```c
// TacTacToe solution checker by Chip Henderson for SMU CS7343


#include <pthread.h> /* Need to reneable this when compiling for linux, disable for Windows */
#include <stdio.h>
#include <stdlib.h>
// #define num_threads == 7 /* Stop gap solution until multithread version is ready */
int num_threads = 7;

char gameBoard[9];
// int solutionArray[num_threads]
int solutionArray[8];
// Note: solution array consists of row, row, row, column, column, column, diag, diag
char winner;

/* structure for passing data to threads. This needs to be reenabled for Linux */

typedef struct {
    int row;
    int column;
} parameters;

void intro()
{
    printf("Welcome to Tac-Tac-toe\n");
    printf("Enter a game board such as XOOOXOOOX\n");

    gets(gameBoard);
    printf( "\nYou entered: %s", gameBoard);


}

void rowCheck ()
{
    int i = 0;
    int j;

    for (j = 0; j < 10; j+=3)
    {
        if (gameBoard[j] == gameBoard[j + 1] && gameBoard[j + 1] == gameBoard[j + 3]) {
            solutionArray[i] = gameBoard[j];
        }
            else {
                solutionArray[i] = 0;
            }
        i++;
    }
}

void columnCheck()
{
    int i = 3;
    int j;

    for (j = 0; j < 3; j++)
```

```
    {
        if (gameBoard[j] == gameBoard[j + 3] && gameBoard[j + 3] == gameBoard[j+6])
{
            solutionArray[i] = gameBoard[j];
        }
            else {
                solutionArray[i] = 0;
            }
        i++;
    }
}

void diagCheck()
{
/*Note: I think it may be possible to get by with only one diagonal result. Will
check later.*/
    int i;

    // for (i = 6; i < 8; i++)
    // {
        if (gameBoard[0] == gameBoard[4] && gameBoard[4] == gameBoard[8]) {
            solutionArray[6] = gameBoard[4];
        }
            else if (gameBoard[2] == gameBoard[4] && gameBoard[4] == gameBoard[6])
{
                solutionArray[6] = gameBoard[4];
            }
            else {
                solutionArray[6] = 0;
            }
    // }

}

int main()
{
    intro();
    rowCheck();
    columnCheck();
    diagCheck();

    int i;
    int solutionSum = 0;
    for (i = 0; i < num_threads; i++) {
        solutionSum += solutionArray[i];
    }

    if (solutionSum == 0) {
        printf("\nThere is no winner!\n");
    }
    else if (solutionSum % 88 == 0) {
        printf("\nWinner is X!\n");
    }
    else printf("\nWinner is O!\n");

    return 0;
}
// parameters *data = (parameters *) malloc(sizeof(parameters));

// data->row = 1;
```

```
113
114 // data->column = 1;
115
116
117
118 // /* Now create the thread passing it data as a parameter */
```