // Display board
   while (player == ' ') {
       cout << "\n";
       for (int i = 0; i < 3; i++) {
           cout << " ";
           for (int j = 0; j < 3; j++) {
               cout << board[i][j];
               if (j < 2) cout << " | ";
           }
           cout << "\n";
           if (i < 2) cout << "---+---+---\n";
       }
       cout << "\n";

       // Player move
```

```
        int move;
        cout << "Player " << currentPlayer << ", enter your move (1-9): ";
        cin >> move;

        if (move < 1 || move > 9) {
            cout << "Invalid move. Please try again.\n";
            continue;
        }

        int row = (move - 1) / 3;
        int col = (move - 1) % 3;

        if (board[row][col] == 'X' || board[row][col] == 'O') {
            cout << "Spot already taken. Please try again.\n";
            continue;
        }

        board[row][col] = currentPlayer;

        // Check winner
        for (int i = 0; i < 3; i++) {
            if (board[i][0] == board[i][1] && board[i][1] == board[i][2])
                player = board[i][0];
            if (board[0][i] == board[1][i] && board[1][i] == board[2][i])
                player = board[0][i];
        }
        if (board[0][0] == board[1][1] && board[1][1] == board[2][2])
            player = board[0][0];
        if (board[0][2] == board[1][1] && board[1][1] == board[2][0])
            player = board[0][2];

        // Check for draw
        bool full = true;
        for (int i = 0; i < 3; i++)
            for (int j = 0; j < 3; j++)
                if (board[i][j] != 'X' && board[i][j] != 'O')
                    full = false;

        if (player == ' ' && full) player = 'D';

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

```
    }

    // Final board
    cout << "\n";
    for (int i = 0; i < 3; i++) {
        cout << " ";
        for (int j = 0; j < 3; j++) {
            cout << board[i][j];
            if (j < 2) cout << " | ";
        }
        cout << "\n";
        if (i < 2) cout << "---+---+---\n";
    }
    cout << "\n";

    // Result
    if (player == 'D')
        cout << "Draw!\n";
    else
        cout << "Player " << player << " wins!\n";

    return 0;
}
```

Output:

Draw Output:

"C:\Users\Jaime Luis\CLionProjects\untitled1\cmake-build-debug\untitled1.exe"

1 | 2 | 3
---+---+---
4 | 5 | 6
---+---+---
7 | 8 | 9

Player X, enter your move (1-9):10

Invalid move. Please try again.

1 | 2 | 3
---+---+---
4 | 5 | 6

```
---+---+---
 7 | 8 | 9

Player X, enter your move (1-9):1


 X | 2 | 3
---+---+---
 4 | 5 | 6
---+---+---
 7 | 8 | 9

Player O, enter your move (1-9):2


 X | O | 3
---+---+---
 4 | 5 | 6
---+---+---
 7 | 8 | 9

Player X, enter your move (1-9):5


 X | O | 3
---+---+---
 4 | X | 6
---+---+---
 7 | 8 | 9

Player O, enter your move (1-9):9


 X | O | 3
---+---+---
 4 | X | 6
---+---+---
 7 | 8 | O
```

```
Player X, enter your move (1-9):4


 X | O | 3
---+---+---
 X | X | 6
---+---+---
 7 | 8 | O

Player O, enter your move (1-9):7


 X | O | 3
---+---+---
 X | X | 6
---+---+---
 O | 8 | O

Player X, enter your move (1-9):8


 X | O | 3
---+---+---
 X | X | 6
---+---+---
 O | X | O

Player O, enter your move (1-9):6


 X | O | 3
---+---+---
 X | X | O
---+---+---
 O | X | O

Player X, enter your move (1-9):3
```

```
 X | O | X
---+---+---
 X | X | O
---+---+---
 O | X | O

Draw!

Process finished with exit code 0
```

Winning Output:

```
"C:\Users\Jaime Luis\CLionProjects\untitled1\cmake-build-debug\untitled1.exe"

 1 | 2 | 3
---+---+---
 4 | 5 | 6
---+---+---
 7 | 8 | 9

Player X, enter your move (1-9):1


 X | 2 | 3
---+---+---
 4 | 5 | 6
---+---+---
 7 | 8 | 9

Player O, enter your move (1-9):5


 X | 2 | 3
---+---+---
 4 | O | 6
---+---+---
 7 | 8 | 9
```

```
Player X, enter your move (1-9):4


 X | 2 | 3
---+---+---
 X | O | 6
---+---+---
 7 | 8 | 9

Player O, enter your move (1-9):7


 X | 2 | 3
---+---+---
 X | O | 6
---+---+---
 O | 8 | 9

Player X, enter your move (1-9):2


 X | X | 3
---+---+---
 X | O | 6
---+---+---
 O | 8 | 9

Player O, enter your move (1-9):3


 X | X | O
---+---+---
 X | O | 6
---+---+---
 O | 8 | 9

Player O wins!
```

Process finished with exit code 0

7. Supplementary Activity

1. Analysis: What I did in this part is making a multiplication table. The first step is to declare variables that will be used to program a multiplication table. And then, I will print out the top message that implies it is a Multiplication Table from 1-10 with two endlines. Then, I will now use for loops (nested loop version) to code the starting numbers and its product. The outer loop will be used to initialize a starting number from 1-10 by row. I used a "col" variable because the starting number increments by descending row. Then, the inner loop will be used to initialize a multiplication operator that will be used to process the products of a starting number (1-10). I used a "num" variable to represent the products of the starting number. That's all for the first item.

2. Analysis:

What I did in this part is making a tic-tac-toe. This part may be my most difficult and lengthy coding task to do as I need to learn something new (i.e Boolean variable type) and code it in almost 100 lines. The first part of this code is that I initialized a board for positions, numbering of positions, and the symbols (X and O) for the players. Then in the second part, I printed out the board with lines to mimic an actual tic-tac-toe board. This is also the first time that I used lines for programming tables and board. After this, in the third part, I initialize a player move mechanism so that players can input their desired positions on the board. Within this part, I also initialize a mechanism if the users will input numbers less than 1 and more than 9 so that their inputs can be invalidated and allow them to attempt inputting their numbers for the second or more time. Plus, this invalidation can also apply to the users who input their numbers that were already taken in previous attempts. And then, I also input a mechanism to precisely position the user's inputs so that the tic-tac-toe game will be properly executed. And then, in the fourth part, I coded a mechanism for winning users. Winning of the game will only happen if the inputs are matched in three consecutive positions horizontally, vertically, or diagonally. In horizontal form, the inputs must be all three column arrays under a one row array and no different row array. In vertical form, it is on vice-versa wherein inputs must be all three row arrays under a one column array. In diagonal form, inputs must all have the same number of both dimension arrays ([0][0], [1][1], [2][2]) in order to be recognized as a winner. In the fifth part, I coded a mechanism for drawing. So, in order for a draw to happen, all positions of the board must be fully filled and no three consecutive positions (straight line) of inputs must occur. I used a boolean data type for this part so that I can use both true or false on the "full" variable to determine if the board is full of inputs or not. Also under this part, if one position is not filled with inputs, the game must continue. In the sixth part, I coded a mechanism for switching player turns using ? and : operator. It will manually switch turns once a former user inputs a number. In the seventh and eight part, this is the final part wherein I will just basically print out the final form board like the proper board, number positions, endlines, and a declarative message if the game has a winner or it is a draw. Overall, this coding is so long in order to execute a tic-tac-toe game.

8. Conclusion

In this activity, I coded a program in a multi-dimensional array. It is important to get to know about multi-dimensional arrays to print out and program something in 2D like the tables and 2D board games. The first item is not hard for me as I already know its structure on how to program a simple table like that through Activity 4. I just need to fill all the tables because what I need to do is print out the products on the multiplication table. The second item is arguably the most difficult coding task that I did and it is also very long code as I need to code multiple mechanisms in order to make a tic-tac-toe game like player move mechanism, winning mechanism, draw mechanism and not just printing out a board. I just used my learnt functions in order to make this work. I learned a new function and that is the boolean function so that I can take my one variable either in true or false to make the draw mechanism work. Other than that, I used my learnt function in most parts of the coding. The output is about our coding and its output and the supplementary activity section

is about our analysis of an item. The analysis of the second item is somewhat long because of course the coding is also long, thus needing a longer takeaway from it.