```
/*************************************************************
 *                     Anthony Rojas                         *
 *                     ID#011819338                          *
 *                     CECS 475                              *
 *                     Assignment 1                          *
 *************************************************************/
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Assignment1
{
    class Program
    {
        static void Main(string[] args)
        {
            TicTacToe game = new TicTacToe();
            game.PrintBoard();
            game.Play();
            Console.WriteLine("Press any key to exit...");
            Console.ReadKey();
        }
    }//end main class Program

    public class TicTacToe
    {
        private const int BOARDSIZE = 3;
        private int[,] board;
        private int player1;
        private int player2;
        public TicTacToe()
        {
            board = new int[BOARDSIZE, BOARDSIZE];
            InitializeBoard();
            player1 = 0;
            player2 = -1;
        }

        public void InitializeBoard()
        {
            for (int i = 0; i < BOARDSIZE; i++)
            {
                for (int j = 0; j < BOARDSIZE; j++)
                {
                    if (i > 0)
                    {
                        board[i, j] = board[i - 1, BOARDSIZE - 1] + (j + 1);
```

```
50                      }
51                      else
52                      {
53                          board[i, j] = (j + 1) * (i + 1);
54                      }
55                  }
56              }
57          }
58
59
60
61          public void PrintBoard()
62          {
63              for (int i = 0; i < 3; i++)
64              {
65                  Console.WriteLine("\n");
66                  for (int j = 0; j < 3; j++)
67                  {
68                      if (board[i, j] == player1)//player 1 = X
69                      {
70                          Console.ForegroundColor = ConsoleColor.Red;
71                          Console.Write("\tX\t");
72                      }
73                      else if (board[i, j] == player2)//player 2 = O
74                      {
75                          Console.ForegroundColor = ConsoleColor.Yellow;
76                          Console.Write("\tO\t");
77                      }
78                      else//space has not been allocated by either player 1 or
                        player 2
79                      {
80                          Console.Write("\t" + board[i, j] + "\t");
81                      }
82                      Console.ResetColor();
83                  }
84              }
85              Console.WriteLine("\n");
86          }
87
88          public Boolean CheckWinner(int player)
89          {
90              if (board[0, 0] == player && board[0, 1] == player && board[0, 2]
                  == player)
91              {//row 0
92                  return true;
93              }
94              else if (board[1, 0] == player && board[1, 1] == player && board[1,
                   2] == player)
95              {//row 2
```

```
96                  return true;
97              }
98              else if (board[2, 0] == player && board[2, 1] == player && board[2, 2] == player)
99              {//row 3
100                 return true;
101             }
102             else if (board[0, 0] == player && board[1, 0] == player && board[2, 0] == player)
103             {//column 1
104                 return true;
105             }
106             else if (board[0, 1] == player && board[1, 1] == player && board[2, 1] == player)
107             {//column 2
108                 return true;
109             }
110             else if (board[0, 2] == player && board[1, 2] == player && board[2, 2] == player)
111             {//column 3
112                 return true;
113             }
114             else if (board[0, 0] == player && board[1, 1] == player && board[2, 2] == player)
115             {//foward diagonal
116                 return true;
117             }
118             else if (board[0, 2] == player && board[1, 1] == player && board[2, 0] == player)
119             {//backward diagonal
120                 return true;
121             }
122             return false;//default has not won
123         }
124
125         public void Play()
126         {
127             int moveCount = 0;
128             bool gameOver = false;
129             Console.WriteLine("Player 1: X \t\t Player 2: O");
130             while (!gameOver)
131             {
132                 moveCount++;
133                 if (moveCount % 2 != 0)
134                 {//player 1
135                     MakeMove(player1);
136                     if (CheckWinner(player1) == true)
137                     {
138                         Console.WriteLine("\nPlayer 1 Wins!");
```

```
139                         PrintBoard();
140                         return;
141                     }
142                 }
143                 else
144                 {
145                     MakeMove(player2);
146                     if (CheckWinner(player2) == true)
147                     {
148                         Console.WriteLine("\nPlayer 2 Wins!");
149                         PrintBoard();
150                         return;
151                     }
152                 }
153                 if (moveCount == (BOARDSIZE*BOARDSIZE))
154                 {
155                     gameOver = true;
156                     if (CheckTie(moveCount) == true)
157                     {
158                         gameOver = true;
159                         Console.WriteLine("It's a tie!");
160                     }
161                 }
162                 PrintBoard();
163             }
164         }
165 
166         public void MakeMove(int player)
167         {
168             string moveSelect;
169             string playerIndicator = "";
170             if (player == 0)
171             {//player 1 turn
172                 playerIndicator = "Player 1 (X)";
173             }
174             else
175             {//player 2 turn
176                 playerIndicator = "Player 2 (O)";
177             }
178             do
179             {
180                 Console.WriteLine(playerIndicator + " turn.");
181                 Console.WriteLine("Enter an integer corresponding to the
                      location on the board you would like to move to. (1-9)");
182                 moveSelect = Console.ReadLine();
183             } while (ValidateInput(moveSelect) == false);
184             for (int i = 0; i < BOARDSIZE; i++)
185             {
186                 for (int j = 0; j < BOARDSIZE; j++)
```

```
187                 {
188                     if (Convert.ToInt32(moveSelect) == board[i, j])
189                     {
190                         board[i, j] = player;
191                         i = BOARDSIZE;
192                         j = BOARDSIZE;
193                     }
194                 }
195             }
196         }
197 
198         public Boolean ValidateInput(object moveSelect)
199         {
200             int move = 0;
201             try
202             {
203                 move = Convert.ToInt32(moveSelect);
204             }
205             catch (FormatException e)
206             {
207                 Console.WriteLine("Entry must be an integer. Try again.");
208                 PrintBoard();
209                 return false;
210 
211             }
212             Console.WriteLine("Move: " + move);
213             if (move <= 0 || move > (BOARDSIZE * BOARDSIZE))
214             {
215                 Console.WriteLine("Move selection is out of bounds. Try
                      again");
216                 PrintBoard();
217                 return false;
218             }
219             if (move > 0 && move <= (BOARDSIZE * BOARDSIZE) && ValueExists
                  (move) == false)
220             {
221                 Console.WriteLine("That space is taken. Please select a space
                      with an available value from 1-9 on the board.");
222                 PrintBoard();
223                 return false;
224             }
225             return true;
226         }
227 
228         public Boolean ValueExists(int moveSelect)
229         {
230             for (int i = 0; i < BOARDSIZE; i++)
231             {
232                 for (int j = 0; j < BOARDSIZE; j++)
```

```
233                 {
234                     if (board[i, j] == moveSelect)
235                     {
236                         return true;
237                     }
238                 }
239             }
240             return false;
241         }
242 
243         public Boolean CheckTie(int numMoves)
244         {
245             if (numMoves == (BOARDSIZE * BOARDSIZE))//checks if there are no
                  places to make a move to
246             {
247                 return true;
248             }
249             return false;
250         }
251     }//end class TicTacToe
252 }
```