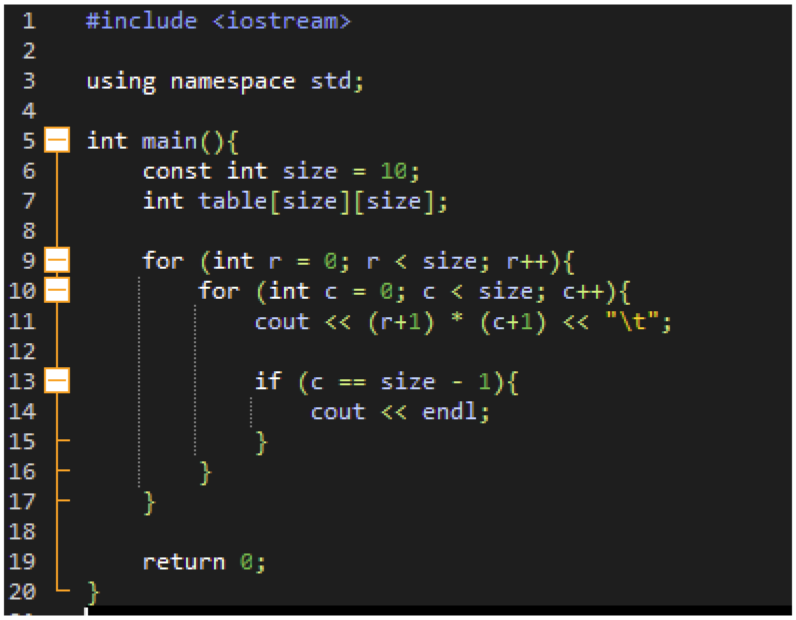
Hands-on Activity 5.1	
Multidimensional Arrays	
Course Code: CPE007	**Program:** Computer Engineering
Course Title: Programming Logic and Design	**Date Performed:9-29-25**
Section:CPE11S1	**Date Submitted:9-30-25**
Name(s): Niel Vincent B. Condino	**Instructor: Engr. Jimlord M. Quejado**

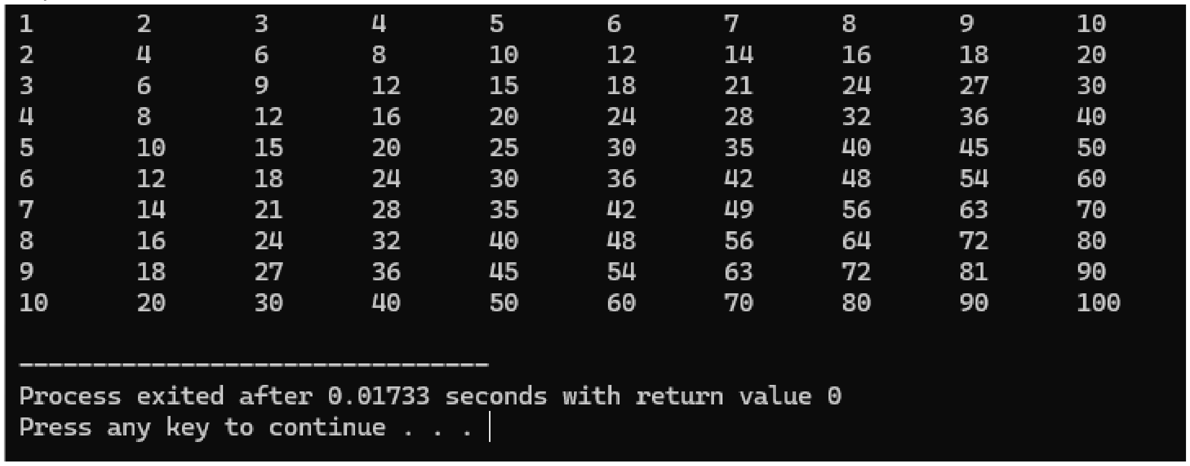
6. Output

Code:



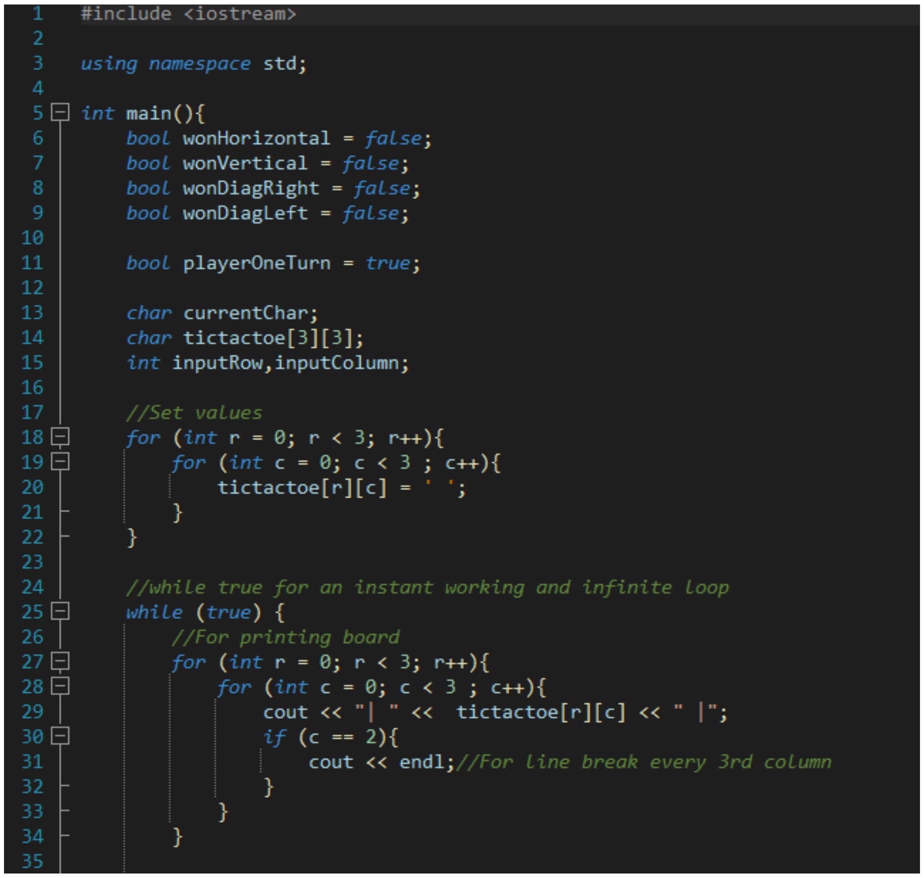
```
#include <iostream>

using namespace std;

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

    for (int r = 0; r < size; r++){
        for (int c = 0; c < size; c++){
            cout << (r+1) * (c+1) << "\t";

            if (c == size - 1){
                cout << endl;
            }
        }
    }

    return 0;
}
```

Output

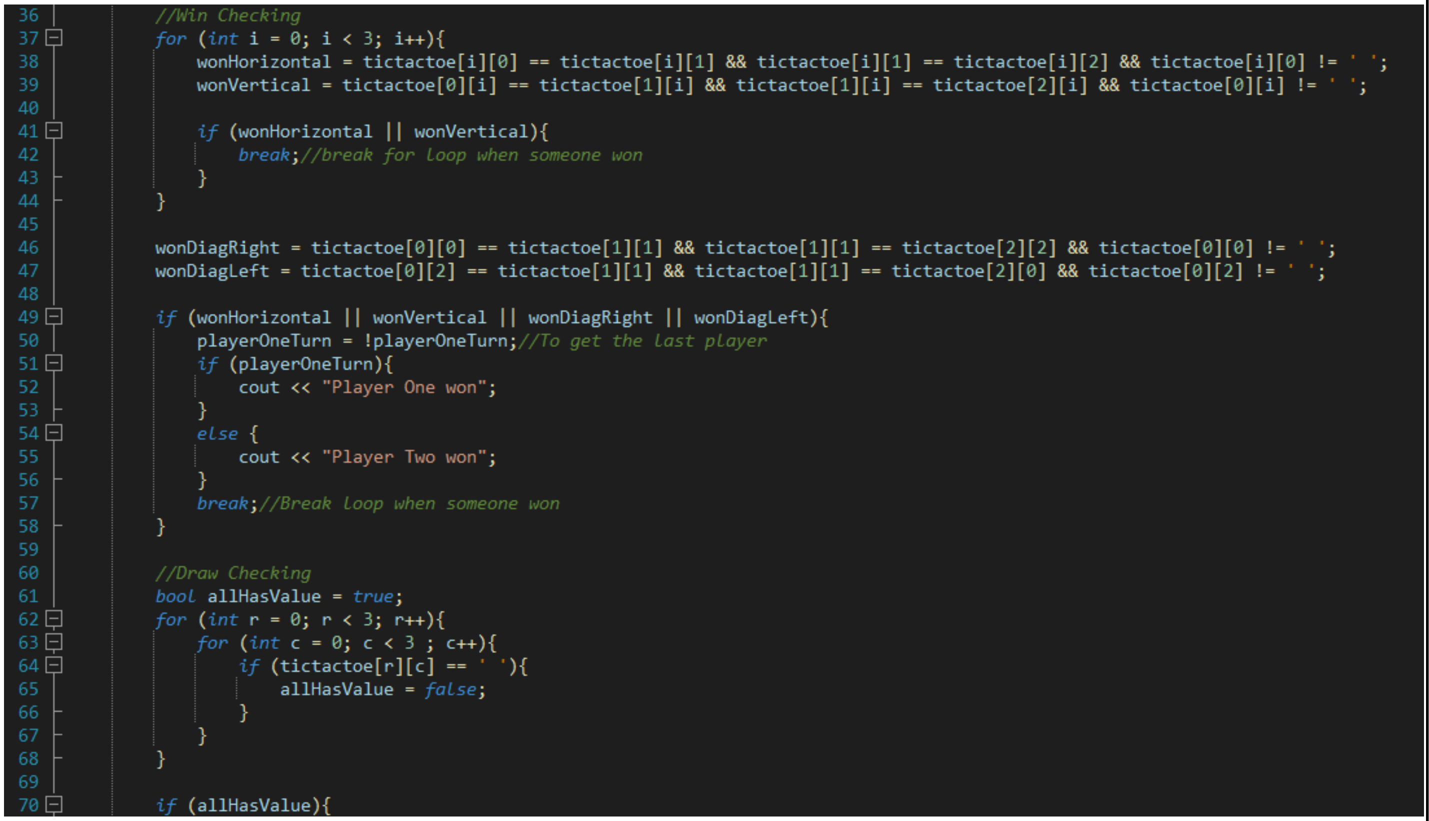


```
1       2       3       4       5       6       7       8       9       10
2       4       6       8       10      12      14      16      18      20
3       6       9       12      15      18      21      24      27      30
4       8       12      16      20      24      28      32      36      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 0.01733 seconds with return value 0
Press any key to continue . . .
```

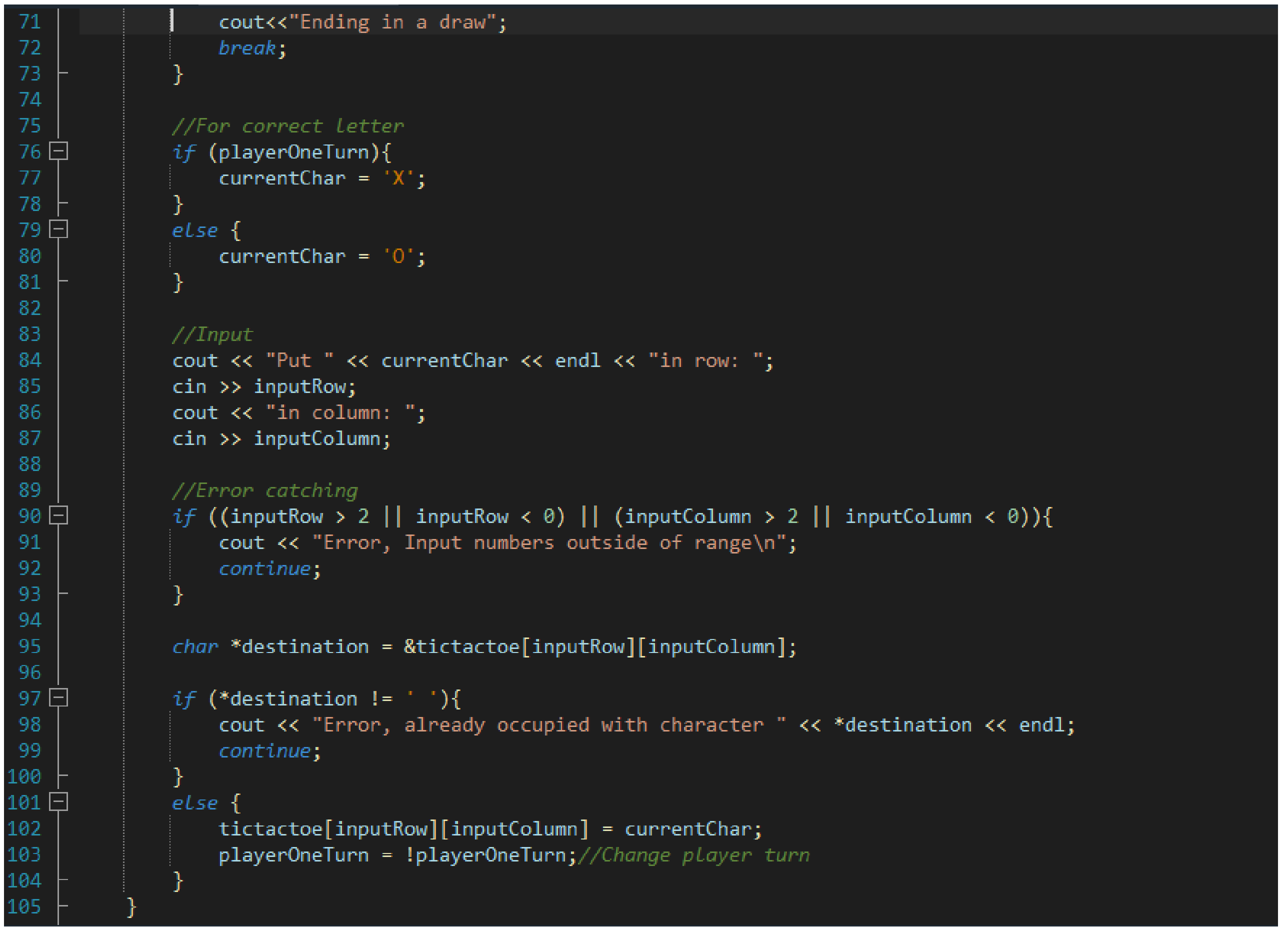
Code:



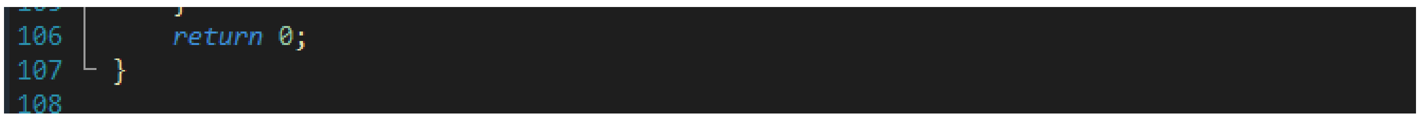
```cpp
#include <iostream>

using namespace std;

int main(){
    bool wonHorizontal = false;
    bool wonVertical = false;
    bool wonDiagRight = false;
    bool wonDiagLeft = false;

    bool playerOneTurn = true;

    char currentChar;
    char tictactoe[3][3];
    int inputRow,inputColumn;

    //Set values
    for (int r = 0; r < 3; r++){
        for (int c = 0; c < 3 ; c++){
            tictactoe[r][c] = ' ';
        }
    }

    //while true for an instant working and infinite loop
    while (true) {
        //For printing board
        for (int r = 0; r < 3; r++){
            for (int c = 0; c < 3 ; c++){
                cout << "| " <<  tictactoe[r][c] << " |";
                if (c == 2){
                    cout << endl;//For line break every 3rd column
                }
            }
        }

```



```cpp
        //Win Checking
        for (int i = 0; i < 3; i++){
            wonHorizontal = tictactoe[i][0] == tictactoe[i][1] && tictactoe[i][1] == tictactoe[i][2] && tictactoe[i][0] != ' ';
            wonVertical = tictactoe[0][i] == tictactoe[1][i] && tictactoe[1][i] == tictactoe[2][i] && tictactoe[0][i] != ' ';

            if (wonHorizontal || wonVertical){
                break;//break for loop when someone won
            }
        }

        wonDiagRight = tictactoe[0][0] == tictactoe[1][1] && tictactoe[1][1] == tictactoe[2][2] && tictactoe[0][0] != ' ';
        wonDiagLeft = tictactoe[0][2] == tictactoe[1][1] && tictactoe[1][1] == tictactoe[2][0] && tictactoe[0][2] != ' ';

        if (wonHorizontal || wonVertical || wonDiagRight || wonDiagLeft){
            playerOneTurn = !playerOneTurn;//To get the last player
            if (playerOneTurn){
                cout << "Player One won";
            }
            else {
                cout << "Player Two won";
            }
            break;//Break loop when someone won
        }

        //Draw Checking
        bool allHasValue = true;
        for (int r = 0; r < 3; r++){
            for (int c = 0; c < 3 ; c++){
                if (tictactoe[r][c] == ' '){
                    allHasValue = false;
                }
            }
        }

        if (allHasValue){
```

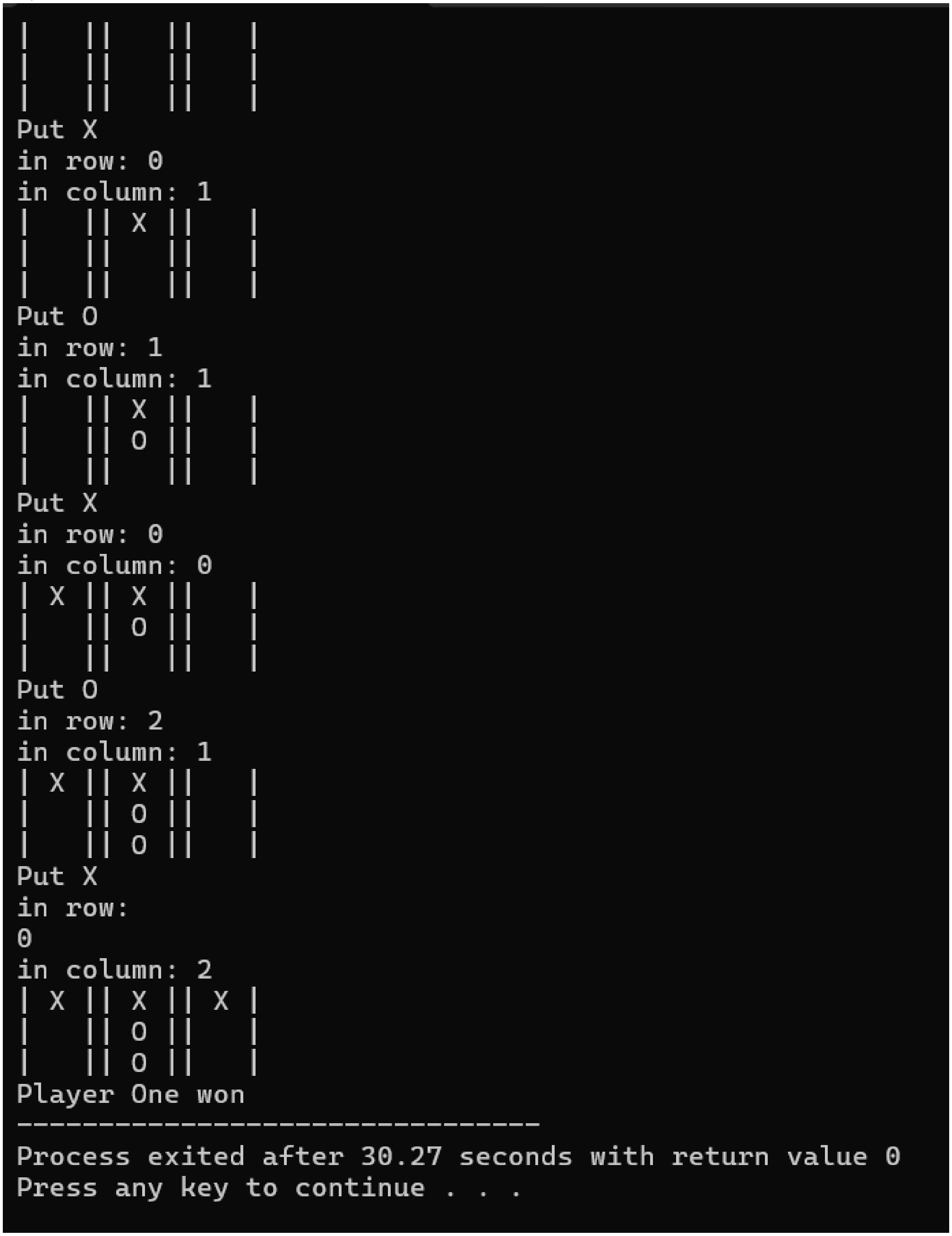


```cpp
            cout<<"Ending in a draw";
            break;
        }

        //For correct letter
        if (playerOneTurn){
            currentChar = 'X';
        }
        else {
            currentChar = 'O';
        }

        //Input
        cout << "Put " << currentChar << endl << "in row: ";
        cin >> inputRow;
        cout << "in column: ";
        cin >> inputColumn;

        //Error catching
        if ((inputRow > 2 || inputRow < 0) || (inputColumn > 2 || inputColumn < 0)){
            cout << "Error, Input numbers outside of range\n";
            continue;
        }

        char *destination = &tictactoe[inputRow][inputColumn];

        if (*destination != ' '){
            cout << "Error, already occupied with character " << *destination << endl;
            continue;
        }
        else {
            tictactoe[inputRow][inputColumn] = currentChar;
            playerOneTurn = !playerOneTurn;//Change player turn
        }
    }
```

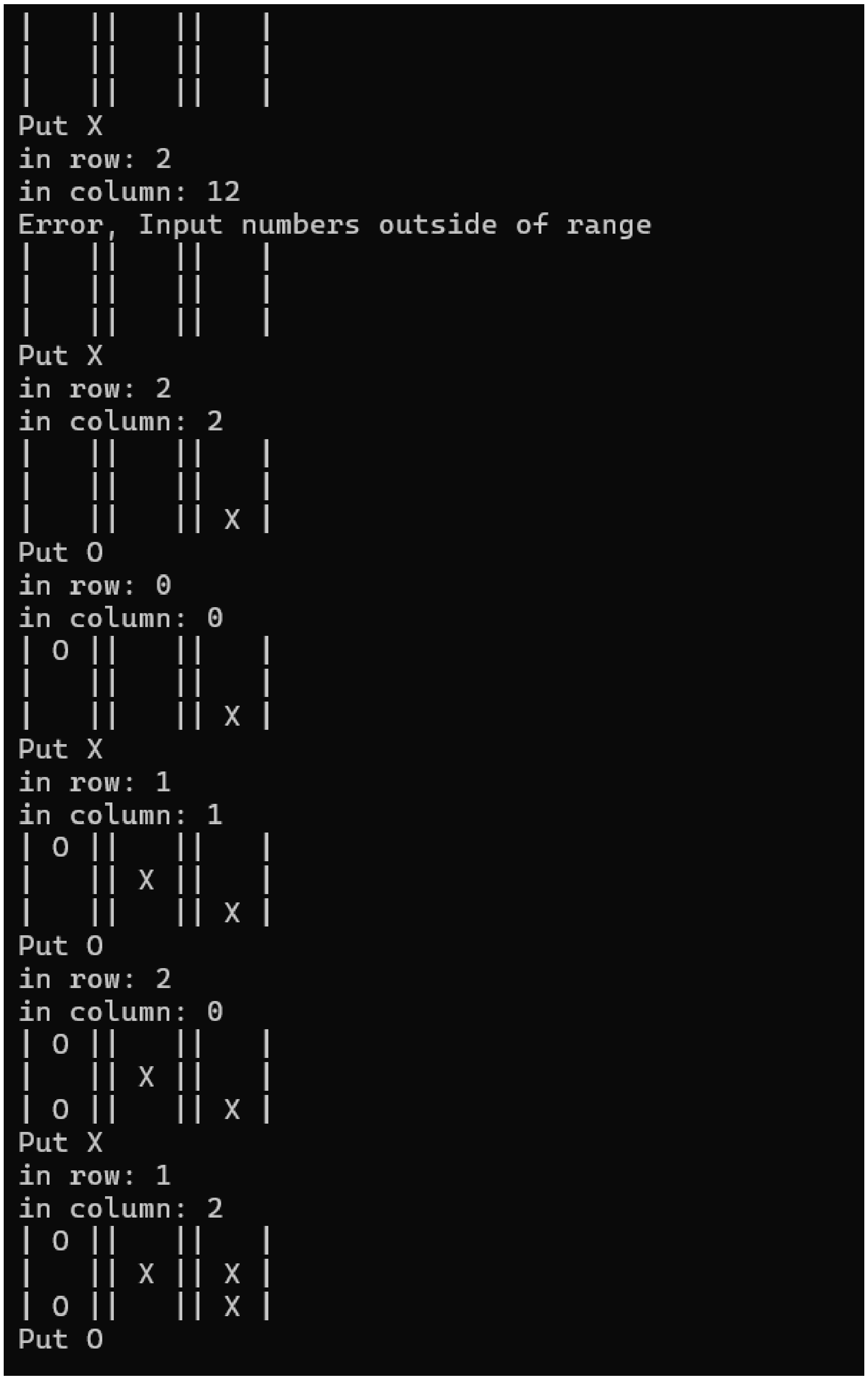


```cpp
    return 0;
}
```

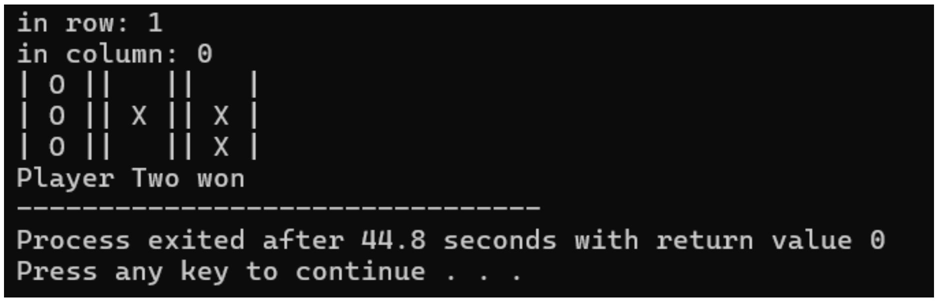
Output:



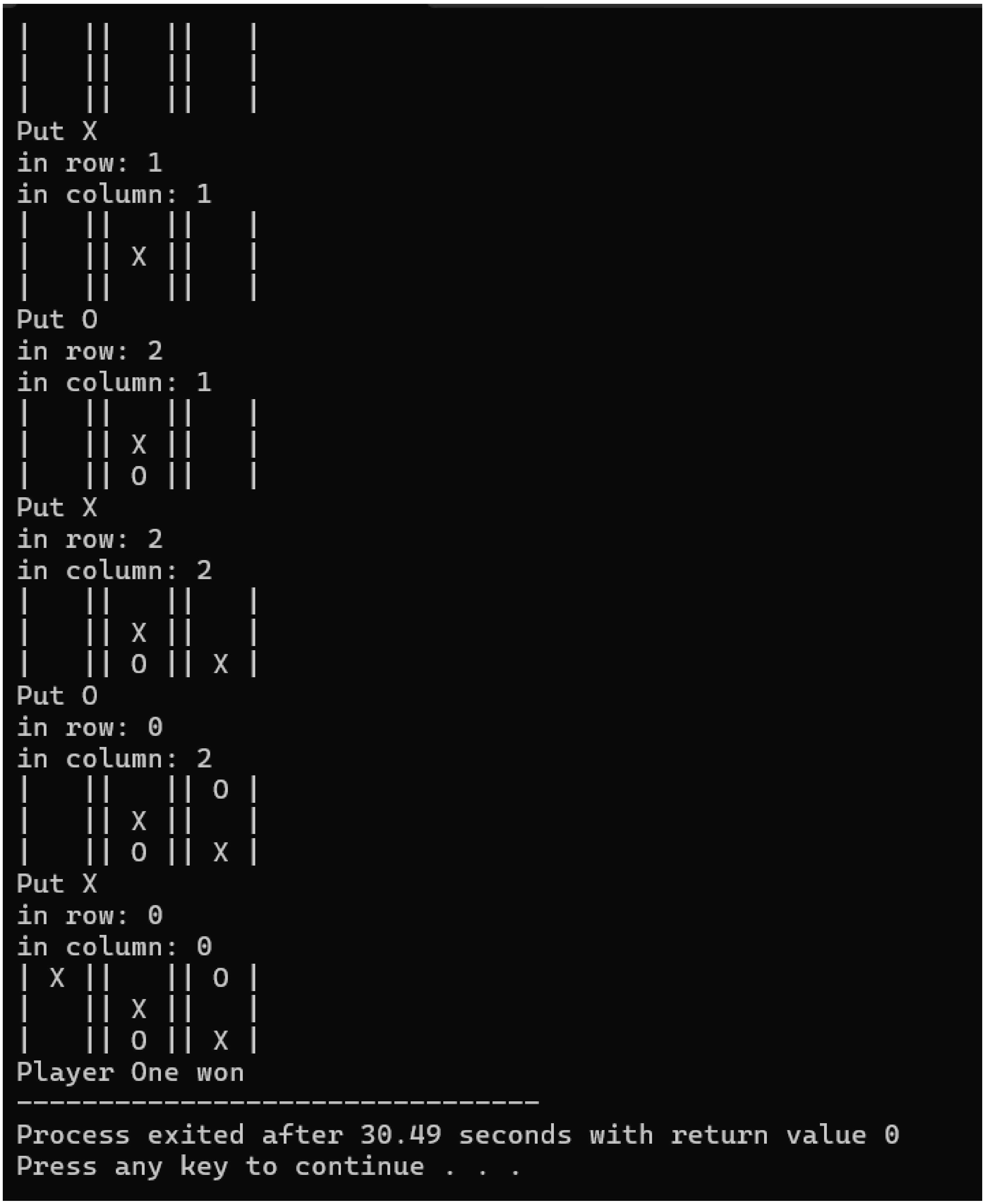
```
|   ||   ||   |
|   ||   ||   |
|   ||   ||   |
Put X
in row: 0
in column: 1
|   || X ||   |
|   ||   ||   |
|   ||   ||   |
Put O
in row: 1
in column: 1
|   || X ||   |
|   || O ||   |
|   ||   ||   |
Put X
in row: 0
in column: 0
| X || X ||   |
|   || O ||   |
|   ||   ||   |
Put O
in row: 2
in column: 1
| X || X ||   |
|   || O ||   |
|   || O ||   |
Put X
in row:
0
in column: 2
| X || X || X |
|   || O ||   |
|   || O ||   |
Player One won
--------------------------------
Process exited after 30.27 seconds with return value 0
Press any key to continue . . .
```



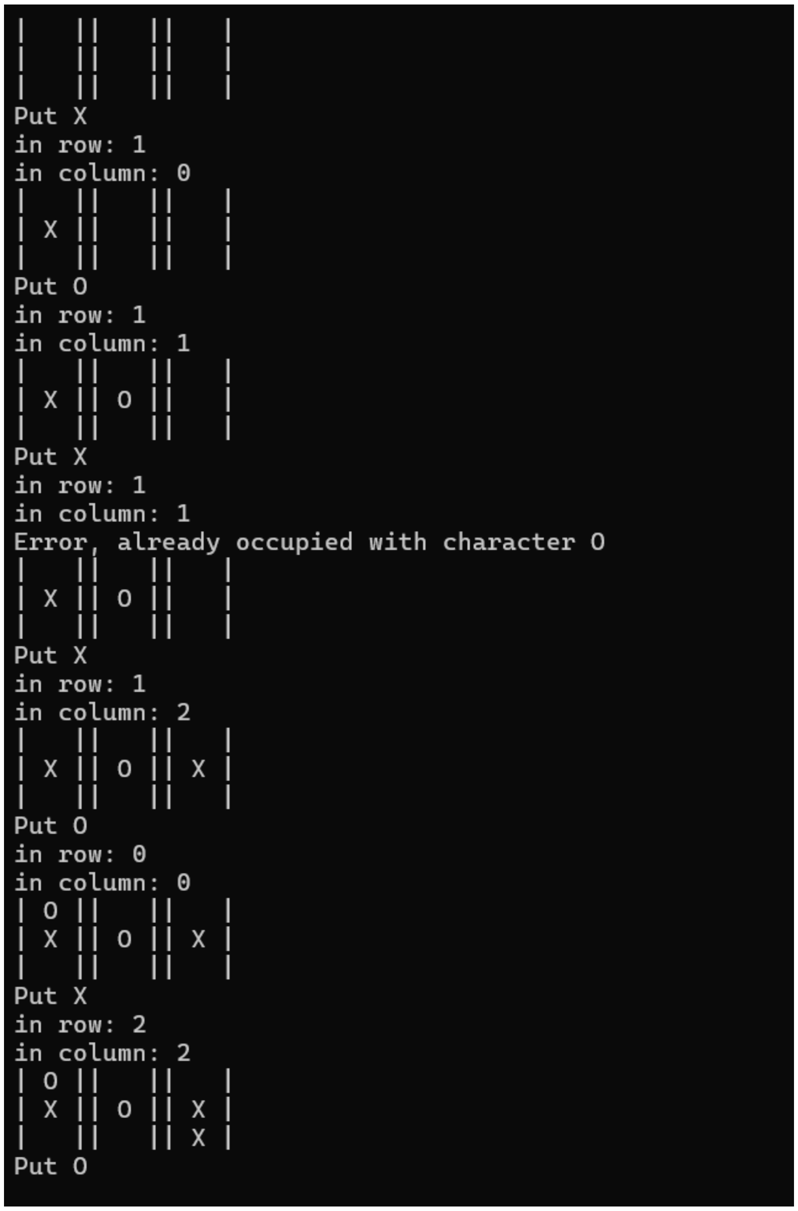
```
|   ||   ||   |
|   ||   ||   |
|   ||   ||   |
Put X
in row: 2
in column: 12
Error, Input numbers outside of range
|   ||   ||   |
|   ||   ||   |
|   ||   ||   |
Put X
in row: 2
in column: 2
|   ||   ||   |
|   ||   ||   |
|   ||   || X |
Put O
in row: 0
in column: 0
| O ||   ||   |
|   ||   ||   |
|   ||   || X |
Put X
in row: 1
in column: 1
| O ||   ||   |
|   || X ||   |
|   ||   || X |
Put O
in row: 2
in column: 0
| O ||   ||   |
|   || X ||   |
| O ||   || X |
Put X
in row: 1
in column: 2
| O ||   ||   |
|   || X || X |
| O ||   || X |
Put O
```



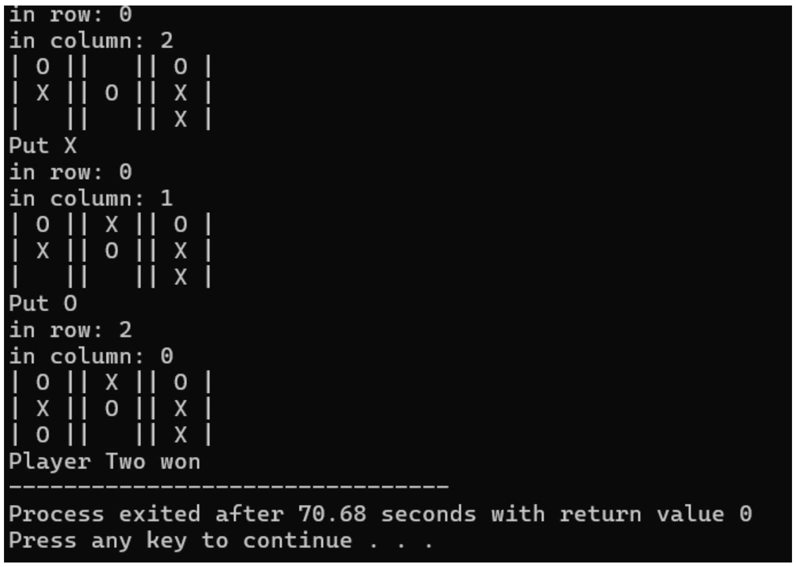
```
in row: 1
in column: 0
| O ||   ||   |
| O || X || X |
| O ||   || X |
Player Two won
--------------------------------
Process exited after 44.8 seconds with return value 0
Press any key to continue . . .
```



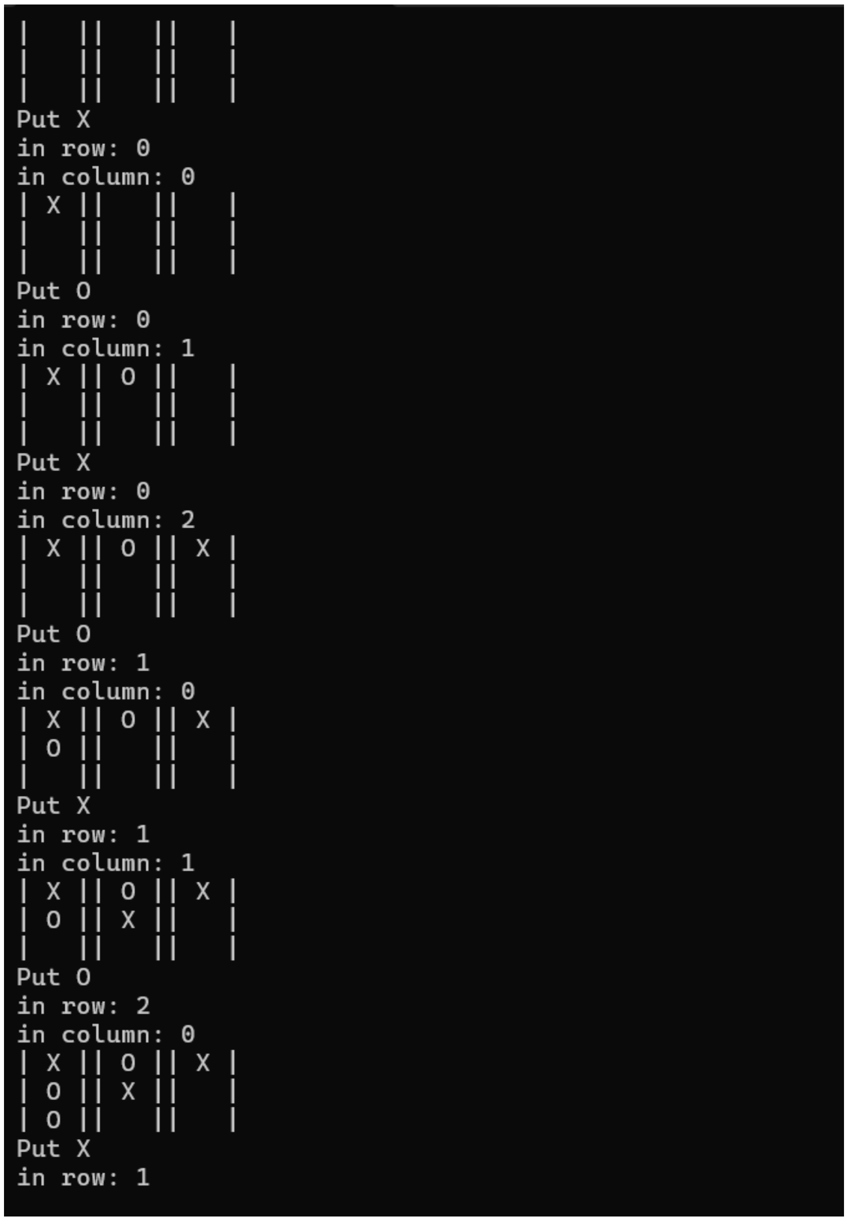
```
|   ||   ||   |
|   ||   ||   |
|   ||   ||   |
Put X
in row: 1
in column: 1
|   ||   ||   |
|   || X ||   |
|   ||   ||   |
Put O
in row: 2
in column: 1
|   ||   ||   |
|   || X ||   |
|   || O ||   |
Put X
in row: 2
in column: 2
|   ||   ||   |
|   || X ||   |
|   || O || X |
Put O
in row: 0
in column: 2
|   ||   || O |
|   || X ||   |
|   || O || X |
Put X
in row: 0
in column: 0
| X ||   || O |
|   || X ||   |
|   || O || X |
Player One won
--------------------------------
Process exited after 30.49 seconds with return value 0
Press any key to continue . . .
```



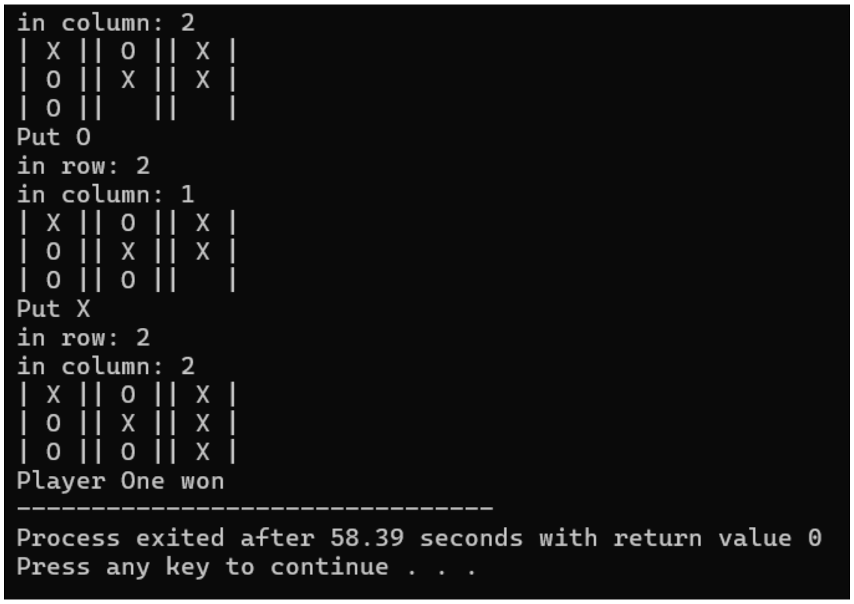
```
|   ||   ||   |
|   ||   ||   |
|   ||   ||   |
Put X
in row: 1
in column: 0
|   ||   ||   |
| X ||   ||   |
|   ||   ||   |
Put O
in row: 1
in column: 1
|   ||   ||   |
| X || O ||   |
|   ||   ||   |
Put X
in row: 1
in column: 1
Error, already occupied with character O
|   ||   ||   |
| X || O ||   |
|   ||   ||   |
Put X
in row: 1
in column: 2
|   ||   ||   |
| X || O || X |
|   ||   ||   |
Put O
in row: 0
in column: 0
| O ||   ||   |
| X || O || X |
|   ||   ||   |
Put X
in row: 2
in column: 2
| O ||   ||   |
| X || O || X |
|   ||   || X |
Put O
```



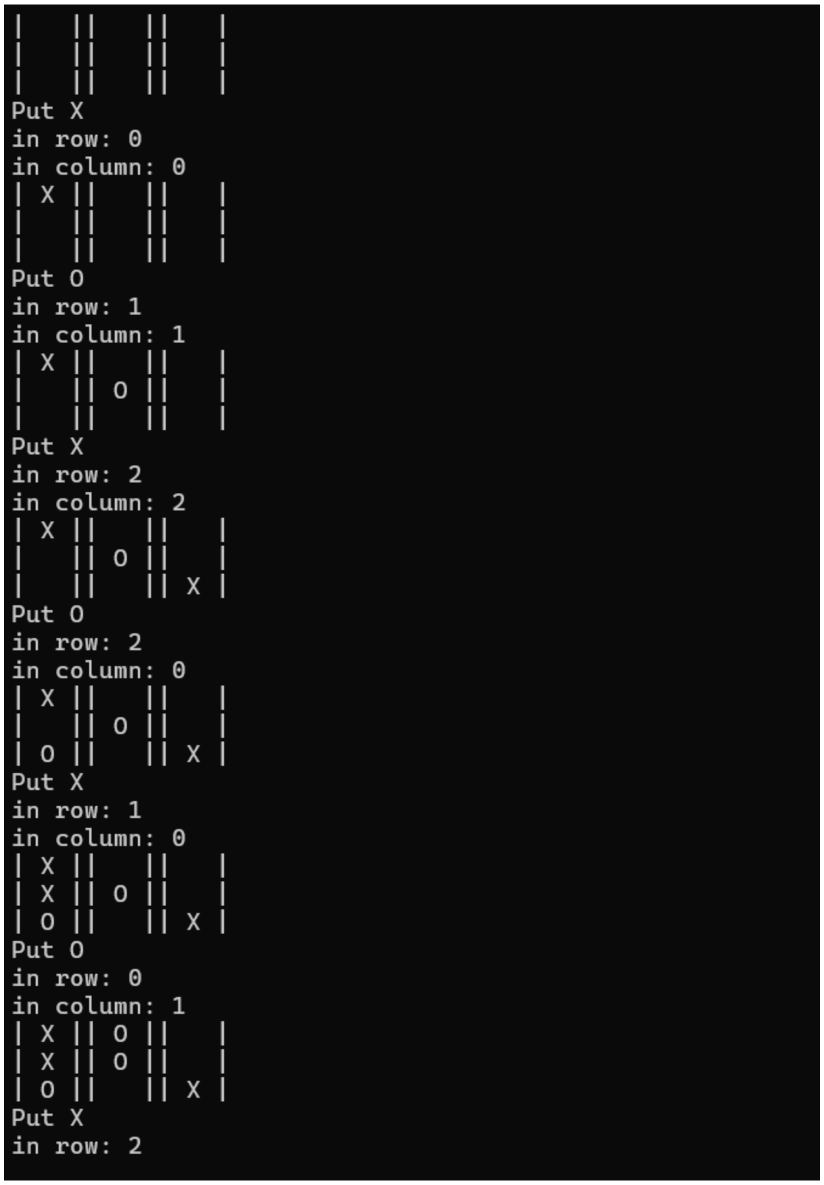
```
in row: 0
in column: 2
| O ||   || O |
| X || O || X |
|   ||   || X |
Put X
in row: 0
in column: 1
| O || X || O |
| X || O || X |
|   ||   || X |
Put O
in row: 2
in column: 0
| O || X || O |
| X || O || X |
| O ||   || X |
Player Two won
--------------------------------
Process exited after 70.68 seconds with return value 0
Press any key to continue . . .
```



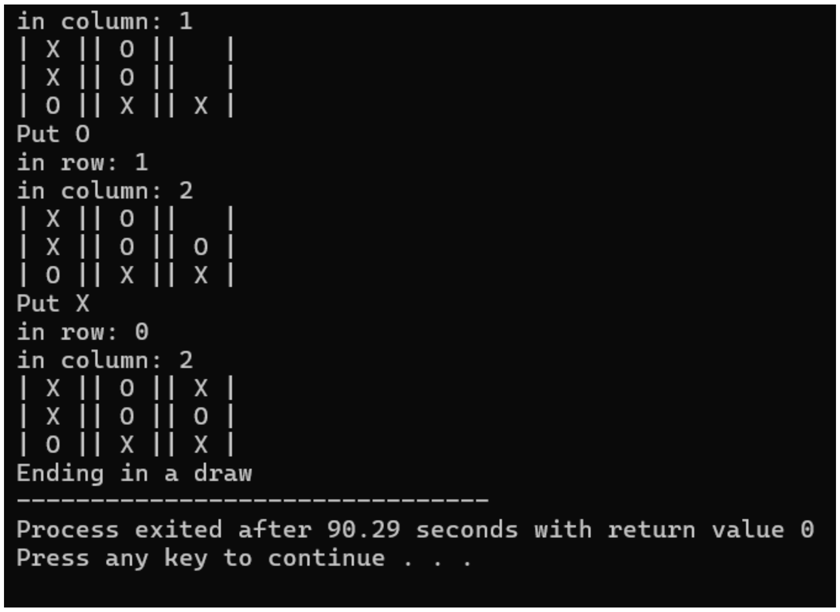
```
|   ||   ||   |
|   ||   ||   |
|   ||   ||   |
Put X
in row: 0
in column: 0
| X ||   ||   |
|   ||   ||   |
|   ||   ||   |
Put O
in row: 0
in column: 1
| X || O ||   |
|   ||   ||   |
|   ||   ||   |
Put X
in row: 0
in column: 2
| X || O || X |
|   ||   ||   |
|   ||   ||   |
Put O
in row: 1
in column: 0
| X || O || X |
| O ||   ||   |
|   ||   ||   |
Put X
in row: 1
in column: 1
| X || O || X |
| O || X ||   |
|   ||   ||   |
Put O
in row: 2
in column: 0
| X || O || X |
| O || X ||   |
| O ||   ||   |
Put X
in row: 1
```



```
in column: 2
| X || O || X |
| O || X || X |
| O ||   ||   |
Put O
in row: 2
in column: 1
| X || O || X |
| O || X || X |
| O || O ||   |
Put X
in row: 2
in column: 2
| X || O || X |
| O || X || X |
| O || O || X |
Player One won
--------------------------------
Process exited after 58.39 seconds with return value 0
Press any key to continue . . .
```



```
|   ||   ||   |
|   ||   ||   |
|   ||   ||   |
Put X
in row: 0
in column: 0
| X ||   ||   |
|   ||   ||   |
|   ||   ||   |
Put O
in row: 1
in column: 1
| X ||   ||   |
|   || O ||   |
|   ||   ||   |
Put X
in row: 2
in column: 2
| X ||   ||   |
|   || O ||   |
|   ||   || X |
Put O
in row: 2
in column: 0
| X ||   ||   |
|   || O ||   |
| O ||   || X |
Put X
in row: 1
in column: 0
| X ||   ||   |
| X || O ||   |
| O ||   || X |
Put O
in row: 0
in column: 1
| X || O ||   |
| X || O ||   |
| O ||   || X |
Put X
in row: 2
```



```
in column: 1
| X || O ||   |
| X || O ||   |
| O || X || X |
Put O
in row: 1
in column: 2
| X || O ||   |
| X || O || O |
| O || X || X |
Put X
in row: 0
in column: 2
| X || O || X |
| X || O || O |
| O || X || X |
Ending in a draw
--------------------------------
Process exited after 90.29 seconds with return value 0
Press any key to continue . . .
```

7. Supplementary Activity

Analysis:

Multiplication Table:
Inside the “main” function, it first initializes a constant integer named “size” with a value of 10. After it, it initializes a 2d array named “table” with both of its sizes being 10. The outer for loop loops through the rows of the array. The inner loop loops in each of the columns. Inside of this loop, it prints the value of (r+1) and (c+1) with “\t” for better formatting. I used (r+1) and (c+1) instead of just initializing them from the start as 1 is because I want it to easily be editable by just changing the “size” variable’s value. Inside also of this function is an if function that checks if the column or c is equal to size - 1. If it is then it creates a line break for better formatting.Lastly it returns 0.

Tic tac toe:
Inside the “main” function, it first initializes bool variables named “wonHorizontal”, “wonVertical”, “wonDiagLeft”, “wonDiagRight” to track whether any player has won horizontally, vertically, or diagonally. It also initializes “playerOneTurn” as true to keep track of whose turn it is. It also initializes the variable “currentChar” where the character will be stored and a 2d array named “tictactoe” with a both of its sizes being 3 .Next, the program fills the board with ' ' characters by using two for nested loops, making the board empty at the start.After this, the program enters an infinite loop. Inside it, the board is printed using a for loops. Each row and column is looped through, printing the character at that position followed by a "|". After the last column of each row, a line break is added for formatting.The code then performs win checking. A for loop loops through the rows and columns of the array to check if there are three equal characters in any row , column , or left and right diagonals that are not blank characters. If any of these win conditions are met, the program first reverts the playerOneTurn variable to get the last player and prints whether Player One or Player Two won, and breaks the loop to end the game.If no player has won, the program checks for a draw. It initializes a bool variable allHasValue with a value of true and scans the board again. If any cell is still blank ', it sets allHasValue = false. If no empty cells remain and nobody has won, it prints "Ending in a draw" and breaks the loop. If the game continues, the program determines whose turn it is by checking playerOneTurn. If it’s true, currentChar is set to 'X'; otherwise, it is set to 'O'.The program then takes input for row and column positions from the user. Error handling if statements checks whether the input is just between 0-2. If the numbers are invalid, it prints an error message and skips to the next iteration of the loop.It also checks whether the chosen cell is already occupied. If it is not empty, it prints an error and asks again. Otherwise, it places the current player’s character into that cell.Finally, the turn is switched by toggling playerOneTurn to the opposite value, and the game loop continues until either a player wins or the game ends in a draw.At the very end, it returns 0.

8. Conclusion

The concept of 2d arrays was already tackled a while ago. 2d arrays are just basically a data holder that has a row and a column like a spreadsheet.The understanding of concept helped to make this activity a little less difficult.On the output part,the multiplication table is pretty easy to do as I just need the concept of it multiplying the number above and number on the side with a loop. The tictactoe code however is pretty challenging to do. It took me a while to finish the winner checking part of it. It took me some time to try and find a solution for its checking to not be hardcoded, but in the end, I did not find a way to do it automatically without hardcoding it. The other parts of it however are pretty challenging but I did it within the laboratory time. I find the other parts as not as challenging as the win checking part as it is only printing,input,checking of input, and checking if at least one value is a blank character. I find the draw checking algorithm being far easier to find without hardcoding it than the win checking code as it has multiple checks to do. The code analysis is pretty easy to write. However, it took time to explain fully the code like for the titactoe one which is fairly long. It took longer to write it than the code analysis of the first one because it was more complicated than the multiplication table program.Overall, I think even though I fairly understand the concept of 2d arrays, I still need to improve upon it. I feel there are still improvements that can be done on the tictactoe program and also improvement on my knowledge about 2 and above dimensional arrays. Learning about the functions, I think the tictactoe program can be further optimized using that in order to shorten it and improve readability.

9. Assessment Rubric