Hands-on Activity 5.1	
Multidimensional Arrays	
Course Code: CPE 007	**Program:** Computer Engineering
Course Title: Programming Logic and Design	**Date Performed: 10/2/2025**
Section: CPE11S1	**Date Submitted: 10/2/2025**
Name(s): Juan Paulo C. Lara	**Instructor: Engr. Jimlord M. Quejado**

6. Output

#1

CODE

```cpp
#include <iostream>
#include <iomanip>
using namespace std;
int main () {
    int number;
    cout << "Input number: ";
    cin >> number;

    int table[number+1][number+1];

    for (int i = 0; i <= number; i++) {
        for (int j = 1; j <= number; j++) {
            table[i][j] = i * j;
        }
    }

    cout << "Multiplication Table for " << number << "\n";
    for (int i = 0; i <= number; i++) {
        for (int j = 1; j <= number; ++j) {
            cout << setw(4) << table[i][j];
         }
         cout << endl;
    }

    return 0;
}
```

OUTPUT

```
Input number: 5
Multiplication Table for 5
   0   0   0   0   0
   1   2   3   4   5
   2   4   6   8  10
   3   6   9  12  15
   4   8  12  16  20
   5  10  15  20  25

--------------------------------
Process exited after 7.628 seconds with return value 0
Press any key to continue . . .
```

#2

CODE

```cpp
#include <iostream>
#include <cstring>
using namespace std;
int main() {
    char board[3][3];
    char symbols[2] = {'X','O'};
    int current = 0, filled = 0;
    int r,c;

    for (int i=0; i<3; i++) {
        for (int j=0; j<3; j++) {
            board[i][j] = ' ';
        }
    }

    while (filled < 9) {
        cout << "-------------\n";
        for (int i=0;i<3;i++) {
            cout << "| ";
            for (int j=0;j<3;j++) cout << board[i][j] << " | ";
            cout << "\n-------------\n";
        }

        while (true) {
            cout << symbols[current] << " turn. Enter row & column (1-3): ";
            cin >> r >> c; r--; c--;
            if (r>=0 && r<3 && c>=0 && c<3 && board[r][c]==' ')
                break;
            cout << "Invalid. Try again.\n";
        }
```

```
31
32          board[r][c] = symbols[current];
33          filled++;
34
35          bool win = false;
36          for (int i=0;i<3;i++)
37              if ((board[i][0]==symbols[current] && board[i][1]==symbols[current] && board[i][2]==symbols[current]) ||
38                  (board[0][i]==symbols[current] && board[1][i]==symbols[current] && board[2][i]==symbols[current]))
39                  win = true;
40          if ((board[0][0]==symbols[current] && board[1][1]==symbols[current] && board[2][2]==symbols[current]) ||
41              (board[0][2]==symbols[current] && board[1][1]==symbols[current] && board[2][0]==symbols[current]))
42              win = true;
43
44          if (win) {
45              cout << symbols[current] << " wins\n";
46              break;
47          }
48
49          current = 1 - current;
50          if (filled==9)
51              cout << "Draw\n";
52      }
53      return 0;
54  }
```

OUTPUT

Winning

```
-------------
|   |   |   |
-------------
|   |   |   |
-------------
|   |   |   |
-------------
X turn. Enter row & column (1-3): 1 2
-------------
|   | X |   |
-------------
|   |   |   |
-------------
|   |   |   |
-------------
O turn. Enter row & column (1-3): 2 2
-------------
|   | X |   |
-------------
|   | O |   |
-------------
|   |   |   |
-------------
X turn. Enter row & column (1-3): 3 3
-------------
|   | X |   |
-------------
|   | O |   |
-------------
|   |   | X |
-------------
O turn. Enter row & column (1-3): 3 1
-------------
|   | X |   |
-------------
|   | O |   |
-------------
| O |   | X |
-------------
X turn. Enter row & column (1-3): 2 3
-------------
|   | X |   |
-------------
|   | O | X |
-------------
| O |   | X |
-------------
O turn. Enter row & column (1-3): 1 3
O wins

--------------------------------
Process exited after 15.97 seconds with return value 0
Press any key to continue . . .
```

Draw

```
-------------
|   |   |   |
-------------
|   |   |   |
-------------
|   |   |   |
-------------
X turn. Enter row & column (1-3): 1 1
-------------
| X |   |   |
-------------
|   |   |   |
-------------
|   |   |   |
-------------
O turn. Enter row & column (1-3): 2 2
-------------
| X |   |   |
-------------
|   | O |   |
-------------
|   |   |   |
-------------
X turn. Enter row & column (1-3): 3 3
-------------
| X |   |   |
-------------
|   | O |   |
-------------
|   |   | X |
-------------
O turn. Enter row & column (1-3): 1 3
-------------
| X |   | O |
-------------
|   | O |   |
-------------
|   |   | X |
-------------
X turn. Enter row & column (1-3): 3 1
```

```
------------
| X |   | O |
------------
|   | O |   |
------------
| X |   | X |
------------
O turn. Enter row & column (1-3): 3 2
------------
| X |   | O |
------------
|   | O |   |
------------
| X | O | X |
------------
X turn. Enter row & column (1-3): 1 2
------------
| X | X | O |
------------
|   | O |   |
------------
| X | O | X |
------------
O turn. Enter row & column (1-3): 2 1
------------
| X | X | O |
------------
| O | O |   |
------------
| X | O | X |
------------
X turn. Enter row & column (1-3): 2 3
Draw

--------------------------------
Process exited after 38.7 seconds with return value 0
Press any key to continue . . .
```

7. Supplementary Activity

#1 ANALYSIS

When running this code, first, line 5 declares the variable needed for the input value for line 6-7, which asks for user input. Line 9 declares an array with the size from the input number value twice to form a x times x multiplication table. A for loop at line 11 scans from 0 to the input number then loops through rows and columns i and j in line 11 and 12. At line 13, each part of the table is placed a product from the designated row and column. Line 17 prints out a label for the inputted number as the value used for the multiplication table. A for loop at line 18 and 19 looks up the processed inputs from the previous for loop, then line 20 prints the table data with setw(4) for proper spacing, and endl at line 22 allows returning to the first column at the next line.

#2 ANALYSIS

During execution of this program, from line 5 to 8, this block of code contains the initialized variables and board grid. Line 5 sets up the grid of the board. Line 6 forms an array of two characters X and O. Line 7 variable current looks for player turns and filled looks for spaces filled up by player inputs. A for loop from line 10 to 14 prints out blanks to form an empty grid at the start of the program. Beginning at line 16, this is where the game loop starts until it meets a break from the logic conditions inside the loop. Before a for loop, line 17 outputs a line of dash symbols to form a line for the grid. A for loop at line 18 to 22 prints out the board and inside grids. An inner while loop at line 24 starts with line 25 outputting the

current side with a message to input a row and column (e.g. 2 3). Line 26 takes the input from the user and uses a decrement to account for placement in the grid, then at line 27, it checks whether an input X or O is in place or empty in a grid. When valid, the loop breaks, and if not, then line 29 outputs an error message then it waits again for input. Line 32 prints the input symbol. Line 33 increments the filled variable as a counter. Line 35 declares a boolean winning condition for the following game logic. From line 35 to 47, a for loop containing 2 if statements checks for the 3 rows and columns in a row and looks for diagonal patterns, and if a condition is met, the win variable becomes true and heads over to the third if statement to output the winner and ends the program. Line 49 cycles the turns between X and O. An if statement at line 50 checks if the inputs are full to print a draw. .

8. Conclusion

Concluding this activity, I gained understanding in the use of multidimensional arrays and learned of its extended capabilities of having more than one dimension in an array. During the making of this activity, multidimensional arrays allowed for printing tables with input values, reading and writing the elements in each row and column. In the first activity, I learned more uses of the iomanip library through the use of the setw() variable, which allowed for a more consistent spacing when printing out the values in the array. For the second activity, forming a board was similar to making the spacings in the first one but this time, it involves more characters to form the board of tic tac toe. The use of logical operators and if statements is also possible with arrays, allowing me to make a tic tac toe game with winning and draw conditions. Overall, this activity expanded my learning of arrays and the ways to read, manipulate, and print the values. The tables also serve as representations of how the computer reads and writes arrays/tables of data.