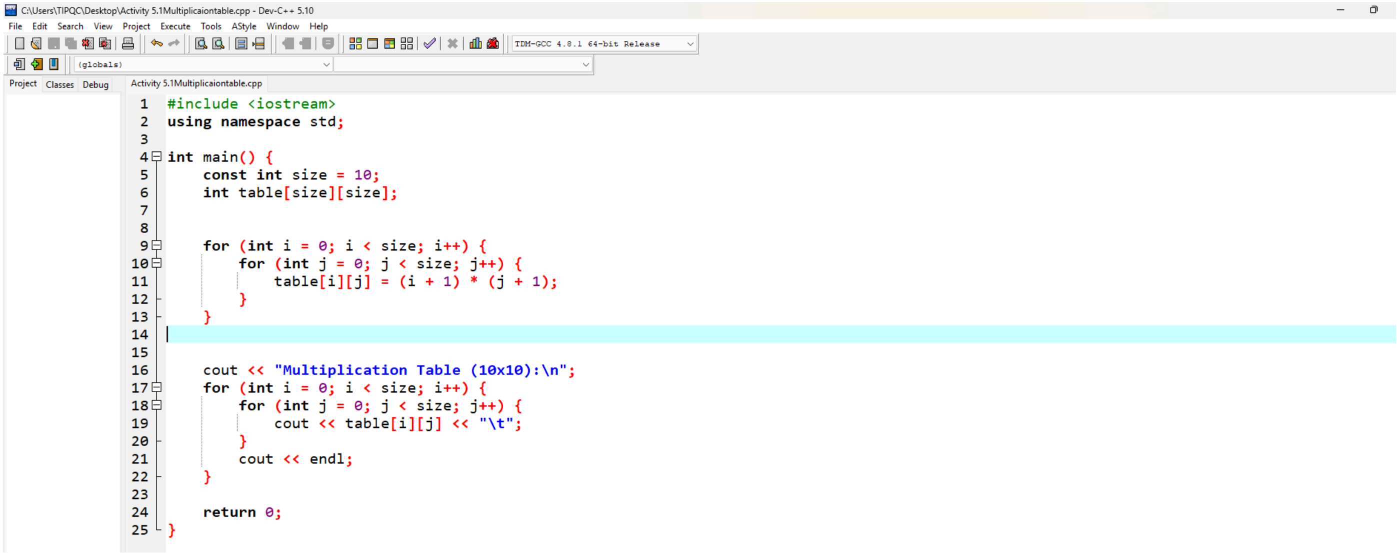
Activity No. n

Replace with Title

Course Code:	**Program:** Computer Engineering
Course Title:	**Date Performed:**
Section:	**Date Submitted:**
Name(s):	**Instructor: Engr. Jimlord M. Quejado**

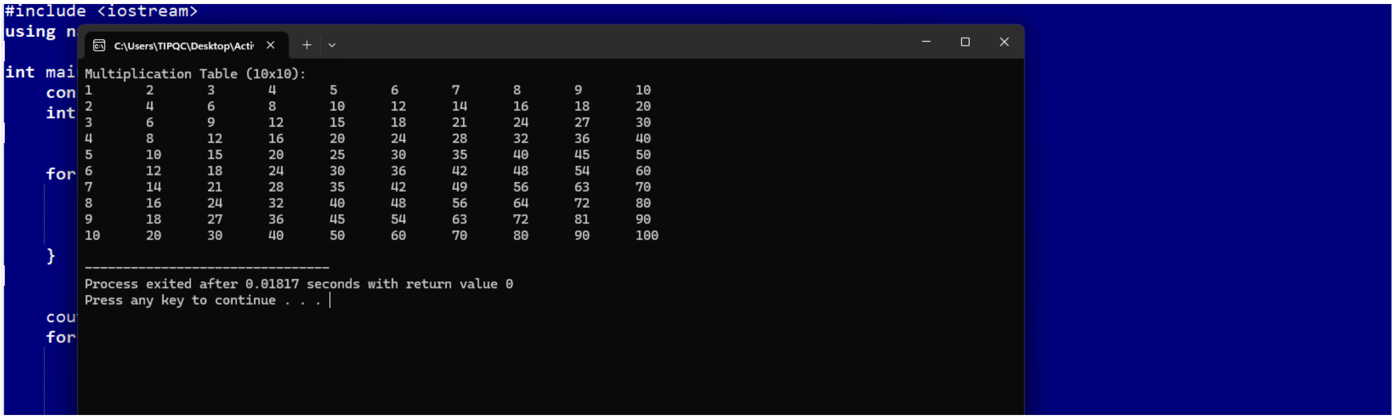
6. Output

INPUT:


C:\Users\TIPQC\Desktop\Activity 5.1Multiplicaiontable.cpp - Dev-C++ 5.10
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.8.1 64-bit Release
(globals)
Project Classes Debug
Activity 5.1Multiplicaiontable.cpp

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

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


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


    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;
}
```

OUTPUT:

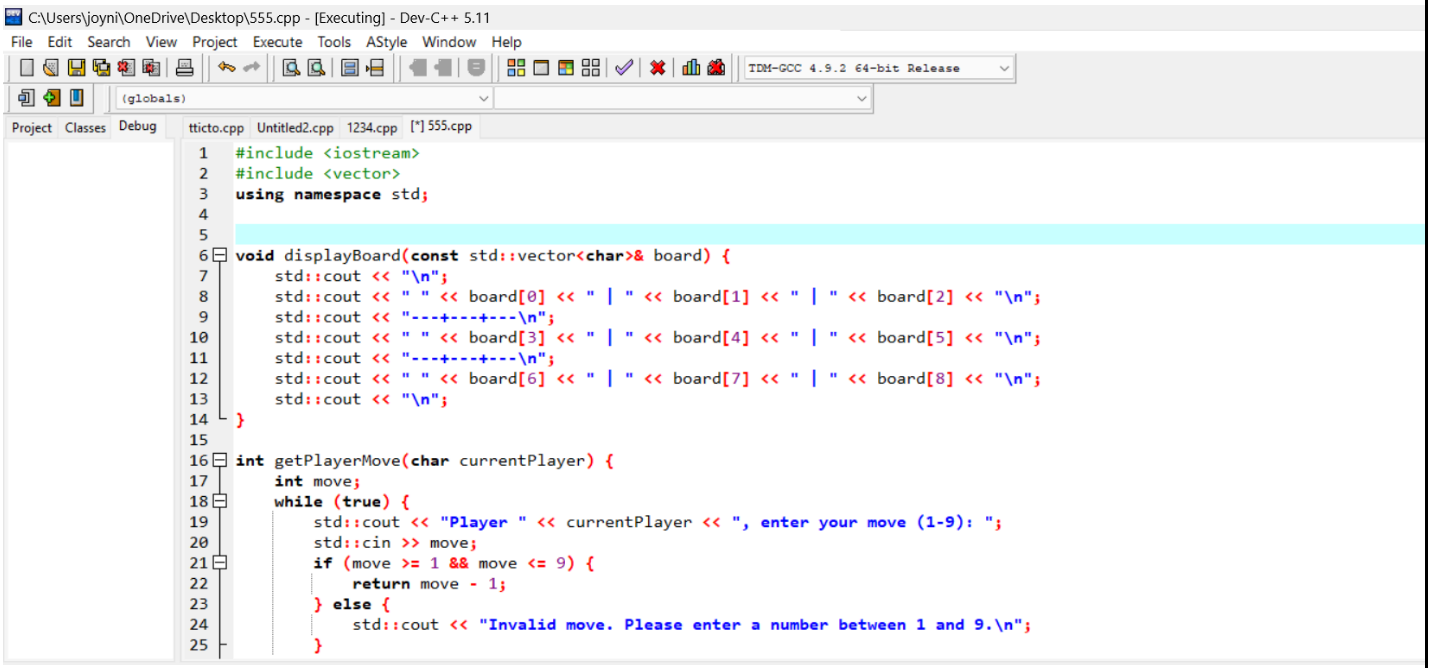


```
#include <iostream>
using n

int mai
    con
    int

    for

    }

    cou
    for
```

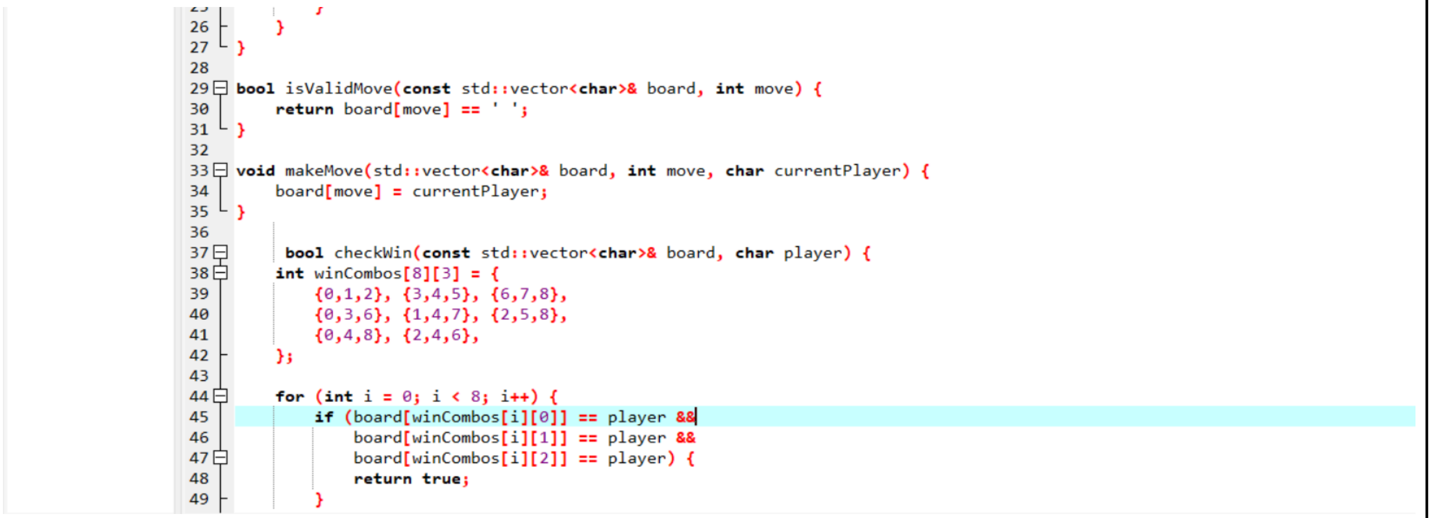
```
Multiplication Table (10x10):
1       2       3       4       5       6       7       8       9       10
2       4       6       8       10      12      14      16      18      20
3       6       9       12      15      18      21      24      27      30
4       8       12      16      20      24      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 0.01817 seconds with return value 0
Press any key to continue . . .
```

To run this code I used multidimensional arrays.This program stores and displays a multiplication table using a two-dimensional array. The array table[SIZE][SIZE] is perfect for representing a grid because it has two indexes: one for rows and one for columns. The array is filled by a nested for loop, in which each element is given the product of (i+1) and (j+1), causing the table to begin at 1 rather than 0. A second nested loop that uses tab characters to maintain column alignment prints the values in a grid format after the array has been filled. The complete multiplication table is displayed on the screen after the program has been compiled using a C++ compiler like g++ and run from the terminal or command prompt.

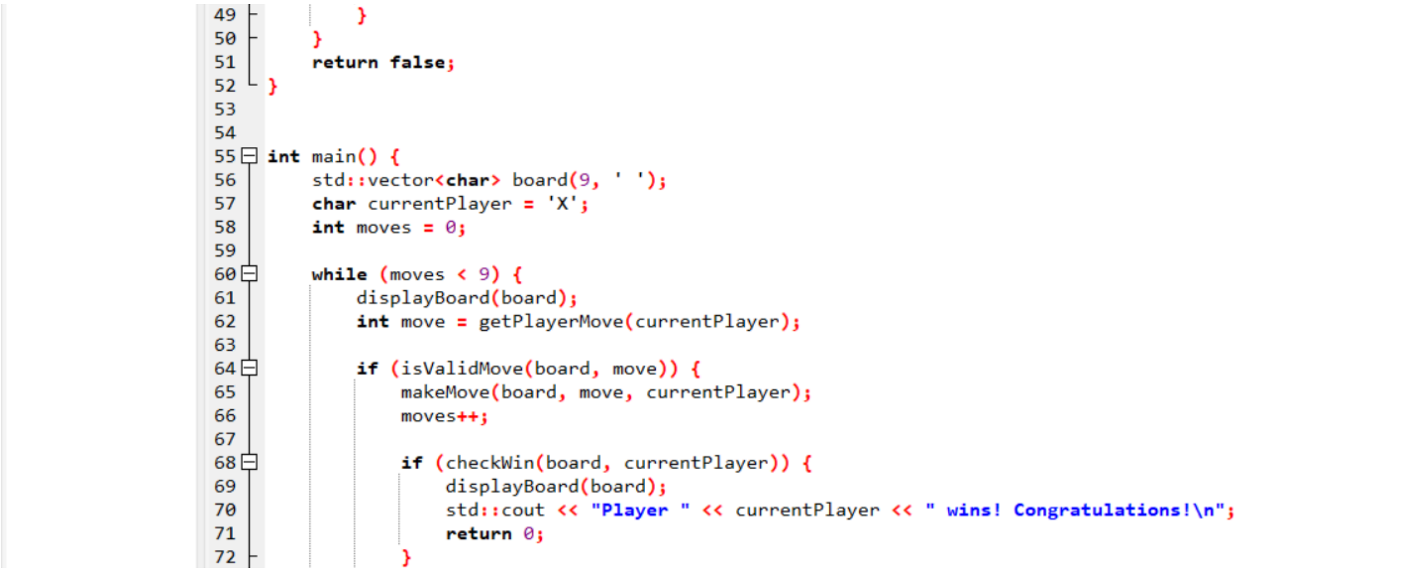
INPUT:


C:\Users\joyni\OneDrive\Desktop\555.cpp - [Executing] - Dev-C++ 5.11
File Edit Search View Project Execute Tools AStyle Window Help
TDM-GCC 4.9.2 64-bit Release
(globals)
Project Classes Debug
tticto.cpp Untitled2.cpp 1234.cpp [*] 555.cpp

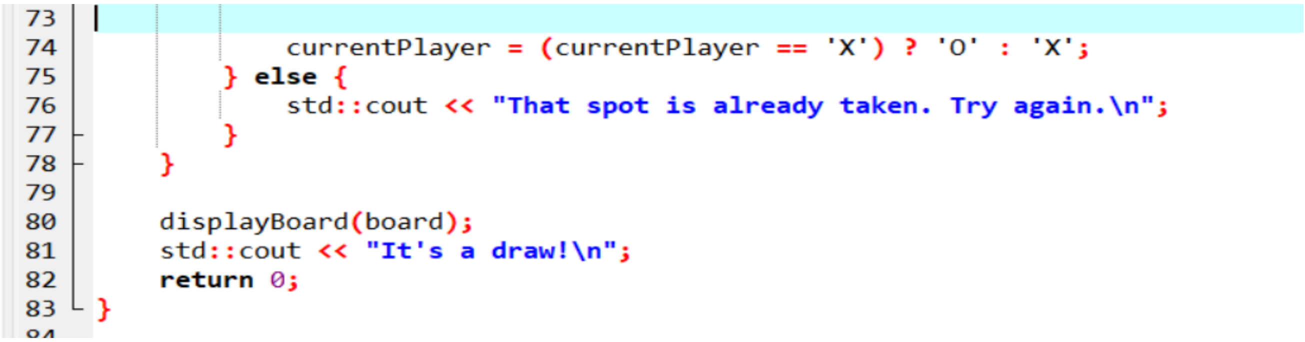
```cpp
#include <iostream>
#include <vector>
using namespace std;


void displayBoard(const std::vector<char>& board) {
    std::cout << "\n";
    std::cout << " " << board[0] << " | " << board[1] << " | " << board[2] << "\n";
    std::cout << "---+---+---\n";
    std::cout << " " << board[3] << " | " << board[4] << " | " << board[5] << "\n";
    std::cout << "---+---+---\n";
    std::cout << " " << board[6] << " | " << board[7] << " | " << board[8] << "\n";
    std::cout << "\n";
}

int getPlayerMove(char currentPlayer) {
    int move;
    while (true) {
        std::cout << "Player " << currentPlayer << ", enter your move (1-9): ";
        std::cin >> move;
        if (move >= 1 && move <= 9) {
            return move - 1;
        } else {
            std::cout << "Invalid move. Please enter a number between 1 and 9.\n";
        }
```



```cpp
    }
}

bool isValidMove(const std::vector<char>& board, int move) {
    return board[move] == ' ';
}

void makeMove(std::vector<char>& board, int move, char currentPlayer) {
    board[move] = currentPlayer;
}

     bool checkWin(const std::vector<char>& board, char player) {
    int winCombos[8][3] = {
        {0,1,2}, {3,4,5}, {6,7,8},
        {0,3,6}, {1,4,7}, {2,5,8},
        {0,4,8}, {2,4,6},
    };

    for (int i = 0; i < 8; i++) {
        if (board[winCombos[i][0]] == player &&
            board[winCombos[i][1]] == player &&
            board[winCombos[i][2]] == player) {
            return true;
        }
```

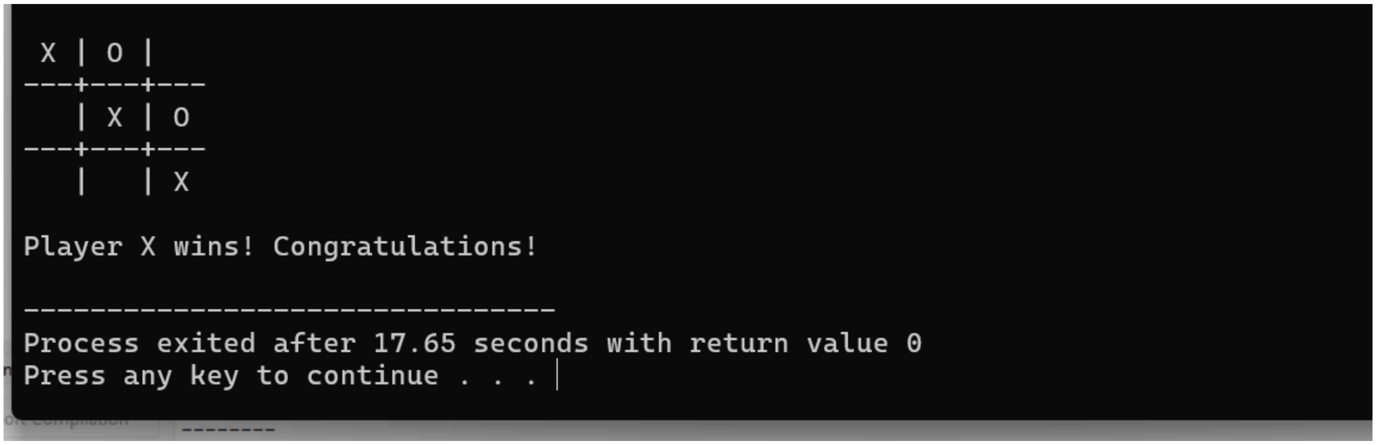


```
        }
    }
    return false;
}


int main() {
    std::vector<char> board(9, ' ');
    char currentPlayer = 'X';
    int moves = 0;

    while (moves < 9) {
        displayBoard(board);
        int move = getPlayerMove(currentPlayer);

        if (isValidMove(board, move)) {
            makeMove(board, move, currentPlayer);
            moves++;

            if (checkWin(board, currentPlayer)) {
                displayBoard(board);
                std::cout << "Player " << currentPlayer << " wins! Congratulations!\n";
                return 0;
            }
```



```

            currentPlayer = (currentPlayer == 'X') ? 'O' : 'X';
        } else {
            std::cout << "That spot is already taken. Try again.\n";
        }
    }

    displayBoard(board);
    std::cout << "It's a draw!\n";
    return 0;
}
```

OUTPUT



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

Player X wins! Congratulations!

--------------------------------
Process exited after 17.65 seconds with return value 0
Press any key to continue . . .
```

To sum up, this C++ application offers a full and working implementation of the game Tic-Tac-Toe. Using a `std::vector` to represent the 3x3 grid, a `getPlayerMove` function to handle player input, and `isValidMove` and `makeMove` to enforce game rules, it effectively handles the core game logic. The `checkWin` function, the game's central component, iteratively goes through each of the eight potential winning combinations to accurately detect a victory. The game is finally coordinated by the main loop, which also controls turns, looks for wins after each move, and appropriately ends the game with a **player victory or a draw*when the board is full.

7. Supplementary Activity

8. Conclusion

A block of memory with slots that each contain a single element of the same type is called an array. For instance, you can declare an array of size five if you wish to store five integers. You can use indices to access the integers that will be stored in each slot. A one-dimensional array functions similarly to a row of sequentially numbered lockers.The representation is more straightforward for a two-dimensional array. After declaring char board[3][3];, you immediately arrange markers by row and column, for example, board[1][2] = 'X';. Since the data is already stored in a table-like structure, there is no need to convert a one-dimensional index into a two-dimensional position. This reduces the likelihood of errors and makes the program easier to read and write.