```cpp
// Khamille A. Sarmiento
// CS 265 - C++ Programming
// Prof. Raheja
// Project 3.2: Tic Tac Toe Game

#include <iostream>
#include <cstdlib>
#include <string>
#include "TicTacToe.h"
using namespace std;

// Default Constructor.
// Creates a board of size 3x3 and initializes each space
// to have stars (*).
TicTacToe::TicTacToe() {
    board = new char *[SIZE]; // creates an array of size three
    for(int i = 0; i < SIZE; i++) {
        *(board + i) = new char[SIZE];
    } // puts an array of size three into each of the three other arrays.
    // ~multidimensional maddness~

    for(int x = 0; x < SIZE; x++) { // initializes the elements to be stars (*)
        for(int y = 0; y < SIZE; y++) {
            *(*(board + x) + y) = '*';
        }
    }
}

// Copy Constructor.
// Copies the elements from another obj's board to this coard.
TicTacToe::TicTacToe(const TicTacToe& obj) {
    //SIZE = obj.SIZE;
    board = new char *[obj.SIZE];
    for(int i = 0; i < SIZE; i++) { // creates a blank new slate to copy to
        *(board+i) = new char[obj.SIZE];
    }
    for(int i = 0; i < SIZE; i++) { // copies elements from obj to this board.
        for(int j = 0; j < SIZE; j++) {
            *(*(board+i)+j) = *(*(obj.board+i)+j);
        }
    }
}

// Destructor.
// Deallocates memory to be used elsewhere.
TicTacToe::~TicTacToe() {
    for(int i = 0; i < SIZE; i++) {
        delete[] *(board+i);
    }
    delete[] board;
}

// Set Function.
// Sets a player's mark at a specified location.
// Parameters: player's character, desired row, desired column.
void TicTacToe::set(char s, int r, int c) {
    *(*(board+r)+c) = s;
}

// Print Function.
// Uses cout to print the current state of the board.
void TicTacToe::print() {
    cout << "C  o   l   s" << endl;
    cout << "R  1   2   3" << endl;
    cout << endl;
    for(int i = 0; i < SIZE; i++) {
```

```
        cout << (i+1) << "  ";
        for(int j = 0; j < SIZE; j++) {
            if(j == 2) {
                if(*(*(board+i)+j) == 'x') {
                    cout << "X";
                } else if(*(*(board+i)+j) == 'o') {
                    cout << "O";
                } else {
                    cout << " ";
                }
                cout << endl;
            } else {
                if(*(*(board+i)+j) == 'x') {
                    cout << "X";
                } else if(*(*(board+i)+j) == 'o') {
                    cout << "O";
                } else {
                    cout << " ";
                }
                cout << " | ";
            }
        } if(i == 2) {
            cout << endl;
        } else {
            cout << "   ---------" << endl;
        }
    }
}

// "Is Taken" Function.
// Checks to see if a specified spot on the board is already
// marked or not.
bool TicTacToe::isMarked(int r, int c) {
    if(*(*(board+r)+c) == '*') {
        return false;
    }
    else {
        return true;
    }
}

// "Is Filled" Function.
// Returns true when the board has no more enpty spaces.
bool TicTacToe::isFilled() {
    for (int i = 0; i < SIZE; i++) {
        for (int j = 0; j < SIZE; j++) {
            if (*(*(board+i)+j) == '*') {
                return false;
            }
        }
    }
    return true;
}

// Winner Function.
// Checks all possible winning combinations
bool TicTacToe::isWinner() {
    // Save the current state of the board by individual characaters
    char r0c0 = *(*(board+0)+0);
    char r0c1 = *(*(board+0)+1);
    char r0c2 = *(*(board+0)+2);
    char r1c0 = *(*(board+1)+0);
    char r1c1 = *(*(board+1)+1);
    char r1c2 = *(*(board+1)+2);
    char r2c0 = *(*(board+2)+0);
    char r2c1 = *(*(board+2)+1);
```

```
    char r2c2 = *(*(board+2)+2);

    // All possible winning combinations:
    if(r0c0 != '*' && r0c0 == r0c1 && r0c1 == r0c2) { // first row winner
        return true;
    } else if(r1c0 != '*' && r1c0 == r1c1 && r1c1 == r1c2) { // second row winner
        return true;
    } else if(r2c0 != '*' && r2c0 == r2c1 && r2c1 == r2c2) { // third row winner
        return true;
    } else if(r0c0 != '*' && r0c0 == r1c0 && r1c0 == r2c0) { // first column winner
        return true;
    } else if(r0c1 != '*' && r0c1 == r1c1 && r1c1 == r2c1) { // second column winner
        return true;
    } else if(r0c2 != '*' && r0c2 == r1c2 && r1c2 == r2c2) { // third column winner
        return true;
    } else if(r2c0 != '*' && r2c0 == r1c1 && r1c1 == r0c2) { // right diagonal winner
        return true;
    } else if(r0c0 != '*' && r0c0 == r1c1 && r1c1 == r2c2) { // left diagonal winner
        return true;
    } else {
        return false;
    }
}
```