**EEE101 C programming and SW engineering 1**

This is the report for Assignment 2 of EEE101.

1. **Problem Statement:**

The assignment asks to design a C program that allow two players 1 and 2 to play the game of tic-tac-toe. In the beginning, computer will decide who play first randomly. Then players take turn to take place using ‘O’ or ‘X’ on the board by choosing the number 1-9. The game is won if someone can place three of their tokens in a row, horizontally, vertically or diagonally. The game will be a tie if all the squares are filled and nobody will.

1. **Analysis:**

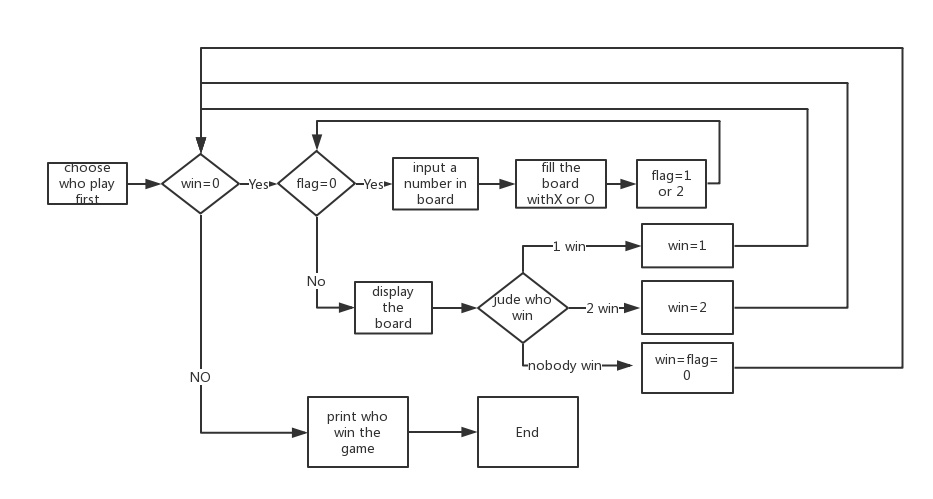
Input:

An integer from 1-9 from user to represent the number in board.

Outputs:

1. Print “Here is a blank Tic-Tac-Toe board for you and a friend to play Tic-Tac-Toe on” to begin the game.
2. Tell user who will play first
3. Display the current board
4. After someone input, tell the other to input a number
5. If someone make invalid input tell the wrong
6. When someone will the game or the game become a tie. Tell the result to the player

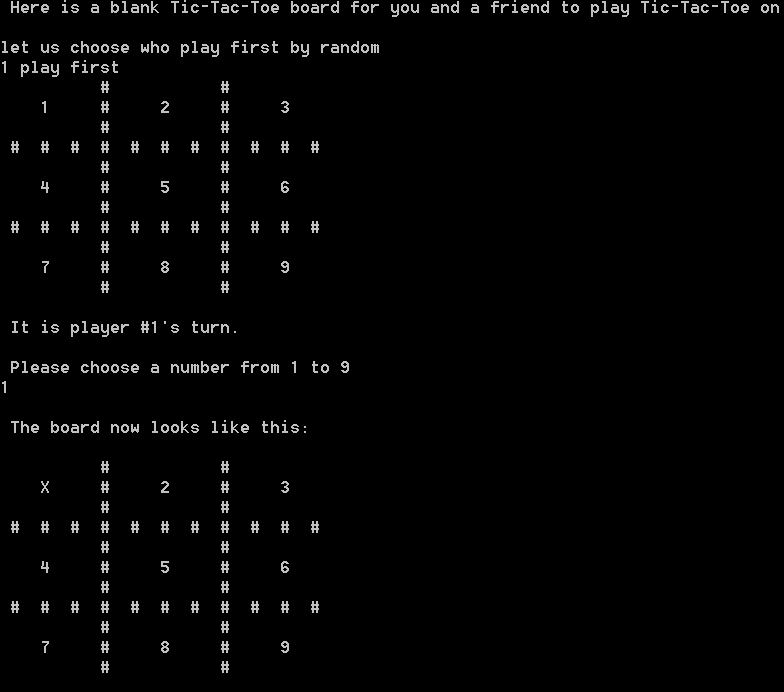
Constraints:

1. The choose who play first by random, so it should include standard library time.h and use function srand.
2. When players input number, they may input a number over the board, so it should make sure the number is greater than or equal to 1 is less than or equal to 9.
3. When players input number, they may input some character, so the program should make sure the invalid input not disturb the program.
4. After the player has completed a step, the program must ensure that two people exchange chess order.
5. When two pieces fall on the same place, make sure the program recognizes and reports。
6. When all the blank are full and nobody wins, the program should recognize the game is a tie
7. **Design:**
8. Declare a function named display\_board. This function has an array input the type of char. It can display the character in array on the screen
9. Declare a function named winner. This function has an array input the type of char. It uses ‘winner\_numer’ as return number and can judge who win the game
10. Tell players 1 and 2 who will play first and the chosen people play first
11. While nobody win, The player input the number and place the number in board with ‘O’ or ‘X’
12. Each time the player enters the number, the program detects if someone wins. If nobody wins, and there are still blanks then reset flag and exchange the order of the chess-playing. If there is no blank, the game will be a tie。If some win print ‘the people win’
13. **For implementation please see the C code in file “Assignment 2-1.c” with comments.**
14. **Testing:**

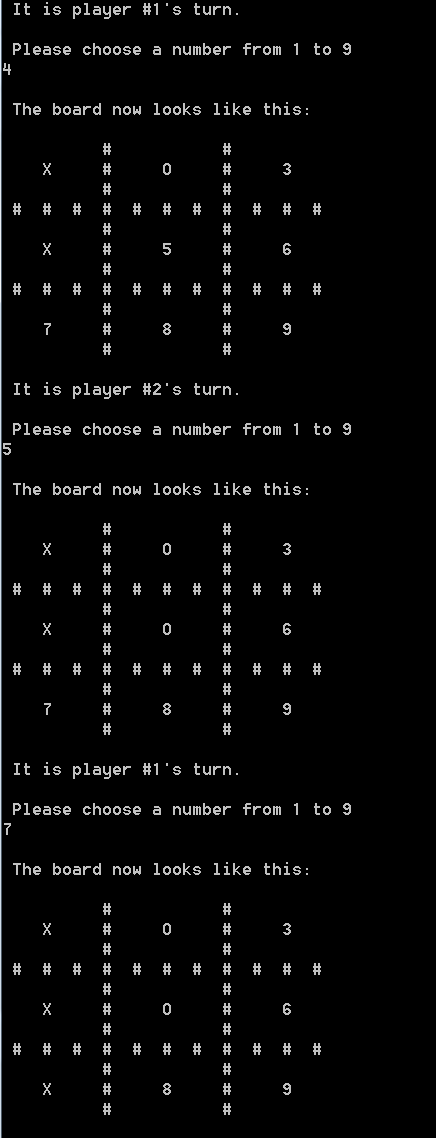
The C program has been test with several experiments and the outputs have been verified successfully. For instance,

Test 1(X win):

Input some numbers to make players who take X win the game

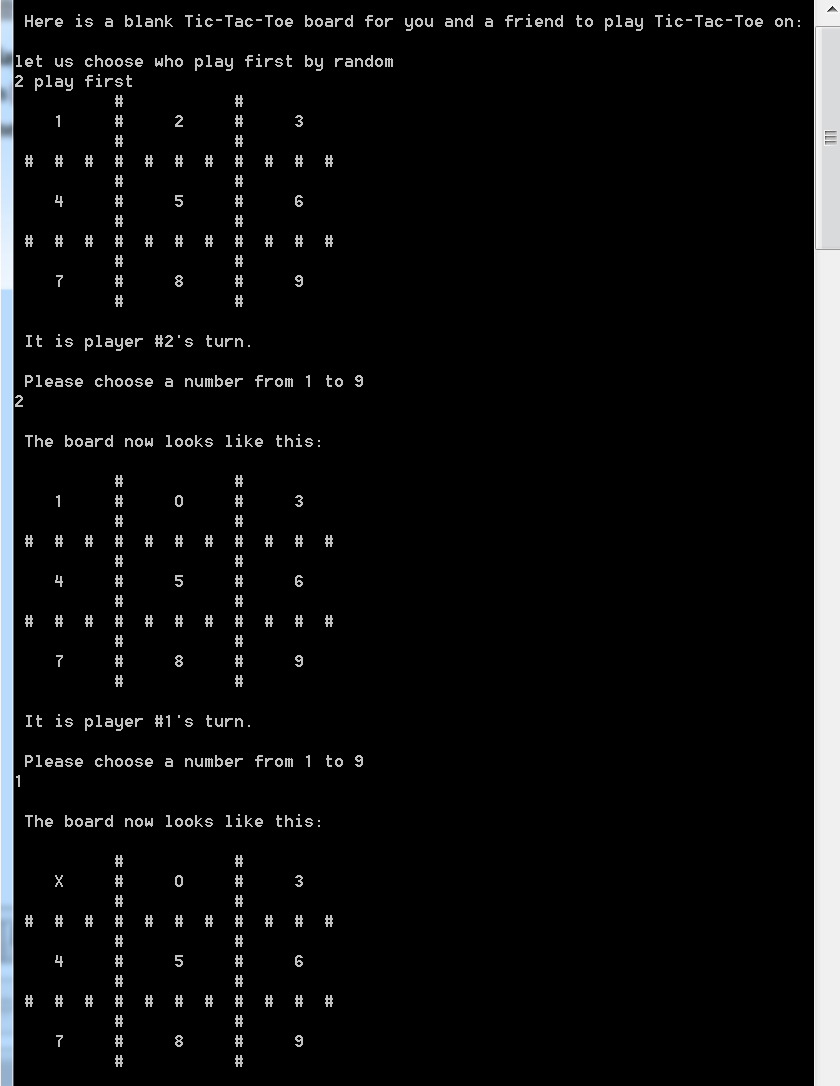


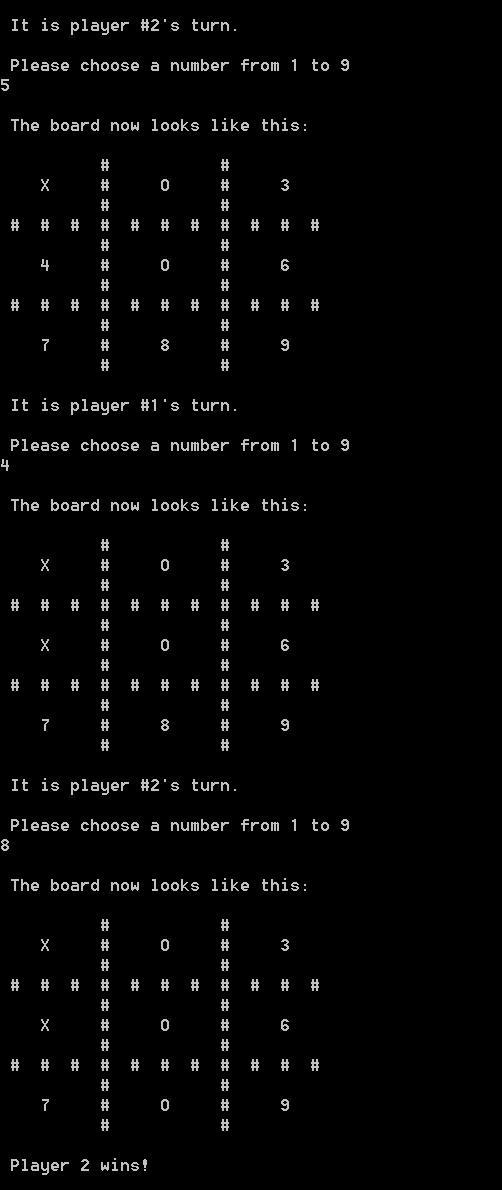




Test 2(O win):

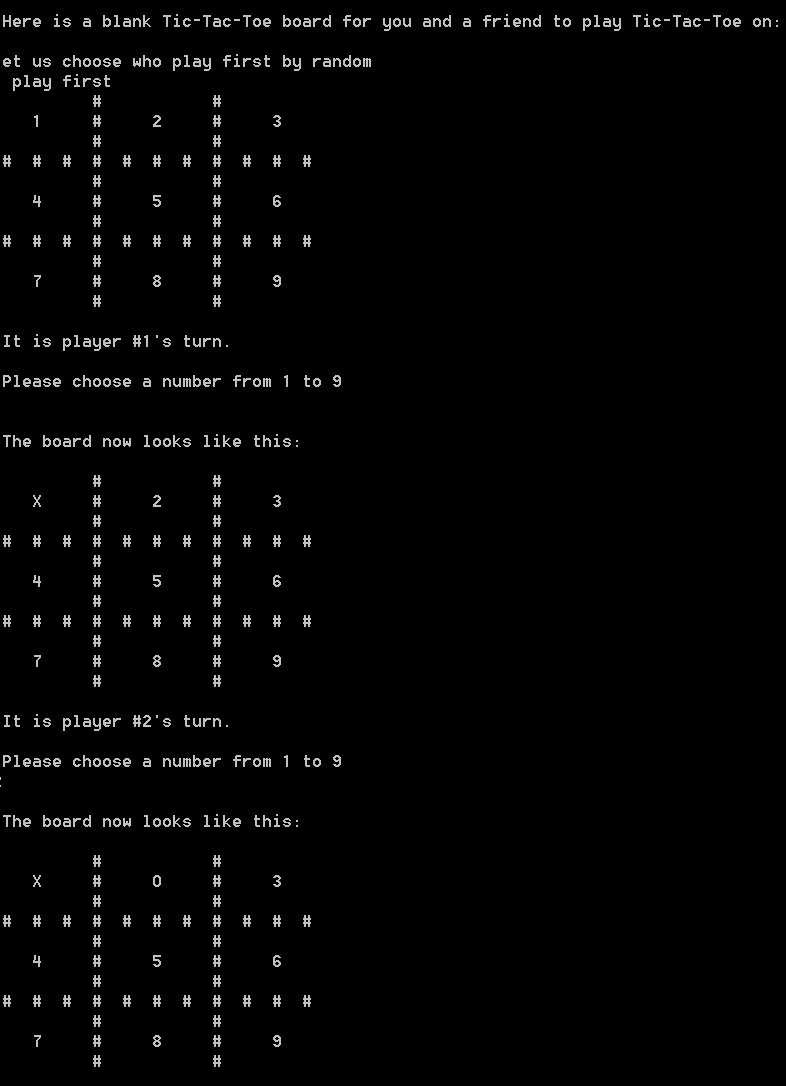
Input some numbers to make players who take O win the game

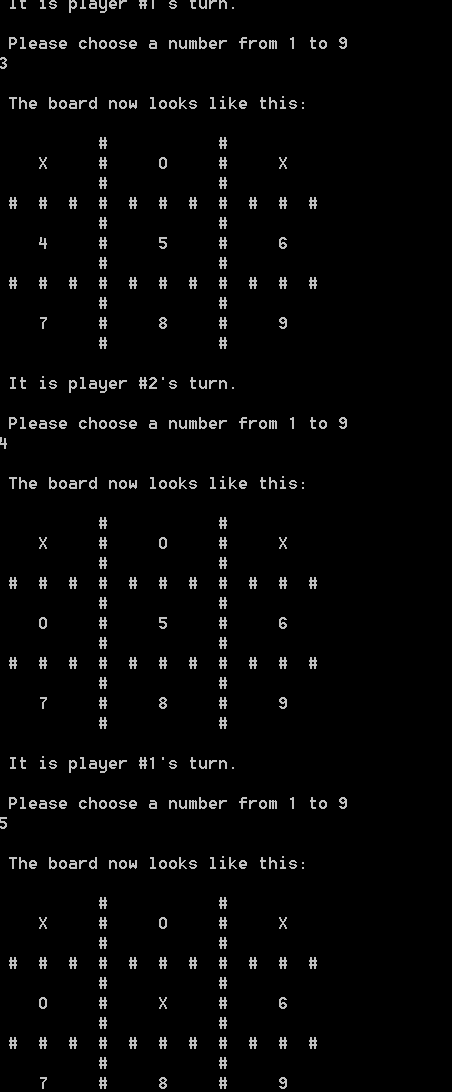


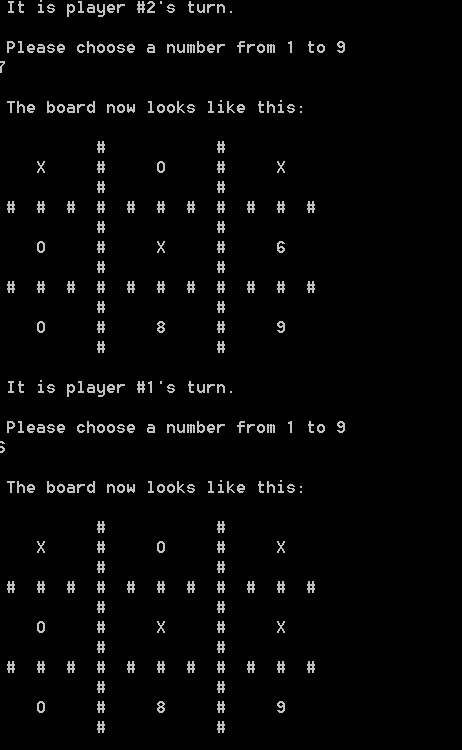


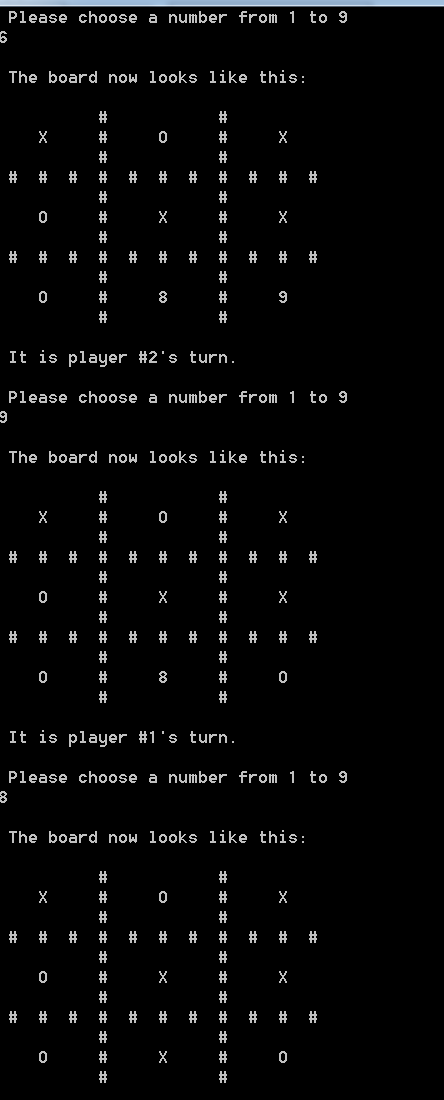
Test 3(Nobody win):

Input some numbers to make nobody win the game





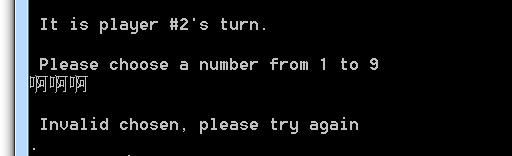




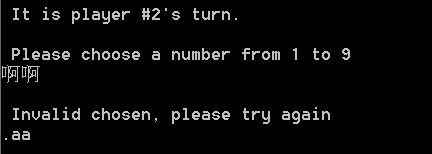


1. **Robustness Test:**

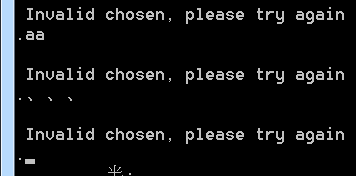
Robustness test 1(enter a Chinese word “啊啊啊” while choosing a number):



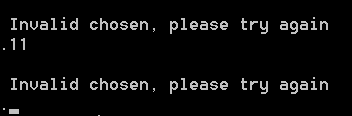
Robustness test 2(enter Chinese word “aa” while choosing a number after typing ‘啊啊啊’):



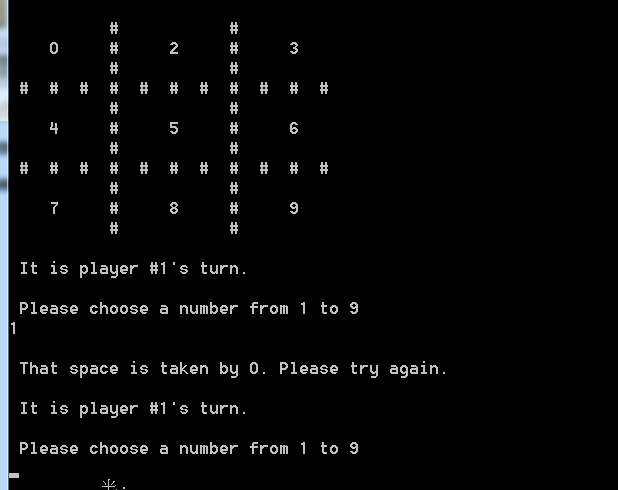
Robustness test 3(enter Chinese word “、、” while choosing a number after typing ‘aa’):



Robustness test 4(enter a number over the board):



Robustness test 5(enter the same number)



1. **C code:**

/\*

Name: Simple Program for Name,Student ID number and Height

File name: 1613453\_2.c

copyright: Free

Author: Zhong Chu ID number: 1613453.

Description: A Programm allow two players to play the game of tic-tac-toe\*/

#include <stdio.h> /\* include information about standard library stdio.h\*/

#include <stdlib.h> /\* include information about standard library stdlib.h\*/

#include <string.h> /\* include information about standard library string.h\*/

#include <time.h> /\* include information about standard library time.h\*/

int winner(char board[11][11]); /\* Define a main function named winner which could judge who is winner\*/

int display\_board(char board[11][11]);/\* Define a main function named winner which could display the board\*/

int main() /\* Define a main function named main\*/

{

int a; /\* Initialize variable which is used to return the value of scanf\*/

int number = 0; /\* Initialize variable which is used to collect number\*/

int player = 1; /\* Represent players as numbers\*/

int flag = 0; /\* Used later in a while-loop\*/

int win = 0; /\* Initialize win condition flag\*/

int row=0;

int column=0;

char board[11][11]=

{

{' ',' ',' ','#',' ',' ',' ','#',' ',' ',' '} ,

{' ','1',' ','#',' ','2',' ','#',' ','3',' '} ,

{' ',' ',' ','#',' ',' ',' ','#',' ',' ',' '} ,

{'#','#','#','#','#','#','#','#','#','#','#'},

{' ',' ',' ','#',' ',' ',' ','#',' ',' ',' '} ,

{' ','4',' ','#',' ','5',' ','#',' ','6',' '} ,

{' ',' ',' ','#',' ',' ',' ','#',' ',' ',' '} ,

{'#','#','#','#','#','#','#','#','#','#','#'},

{' ',' ',' ','#',' ',' ',' ','#',' ',' ',' '} ,

{' ','7',' ','#',' ','8',' ','#',' ','9',' '} ,

{' ',' ',' ','#',' ',' ',' ','#',' ',' ',' '} ,

}; /\* Initialize the board\*/

printf("\n Here is a blank Tic-Tac-Toe board for you and a friend to play Tic-Tac-Toe on: \n");

printf("let us choose who play first by random\n");

srand(time(0));

player=rand()%(2)+1;/\*choose who play first by random\*/

printf("%d play first",player);

display\_board(board); /\* Show initial position of board \*/

while (win == 0) /\* Continue the game until a player wins\*/

{

while (flag == 0)

{

printf("\n It is player #%i's turn. \n", player); /\* Statement for begining\*/

printf("\n Please choose a number from 1 to 9\n");

a = scanf("%d", &number);

fflush(stdin); /\*remove the data in the input buffer\*/

while (a == 0)

{

printf("\n Invalid chosen, please try again\n.");

a=scanf("%d", &number);

fflush(stdin);/\*remove the data in the input buffer\*/

}

while(number!=9&&number!=8&&number!=7&&number!=6&&number!=5&&number!=4&&number!=3&&number!=2&&number!=1)/\*make sure the number is on board\*/

{

printf("\n Invalid chosen, please try again\n.");

scanf("%d",&number);

fflush(stdin);/\*remove the data in the input buffer\*/

}

if (number==1){

row=1;

column=1;}

else if(number==2){

row=1;

column=5;}

else if(number==3){

row=1;

column=9;}

else if(number==4){

row=5;

column=1;}

else if(number==5){

row=5;

column=5;}

else if(number==6){

row=5;

column=9;}

else if(number==7){

row=9;

column=1;}

else if(number==8){

row=9;

column=5;}

else if(number==9){

row=9;

column=9;}

else

printf("\n Invalid chosen, please try again.");

if (player == 1 && board[row][column] <= '9'&& board[row][column] >= '1') {

board[row][column] = 'X';

flag = 1;

} else if (player == 2 && board[row][column] <= '9'&& board[row][column] >= '1') {

board[row][column] = 'O';

flag = 2;

} else if (board[row][column] == 'O') {

printf("\n That space is taken by O. Please try again. \n");

} else if (board[row][column] == 'X') {

printf("\n That space is taken by X. Please try again. \n");

} else {

printf("This isn't supposed to happen"); /\* Just in case\*/

exit(1); /\* throw an error to get out of the loop\*/

}

number = 0; /\* Reset\*/

}

printf("\n The board now looks like this: \n");

display\_board(board); /\* Update board\*/

win = winner(board); // Check for winner

if (win == 1) { // Function winner() should return 1 if player 1 wins

printf("\n Player 1 wins! \n");

} else if (win == 2) { // Function winner() should return 2 if player 2 wins

printf("\n Player 2 wins! \n");

} else if (win == 3) { // Function winner() should return 3 if it's a tie.

printf("\n It's a tie. \n");

} else {

flag = 0; /\* Reset\*/

win = 0; /\* Reset\*/

}

if (player == 1) { /\* Switch players (with below)\*/

player = 2; }

else if (player == 2) {

player = 1;}

else {

printf("This isn't supposed to happen"); /\* Just in case\*/

}

}

printf(" \n ");

system("pause");

return 0;

}

int display\_board(char board[11][11])/\*Placeholder for display\_board() functio\*/

{

int i = 0; /\* initialize loop counter\*/

int j = 0; /\* initialize loop counter\*/

for (i = 0; i < 11; i++)

{

printf("\n"); /\* Each row should end in a newline\*/

for (j = 0; j < 11; j++) /\* Go through each column\*/

{

printf(" %c ", board[i][j]); /\* Not just a statement for debugging this time around.\*/

}

}

printf(" \n "); /\* just some extra whitespace for no reason\*/

return 0;

}

int winner(char board[11][11]) /\* Placeholder for winner() function \*/

{

int winner\_number = 0; /\*Returned to main function later\*/

int blank = 0; /\* Used to tell if the game is over or there's a tie\*/

int i = 0; /\* Initialize loop-counter (used to go through board to detect blank spaces)\*/

int j = 0; /\* Initialize loop-counter (used to go through board to detect blank spaces)\*/

if (((board[1][1] == 'X') && (board[1][5] == 'X') && (board[1][9] == 'X')) || /\* first row\*/

((board[1][1] == 'X') && (board[5][1] == 'X') && (board[9][1] == 'X')) || /\* first column\*/

((board[1][1] == 'X') && (board[5][5] == 'X') && (board[9][9] == 'X')) || /\* diagonal 1\*/

((board[5][1] == 'X') && (board[5][5] == 'X') && (board[5][9] == 'X')) || /\* second row\*/

((board[9][1] == 'X') && (board[9][5] == 'X') && (board[9][9] == 'X')) || /\* third row\*/

((board[1][5] == 'X') && (board[5][5] == 'X') && (board[9][5] == 'X')) || /\* second column\*/

((board[1][9] == 'X') && (board[5][9] == 'X') && (board[9][9] == 'X')) || /\* third column\*/

((board[1][9] == 'X') && (board[5][5] == 'X') && (board[9][1] == 'X')) /\* diagonal 2\*/

)

{

winner\_number = 1;

} else if (((board[1][1] == 'O') && (board[1][5] == 'O') && (board[1][9] == 'O')) || /\* first row\*/

((board[1][1] == 'O') && (board[5][1] == 'O') && (board[9][1] == 'O')) || /\* first column\*/

((board[1][1] == 'O') && (board[5][5] == 'O') && (board[9][9] == 'O')) || /\* diagonal 1\*/

((board[5][1] == 'O') && (board[5][5] == 'O') && (board[5][9] == 'O')) || /\* second row\*/

((board[9][1] == 'O') && (board[5][9] == 'O') && (board[9][9] == 'O')) || /\* third row\*/

((board[1][5] == 'O') && (board[5][5] == 'O') && (board[9][5] == 'O')) || /\* second column\*/

((board[1][9] == 'O') && (board[5][9] == 'O') && (board[9][9] == 'O')) || /\* third column\*/

((board[1][9] == 'O') && (board[5][5] == 'O') && (board[9][1] == 'O')) /\* diagonal 2\*/

)

{

winner\_number = 2;

} else

{

for (i = 1; i < 10; i=i+4) /\* Go through each row\*/

{

for (j = 1; j < 10; j=j+4) /\* Go through each column\*/

{

if (board[i][j] <= '9'&&board[i][j]>='1') {

blank = 1; /\* 1 means there's still a blank\*/

}

}

}

if (blank == 0) { /\* It will still be 0 if no blanks were found\*/

winner\_number = 3;

}

}

}

return winner\_number;