```
/* C program by Dave Russillo. Made for CS1310. TicTacToe.
 2
 3
 4
 5
 6
 7
 8
 9
10
11
12
13
      #include <stdio.h>
14
      char ttt[3][3]; // array represents board
char player; // current player: either X or 0
int row; // variable used for loops and more
int col; // variable used for loops and more
15
16
17
18
19
      int moves; // tracks number of moves
20
21
22
23 void draw_board_options(void) { // draws equivalency of numbers [1-9] to places on board
         printf("\n"
24
                    "_1_|_2_|_3_\n"
"_4_|_5_|_6_\n"
" 7 | 8 | 9 \n\n");
25
26
27
28
29
30
31
32 void draw_board(int winstate) { // draws board and current status. adds lines based on winstate
33
34
          int wintype = winstate / 10; // relies on truncating
35
          int wincoord = winstate - winstate / 10 * 10; // relies on truncating
36
37日
38日
39日
40日
         for(row = 0; row < 3; row++) { // iterate through rows
  for(i = 0; i < 5; i++) { // iterate through character lines within the row</pre>
               for(col = 0; col < 3; col++) {    // iterate through each column
    if(ttt[row][col] == ' ') {        // based on which line it's on, draw different parts of the ascii for X
        if(i == 4 && row != 2) printf("_____");        // separator between rows: dont add if last row</pre>
41
42
                     else printf("
                                                    ");
                     if(col != 2) printf("|"); // dont add column separator if last column
43
```

```
} else {
45 🖨
                     switch(i) {
                       case 0: // first line in row
46
                                                                                                    "); // if win on column, add pipe
"); // if backward win, add backslashes
/"); // if backward win, add slash
47
                          if(wintype == 2 && wincoord == col) printf("
                          else if(wintype == 3 && row == col) printf("\\\\
48
49
                          else if(wintype == 4 && row + col == 2) printf("
                                                       "); // if no win yet
50
                          else printf("
51
                          break;
52
                       case 1: // second line in row
                          if(ttt[row][col] == 'X') {
53 🛱
                            if(wintype == 2 && wincoord == col) printf(" X|X "); // if win on column, add pipe
else if(wintype == 3 && row == col) printf(" \\X X "); // if backward win, add backslash
else if(wintype == 4 && row + col == 2) printf(" X X//"); // if forward win, add slashes
54
55
56
57
                             else printf(" X X "); // if no win yet
58
                          } else {
                             if(wintype == 2 && wincoord == col) printf(" 0|0 "); // if win on column, add pipe
else if(wintype == 3 && row == col) printf(" \\\000 "); // if backward win, add backslash
else if(wintype == 4 && row + col == 2) printf(" 000// "); // if forward win, add slashes
59
60
61
                             else printf(" 000 "); // if no win yet
62
63
64
                         break;
65
                       case 2: // third line in row
66 [
                          if(ttt[row][col] == 'X') {
                             if(wintype == 1 && wincoord == row) printf("======="); // if win on row, add equal signs
67
                             else if(wintype == 2 && wincoord == col) printf(" | ");
else if(wintype == 3 && row == col) printf(" |\\\\ "); // if backward win, add backslashes
else if(wintype == 4 && row + col == 2) printf(" // "); // if forward win, add slashes
68
69
70
                                                   X "); // if no win yet
71
                             else printf("
72
                          } else {
73
                             if(wintype == 1 && wincoord == row) printf("==0===0=="); // if win on row, add equal signs
                            74
75
76
                             else printf(" 0 0 "); // if no win yet
77
78
79
                          break:
80
                       case 3: // fourth line in row
81 🖨
                          if(ttt[row][col] == 'X') {
                            if(wintype == 2 && wincoord == col) printf(" X|X "); // if win on column, add pipe
else if(wintype == 3 && row == col) printf(" X X\\\\"); // if backward win, add backslashes
else if(wintype == 4 && row + col == 2) printf(" /X X "); // if forward win, add slash
82
83
84
                             else printf(" X X "); // if not win yet
85
                          } else {
```

```
88
 89
 90
 91
 92
                           break;
  93
                         case 4: // last line in row
                           if(row != 2) { // if not on last row, add separator between rows
if(wintype == 2 && wincoord == col) printf("___|__"); // if win on column, add pipe
else if(wintype == 3 && row == col) printf("___\"); // if backward win, add backslash
else if(wintype == 4 && row + col == 2) printf("//___"); // if forward win, add slash
 94 🖨
  95
  96
 97
 98
                              else printf("_____"); // if no win yet
 99
                           } else {
                              if(wintype == 2 && wincoord == col) printf(" | "); // if win on column, add pipe
else if(wintype == 3 && row == col) printf(" | \\"); // if backward win, add backslash
else if(wintype == 4 && row + col == 2) printf("// "); // if forward win, add slash
100
101
102
                                                             "); // if no win yet
103
                              else printf("
104
105
                      // separator between columns
106
                      if(col != 2 && wintype == 1 && wincoord == row && i == 2) printf("="); // if win is on row, add equal sign else if(wintype == 3 && row == col && i == 4) printf("\\"); // add backslash instead of pipe if it's a backward win else if(wintype == 4 && row + col == 2 && i == 0) printf("/"); // add slash instead of pipe if it's a backward win
107
108
109
110
                      else if(col != 2) printf("|"); // dont add pipe if last column
111
112
113
114
                printf("\n"); // newline between each line in row
115
116
          if(wintype == 5) { // if it's a tie
printf("\nTie! \n\n");
117 白
118
           } else if(wintype != 0) {
119
             printf("\n%c won! \n\n", player);
120
121
121 [ }
123
124
125
127日
128日
         129
```

```
130 -
131 - }
132 moves = 0; // reset moves
133 }
134
135
136
137 ☐ void take_turn(void) { // lets current player take turn and checks for valid move
138
        int position;
139
         int position_valid = 0;
        int x = 0;
140
141
        int y = 0;
142
143 🛱
        while(position_valid == 0) { // check for valid move
          printf("\nIt's %c's turn. Choose an empty field (1-9): ", player); // prints instructions
scanf("%d", &position);
144
145
           position_valid = 1; // assumes valid
switch(position) {  // sets coords based on position number
146
147 🖨
148
            case 1:
149
               x = 0;
               y = 0;
150
151
               break;
152
             case 2:
153
               x = 0;
154
               y = 1;
155
               break;
156
             case 3:
157
               x = 0;
               y = 2;
158
159
               break;
160
             case 4:
161
               x = 1;
162
               y = 0;
163
               break;
164
             case 5:
165
               x = 1;
166
               y = 1;
167
               break;
168
             case 6:
169
               x = 1;
170
               y = 2;
171
               break;
172
             case 7:
```

```
173
            x = 2;
174
            y = 0;
175
            break;
176
          case 8:
177
            x = 2;
178
            y = 1;
179
            break;
180
          case 9:
181
            x = 2;
182
            y = 2;
183
            break;
184
          default: // means position is invalid because not in [1-9]
185
            position_valid = 0;
186
            break;
187
         if(ttt[x][y] == ' ' && position_valid == 1) { // check if spot is free
188 🖨
189
          ttt[x][y] = player;
190
         } else {
          position_valid = 0;
191
          192
193
          draw_board_options();
194
          draw_board(0);
195
          printf("Invalid input! Try again.");
196
197
198
       moves++; // increment moves to keep track of them
199 L }
200
201
202
203 □ void reassign_player(void) { // switches player between X and 0
      if(player == 'X')
204
205
        player = '0';
206
       else
207
         player = 'X';
208 L }
209
210
211
212 ☐ int check_for_end(void) { // returns wintype in tenths and coord number in units
       // stores two single digit ints in double digit int: 2, 4 -> 24
213
214
       int i;
215
```

```
for (i = 0; i < 3; i++) { // i is which row or column the win is on
          if(ttt[i][0] != ' ' && ttt[i][0] == ttt[i][1] && ttt[i][1] == ttt[i][2]) {
  return 10 + i; // wintype 1 is horizontal win
217 🗀
218
219
          } else if(ttt[0][i] != ' ' && ttt[0][i] == ttt[1][i] && ttt[1][i] == ttt[2][i]) {
220
            return 20 + i; // wintype 2 is vertical win
221
222
223 🖨
        if(ttt[0][0] != ' ' && ttt[0][0] == ttt[1][1] && ttt[1][1] == ttt[2][2]) {
        return 30; // wintype 3 is backward diagonal (needs no coord)
} else if(ttt[0][2] != ' ' && ttt[0][2] == ttt[1][1] && ttt[1][1] == ttt[2][0]) {
224
225
          return 40; // wintype 4 is forward diagonal (needs no coord)
226
227
        } else if(moves == 9) {
          return 50; // wintype 5 is for ties
228
229
230
        return 0:
231 L }
232
233
234
235 ☐ int check_for_two_in_row(int coords) {
236
        int r = coords / 10; // extract row out of coords. relies on truncating.
        int c = coords % 10; // extract column out of coords
237
238
        char current = ttt[r][c]; // current sign in cell
239
        char opposite;
        int i; // to iterate (row and col already in use)
240
        int has_current; // keeps track if there's the current sign on the row/column/diagonal
241
        int has_opposite; // keeps track if there's an opposite to the current sign on the row/column/diagonal
242
243
        if(current == 'X') { // find opposite
244 🗀
          opposite = '0';
245
246
        } else {
247
          opposite = 'X';
248
249
250 🖨
        if(ttt[r][c] == ' ') { // don't check if given cell is empty
251
          return 0;
252
        } else {
          ttt[r][c] = ' ';
253
254
255
256
        // assume coords are not (1, 1) because middle will be occupied either way
257 白
        if(coords == 11) {
258
          return 0;
```

```
259 -
260
261
        // check each column on row
262
        has_current = 0;
263
        has_opposite = 0;
264日
265日
        for(i = 0; i < 3; i++) {
          if(ttt[r][i] == current) {
           has_current = 1;
266
          } else if(ttt[r][i] == opposite) {
267
268
           has_opposite = 1; }
269
270 🖨
       if(has_current == 1 && has_opposite == 0) {
271
          ttt[r][c] = current;
          return 1;
272
273
274
        // check each row on column
275
276
        has_current = 0;
277
        has_opposite = 0;
278 🛱
        for(i = 0; i < 3; i++) {
279 🛱
          if(ttt[i][c] == current) {
280
           has_current = 1;
281
          } else if(ttt[i][c] == opposite) {
282
           has_opposite = 1;
283
284
285
       if(has_current == 1 && has_opposite == 0) {
286
          ttt[r][c] = current;
287
          return 1;
288
289
290
        // check each cell on diagonal
291
        has_current = 0;
292
        has_opposite = 0;
        for(i = 0; i < 3; i++) {
293 🖨
294 🖨
          if(ttt[i][i] == current) {
295
           has_current = 1;
296
          } else if(ttt[i][i] == opposite) {
297
           has_opposite = 1;
298
299
       if(has_current == 1 && has_opposite == 0) {
300 🖨
301
         ttt[r][c] = current;
```

```
return 1;
}
302
303
304
                  // check each cell on backward diagonal
has_current = 0;
has_opposite = 0;
for(i = 0; i < 3; i++) {
   if(ttt[i][2 - i] == current) {
      has_current = 1;
   } else if(ttt[i][2 - i] == opposite) {
      has_opposite = 1;
   }</pre>
305
306
307
308 E
309 E
310
311
312
313
314 <del>|</del>
315 □
                   if(has_current == 1 && has_opposite == 0) {
316
                      ttt[r][c] = current;
317
                     return 1;
318
319
                 ttt[r][c] = current;
return 0;
320
321
322 \ }
323
324
325
            // can make 3 in a row
326
          // can make 3 in a row
// block an opponents 3 in a row
// make 2 in a row
// place in middle if available
// place in corner
// place in available spot
327
328
329
330
331
332
333 ₱ void take_turn_cpu(void) {
334
335
                 int coords;
336
                  // 3 in a rows
for(row = 0; row < 3; row++) {
    for(col = 0; col < 3; col++) {
        if(ttt[row][col] == ' ') {
            ttt[row][col] = ' ') * / test with 0 for three in a rows
        if(check_for_end() == 0) { // if it's not the end
            ttt[row][col] = 'X'; // test with X for three in a rows
        if(check_for_end() == 0) { // if it's not the end
            ttt[row][col] = ' '; // clear because no 3 in a rows</pre>
337 F
338 F
339 F
340 
341 □
342 |
343 |=
344 |
```

```
345
                 } else {
346
                   ttt[row][col] = '0';
                   moves++;
347
348
                   return;
349
350
               } else {
351
                 ttt[row][col] = '0';
352
                 moves++;
353
                 return;
354
355
356
357
358
         // 2 in a rows
359
360 🚍
         for(row = 0; row < 3; row++) {</pre>
361
           for(col = 0; col < 3; col++) {
             if(ttt[row][col] == ' ') {
362
               ttt[row][col] = '0'; // test with 0 for two in a rows
363
               coords = row * 10 + col; // store current single digit coords in double digit int
364
365
               if(check_for_two_in_row(coords) == 0) { // if it's not two 0 in a row
                   ttt[row][col] = ' '; // clear because no 2 in a rows
366
367
               } else {
368
                 ttt[row][col] = '0'; // set to 0 to two in a row
369
                 moves++; // keep track of moves
370
                 return;
371
372
             }
373
374
375
         if(ttt[1][1] == ' ') { // if middle is available
376 -
         ttt[1][1] = '0'; // set middle to '0'
} else if(ttt[0][0] == ' ') { // if top left is available
377
378
           ttt[0][0] = '0'
379
380
         } else if(ttt[0][2] == ' ') { // if top right is available
381
           ttt[0][2] = '0';
         } else if(ttt[2][0] == ' ') { // if bottom left is available
382
           ttt[2][0] = '0';
383
         } else if(ttt[2][2] == ' ') { // if bottom right is available
384
385
           ttt[2][2] = '0';
386
         } else {
387 <del>-</del>
           for(row = 0; row < 3; row++) {
             for(col = 0; col < 3; col++) {
               if(ttt[row][col] == ' ') {
389 🗐
                ttt[row][col] = '0'; // pick first available spot
390
391
392
393 -
```

```
388 -
389
       moves++;
390 L }
391
392
393
394 ☐ int main(void) {
395
       char input; // user input. used for capturing newlines
396
       char restart;
397
       int gamemode;
398
399 🖨
400
         printf("\nWelcome to TicTacToe! First to get three in a row, column, or diagonal wins! X goes first. \n");
401
         player = 'X';
402
403 🖨
         while(gamemode != 1 && gamemode != 2) {
          printf("
404
405
406
                 " | 1 : Player vs Player
407
                                        \n"
408
409
                                       \n\n");
410
           printf("Select a gamemode... ");
411
           scanf("%d", &gamemode);
412 🖨
           if(gamemode != 1 && gamemode != 2) {
413
            printf("Invalid gamemode selection. Try again. \n\n");
414
415
416
         clear_board();
417 🖨
         while(check_for_end() == 0) {
418
          419
           draw_board_options();
420
           draw_board(0);
421
422 🖨
           if(gamemode == 1 || player == 'X') {
423
            take_turn();
424
          } else {
425
            take_turn_cpu();
426
427
          reassign_player();
428
429
         reassign player();
430
         431
        draw board options();
432
        draw_board(check_for_end());
433
        scanf("%c", &input);
434
        printf("Would you like to restart? (Y/N)");
      scanf("%c", &restart);
scanf("%c", &input);
} while(restart == 'y' || restart == 'Y');
printf("Thank you for playing! \n\n");
435
436
437
438
439
440
      return 0;
441 L }
442
Welcome to TicTacToe! First to get three in a row, column, or diagonal wins! X goes first.
  1 : Player vs Player
  2 : Player vs CPU
Select a gamemode...
```

**INPUT 1** 



_1_ _2_ _3_ _4_ _5_ _6_ 7   8   9		
X X X X X X X X X X X X X X X X X X X	000 0 0 000	
	X X// // /X X //	//   000   0 0   000
		000 0 0 000
X won! Would you like to restart? (Y/N)		
_1_ _2_ _3_ _4_ _5_ _6_ _7   8   9		
X X   X X   X X	000   0 0   000	X X X X X X X X X X X X X X X X X X X
000   0 0   000	000   0 0   000	X X X X X
X X   X X   X X   X X   X X   X X   X X   X X   X X   X X   X X   X X   X X   X X X   X X X   X X X X   X X X X   X X X X   X	X X   X   X X	000 0 0 000
Tie! Would you like to restart? (Y/N)		

```
|_2_|_3_
|_5_|_6_
| 8 | 9
                  ХХ
    ХХ
                                000
                  X
X X
                              000
    X
                                {\displaystyle \mathop{\mathsf{X}}_{\mathsf{X}}}{\displaystyle \mathop{\mathsf{X}}_{\mathsf{X}}}
                  X
                  ХХ
                                ХХ
    000
                  000
                                000
==0===0===0===0====0===0==
    000
                  000
                                000
0 won!
Would you like to restart? (Y/N)
Welcome to TicTacToe! First to get three in a row, column, or diagonal wins! X goes first.
  1 : Player vs Player
  2 : Player vs CPU
```

## **INPUT 2**

Select a gamemode...



Player is X and CPU plays O (smart)