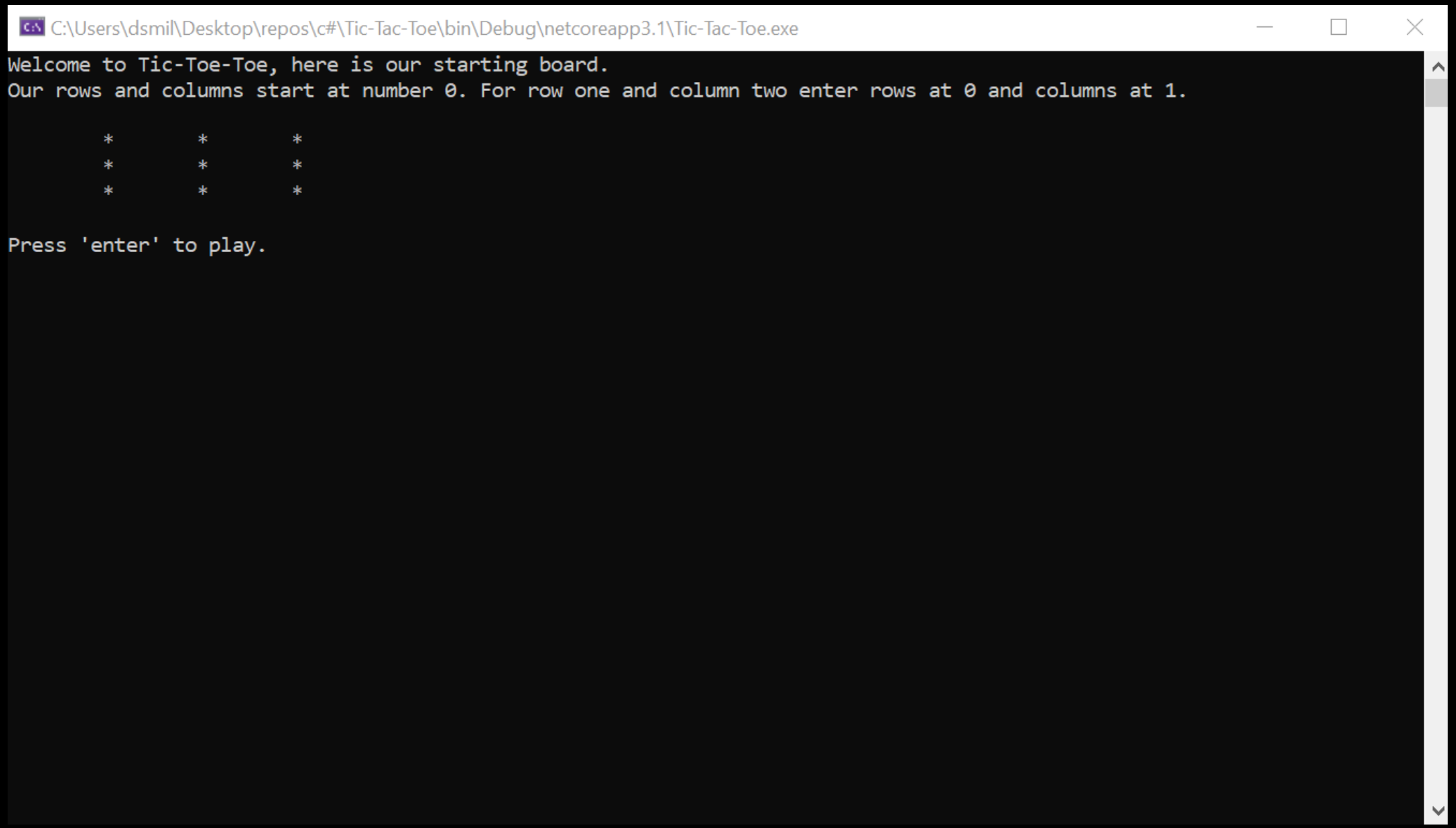
TIC-TAC-TOE

DUSTIN MILLER

Game Aspects

- Displays our board game.
- Allow player 1 & 2 to select a location on the board by entering in a row and column number.
- Determine whether a player has won or if a tie has occurred.


C:\Users\dsmil\Desktop\repos\c#\Tic-Tac-Toe\bin\Debug\netcoreapp3.1\Tic-Tac-Toe.exe
Welcome to Tic-Toe-Toe, here is our starting board.
Our rows and columns start at number 0. For row one and column two enter rows at 0 and columns at 1.
* * *
* * *
* * *
Press 'enter' to play.

Main

```
7   static void Main(string[] args) {
8       //declare variables for our game marker position
9       int x_coord = 0;
10      int y_coord = 0;
11
12      //create 2d array for tic-tac-toe board
13      string[,] gameBoard = new string[3, 3] { {"*","*","*"},
14                                               {"*","*","*"},
15                                               {"*","*","*"} };
16
17      //function calls
18      DrawBoard(gameBoard);
19      GameLoop(gameBoard, x_coord, y_coord);
20  }//end main
21
```

- Driver function that contains our game board and marker coordinates.
- Calls our functions to run through our game.

Display our game board

```
22      static void DrawBoard(string[,] gameBoard)
23      {
24          //get length of the rows. grabs the elements in the first dimension of our array.
25          var rows = gameBoard.GetLength(0);
26
27          //get length of columns. grabs the elements in the second dimension of our array.
28          var columns = gameBoard.GetLength(1);
29
30          //greet our user
31          Console.WriteLine("Welcome to Tic-Toe-Toe, here is our starting board. ");
32          Console.WriteLine("Our rows and columns start at number 0. For row one and column two enter rows at 0 and columns at 1. ");
33          Console.WriteLine();
34
```

```
35              // for loop to run though 2d array (rows & columns)
36              for (int i = 0; i < rows; i++) {
37                  for (int j = 0; j < columns; j++) {
38                      //writes our rows and columns with a tab inbetween *
39                      Console.Write("\t" + gameBoard[i, j]);
40
41                  }//end inside for loop
42
43                  //writes new line for row 1 and so on so the board is in a grid and not on one line
44                  Console.WriteLine();
45              }//end outside for loop
46
47              //ask user if they would like to play
48              Console.WriteLine("\nPress 'enter' to play. ");
49              Console.ReadLine();
50          }//end function
```

- Prompts user with welcome message.
- For loop to traverse through our 2d array.

C:\Users\dsmil\Desktop\repos\c#\Tic-Tac-Toe\bin\Debug\netcoreapp3.1\Tic-Tac-Toe.exe

```
Welcome to Tic-Toe-Toe, here is our starting board.
Our rows and columns start at number 0. For row one and column two enter rows at 0 and columns at 1.

        *       *       *
        *       *       *
        *       *       *

Press 'enter' to play.


Player 1 please input the row you would like to place your 'X' at : 0
Player 1 please input the column you would like to place your 'X' at : 0

        x       *       *
        *       *       *
        *       *       *

Player 2 please input the row you would like to place your 'o' at : 0
Player 2 please input the column you would like to place your 'o' at : 0

        x       *       *
        *       *       *
        *       *       *
You can't steal positions! Try again!

Player 2 please input the row you would like to place your 'o' at : _
```

Place player markers on game board

```
65          //prompt user to input the coordinates of where they would like to place their marker. X will always be player one
66          x_coord = RangeCheck(0,2, PromptInt("Player 1 please input the row you would like to place your 'X' at : "));
67          y_coord = RangeCheck(0,2, PromptInt("Player 1 please input the column you would like to place your 'X' at : "));
68          Console.WriteLine();
```

```
70          //for loop to modify our board with users input
71          for (int i = 0; i < rows; i++) {
72              for (int j = 0; j < columns; j++) {
73                  //our current position is x and y coordinates
74                  currentPos = gameBoard[x_coord, y_coord];
75
76                  //determine if our current position holds a '*'
77                  if (currentPos == "*") {
78                      //if current position '*' , replace it with an 'X'
79                      gameBoard[x_coord, y_coord] = markerX;
80
81                  }//end if
82
83                  //write updated board to console
84                  Console.Write("\t" + gameBoard[i, j]);
85              }//end inside for loop
86
87              //writes new line for row 1 and so on so the board is in a grid and not on one line
88              Console.WriteLine();
89          }//end outside for loop
```

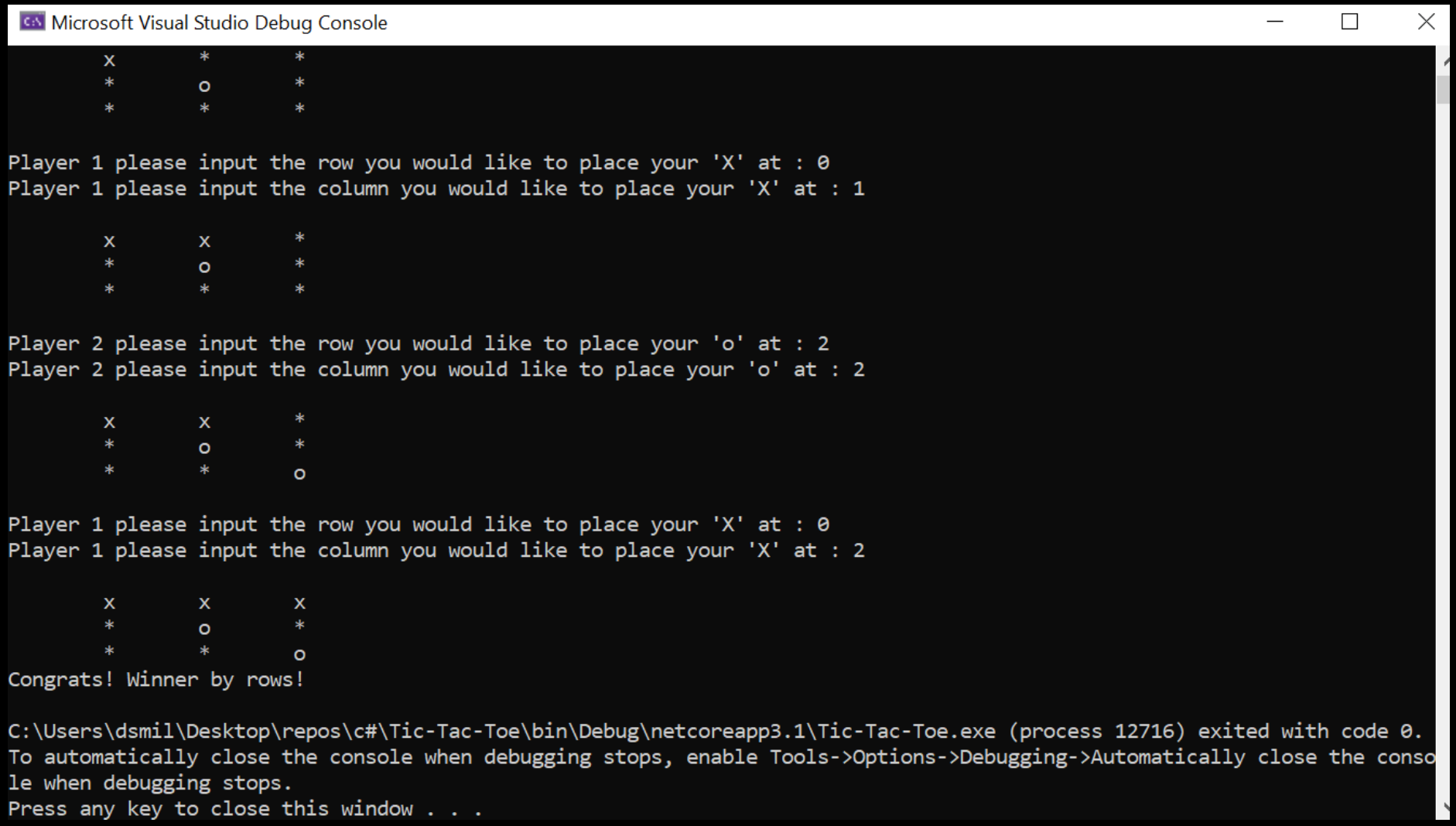
```
91          //check if the position on our board already has the opposite player marker
92          for (int i = 0; i < gameBoard.Length; i++) {
93              //our current position is x and y coordinates
94              currentPos = gameBoard[x_coord, y_coord];
95
96              //if our position already has the opposite marker then
97              if (currentPos == "o") {
98                  //display that they cannot steal that spot
99                  Console.WriteLine("You can't steal positions! Try again!");
100
101                 //recall our loop function so user can place a marker on the board
102                 GameLoop(gameBoard, x_coord, y_coord);
103
104             }//end if
105         }//end for loop
106     }//end function
```

- Prompt user for input. Asks for row and column number.
- For loops to traverse 2d array.
- Check current position on board.
- Write updated gameboard to screen.

Win conditions

- Compares our positions on the board for each win condition.
- If no condition is met the function should return –1.

```
164     static int CompleteRow(string[,] gameBoard)
165     {
166
167         if (gameBoard[0, 0] == gameBoard[0,1] && gameBoard[0, 0] == gameBoard[0,2] && gameBoard[0, 0] != "*") {
168             //winner in first row
169             return 1;
170         }
171         else if (gameBoard[1, 0] == gameBoard[1, 1] && gameBoard[1, 0] == gameBoard[1,2] && gameBoard[1, 0] != "*") {
172             //winner in second row
173             return 1;
174         }
175         else if (gameBoard[2, 0] == gameBoard[2,1] && gameBoard[2, 0] == gameBoard[2, 2] && gameBoard[2, 0] != "*") {
176             //winner in third row
177             return 1;
178         }
179
180         return -1;
181     }//end function
182     static int CompleteCol(string[,] gameBoard)
183     {
184         if (gameBoard[0, 0] == gameBoard[1,0] && gameBoard[0, 0] == gameBoard[2,0] && gameBoard[0, 0] != "*") {
185             //first column
186             return 2;
187         }
188         else if (gameBoard[0, 1] == gameBoard[1, 1] && gameBoard[0,1] == gameBoard[2,1] && gameBoard[0, 1] != "*") {
189             //second column
190             return 2;
191         }
192         else if (gameBoard[0,2] == gameBoard[1,2] && gameBoard[0,2] == gameBoard[2, 2] && gameBoard[0,2] != "*") {
193             //third column
194             return 2;
195         }
196
197         return -1;
198     }//end function
199     static int CompleteDiagonal(string[,] gameBoard)
200     {
201         if (gameBoard[2,0] == gameBoard[1,1] && gameBoard[2,0] == gameBoard[0,2] && gameBoard[2,0] != "*") {
202             //diagonal from bottom left to top right
203             return 3;
204         }
205         else if (gameBoard[0,2] == gameBoard[1,1] && gameBoard[0,2] == gameBoard[2,0] && gameBoard[0,2] != "*") {
206             //diagonal from top right to bottom left
207             return 3;
208         }
209         else if (gameBoard[2,2] == gameBoard[1,1] && gameBoard[2,2] == gameBoard[0,0] && gameBoard[2,2] != "*") {
210             //diagnal from bottom right to the top left
211             return 3;
212         }
213         else if (gameBoard[0,0] == gameBoard[1,1] && gameBoard[0,0] == gameBoard[2,2] && gameBoard[0,0] != "*") {
214             //diagonal from top left to bottom right
215             return 3;
216         }
217
218         return -1;
219     }//end function
```


Microsoft Visual Studio Debug Console
x * *
* o *
* * *
Player 1 please input the row you would like to place your 'X' at : 0
Player 1 please input the column you would like to place your 'X' at : 1
x x *
* o *
* * *
Player 2 please input the row you would like to place your 'o' at : 2
Player 2 please input the column you would like to place your 'o' at : 2
x x *
* o *
* * o
Player 1 please input the row you would like to place your 'X' at : 0
Player 1 please input the column you would like to place your 'X' at : 2
x x x
* o *
* * o
Congrats! Winner by rows!
C:\Users\dsmil\Desktop\repos\c#\Tic-Tac-Toe\bin\Debug\netcoreapp3.1\Tic-Tac-Toe.exe (process 12716) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging->Automatically close the conso
le when debugging stops.
Press any key to close this window . . .

Check for a winner

```
221     static int WinCheck(string[,] gameBoard, string markerX, string markerO, int gameTurns)
222     {
223         //tie
224         if (gameTurns >= 9) {
225             Console.WriteLine("Its a tie!");
226             return 1;
227         }//end if
228
229         //rows
230         if (CompleteRow(gameBoard) == 1) {
231             Console.WriteLine("Congrats! Winner by rows!");
232             return 1;
233         }//end if
234
235         //columns
236         if (CompleteCol(gameBoard) == 2) {
237             Console.WriteLine("Congrats! Winner by columns!");
238             return 1;
239         }//end if
240
241         //diagonally
242         if (CompleteDiagonal(gameBoard) == 3) {
243             Console.WriteLine("Congrats! Winner diagonally! ");
244             return 1;
245         }//end if
246
247         return -1;
248     }//end function
```

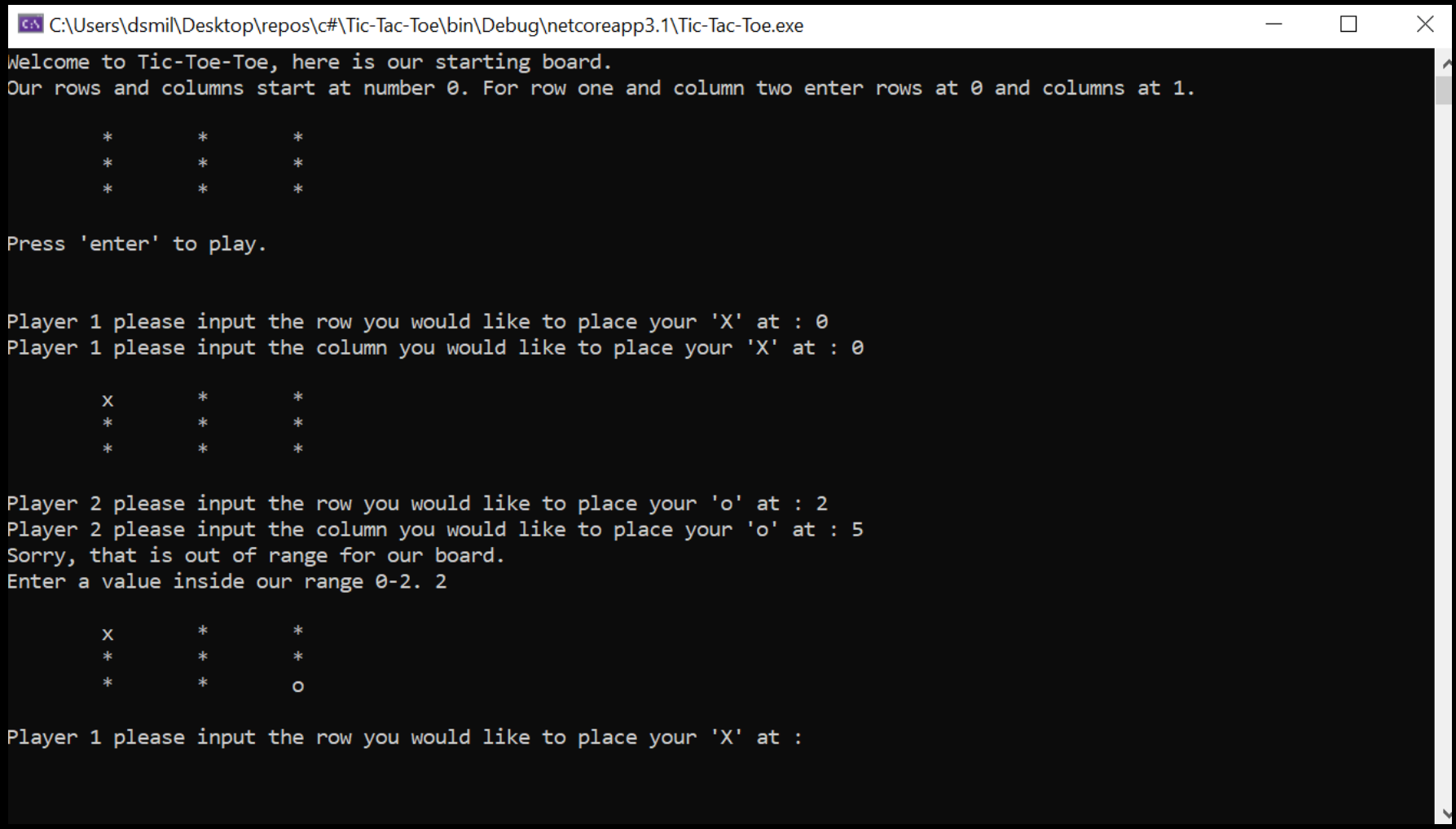
- Function will check for a winner based on the return values.
- Win by row, column, diagonal. If no win, a tie occurs.

Game loop

```
249     static void GameLoop(string[,] gameBoard, int x_coord, int y_coord)
250     {
251         //counter for our game to see if we have a tie or not
252         int gameTurns = 0;
253
254         //if our return value from winCheck is -1 that means no one has won yet and continue playing on
255         while (WinCheck(gameBoard, "x", "o", gameTurns) == -1) {
256
257             //player 1
258             PlaceMarkerX(gameBoard, "x", x_coord, y_coord);
259
260             //increase our counter
261             gameTurns++;
262
263             //after player ones move, check the board for a win, if we have one break out the loop.
264             if (CompleteRow(gameBoard) == 1 || (CompleteCol(gameBoard) == 2) || (CompleteDiagonal(gameBoard) == 3)) {
265                 //check if board has winner after player 1 turn. If player 1 has won it wont prompt player 2 to play.
266                 WinCheck( gameBoard, "x",  "o", gameTurns);
267                 break;
268
269             }//end if
270
271             //if we have a tie, break out of our loop
272             if (gameTurns == 9) {
273                 break;
274             }
```

```
276             //player 2 move now if above conditions are not true
277             PlaceMarkerO(gameBoard, "o", x_coord, y_coord);
278
279             //increase counter again for second player move
280             gameTurns++;
281
282             //if our return value from winCheck is not -1 than someone has won and we should break out of our loop
283             if (WinCheck(gameBoard, "x", "o", gameTurns) != -1) {
284                 break;
285             }//end if
286
287         }//end while
288     }//end function
```

- Check for a winner after players move.
- Check for a tie.
- Place player twos marker and increment our turns counter.


C:\Users\dsmil\Desktop\repos\c#\Tic-Tac-Toe\bin\Debug\netcoreapp3.1\Tic-Tac-Toe.exe
Welcome to Tic-Toe-Toe, here is our starting board.
Our rows and columns start at number 0. For row one and column two enter rows at 0 and columns at 1.
* * *
* * *
* * *
Press 'enter' to play.
Player 1 please input the row you would like to place your 'X' at : 0
Player 1 please input the column you would like to place your 'X' at : 0
x * *
* * *
* * *
Player 2 please input the row you would like to place your 'o' at : 2
Player 2 please input the column you would like to place your 'o' at : 5
Sorry, that is out of range for our board.
Enter a value inside our range 0-2. 2
x * *
* * *
* * o
Player 1 please input the row you would like to place your 'X' at :

Error Checking

```
313     static int RangeCheck(int lowestRange, int highestRange, int userRange)
314     {
315         //check if the user inputted values to place a marker on our board are within range
316         while (userRange < lowestRange || userRange > highestRange) {
317             //if not in range display message
318             Console.WriteLine("Sorry, that is out of range for our board. ");
319
320             //prompt the user to enter a new value
321             userRange = PromptInt("Enter a value inside our range 0-2. ");
322         }//end while
323
324         return userRange;
325     }//end function
```

```
123     //prompt user to input the coordinates of where they would like to place their marker. O will always be player 2.
124     x_coord = RangeCheck(0,2, PromptInt("Player 2 please input the row you would like to place your 'o' at : "));
125     y_coord = RangeCheck(0,2, PromptInt("Player 2 please input the column you would like to place your 'o' at : "));
126     Console.WriteLine();
```

- RangeCheck function will check our row and column number and evaluate if it is within range or not.

CLOSING THOUGHTS, QUESTIONS?