```
 1: /*********************************************
 2:  * tictac.c
 3:  * Doug Lloyd
 4:  * September 22, 2010
 5:  *
 6:  * A working, text-based, Tic-Tac-Toe game
 7:  *********************************************/
 8:
 9: /* Header files */
10: #include <stdio.h>
11: #include <cs50.h>
12:
13: /* Global variables and constants */
14: #define DIM 3
15: #define MAX_SQ 9
16: #define BLANK '_'
17: char ttt_board[DIM][DIM];
18:
19: /* Function Declarations */
20: void init_board();
21: bool is_valid_move(int sq);
22: void make_move(char c, int sq);
23: void print_board();
24: bool game_over();
25: bool check_row(int row);
26: bool check_col(int col);
27: bool check_diag(int diag);
28: bool all_full();
29:
30: /* Function Definitions */
31: int main(int argc, char *argv[]) {
32:   init_board();
33:   print_board();
34:   int sq = 0;
35:   char c = BLANK;
36:   while(!game_over()) {
37:     do {
38:       printf("Enter your square (1-9): ");
39:       sq = GetInt();
40:     } while((sq < 1) || (sq > MAX_SQ) || !is_valid_move(sq));
41:     do {
42:       printf("Enter your letter (X/O): ");
43:       c = GetChar();
44:     } while((c != 'X') && (c != 'O'));
45:     make_move(c, sq);
46:     print_board();
47:   }
48:   printf("\nGame Over!\n");
49:   return 0;
50: }
51:
52: void init_board() {
53:   for(int i = 0; i < DIM; i++)
54:     for(int j = 0; j < DIM; j++)
55:       ttt_board[i][j] = BLANK;
56:   return;
57: }
58:
59: bool is_valid_move(int sq) {
60:   int usable_num = (sq - 1);
61:   int row = usable_num / DIM;
62:   int col = usable_num % DIM;
63:
64:   if(ttt_board[row][col] == BLANK)
```

```
 65:     return true;
 66:   else
 67:     return false;
 68: }
 69:
 70: void make_move(char c, int sq) {
 71:   int usable_num = (sq - 1);
 72:   int row = usable_num / DIM;
 73:   int col = usable_num % DIM;
 74:
 75:   ttt_board[row][col] = c;
 76:   return;
 77: }
 78:
 79: void print_board() {
 80:   printf("\n\n\n");
 81:   printf("-------------\n");
 82:   for(int j = 0; j < DIM; j++) {
 83:     printf("|");
 84:     for(int k = 0; k < DIM; k++)
 85:       printf(" %c |", ttt_board[j][k]);
 86:     printf("\n-------------\n");
 87:   }
 88:   return;
 89: }
 90:
 91: bool game_over() {
 92:   if(check_row(0) || check_row(1) || check_row(2))
 93:     return true;
 94:   if(check_col(0) || check_col(1) || check_col(2))
 95:     return true;
 96:   if(check_diag(1) || check_diag(2))
 97:     return true;
 98:   if(all_full())
 99:     return true;
100:   return false;
101: }
102:
103: bool check_row(int row) {
104:   if(ttt_board[row][0] != ttt_board[row][1])
105:     return false;
106:   if(ttt_board[row][1] != ttt_board[row][2])
107:     return false;
108:   if(ttt_board[row][0] == BLANK)
109:     return false;
110:   return true;
111: }
112:
113: bool check_col(int col) {
114:   if(ttt_board[0][col] != ttt_board[1][col])
115:     return false;
116:   if(ttt_board[1][col] != ttt_board[2][col])
117:     return false;
118:   if(ttt_board[0][col] == BLANK)
119:     return false;
120:   return true;
121: }
122:
123: bool check_diag(int diag) {
124:   if(diag == 1) {
125:     if(ttt_board[0][0] != ttt_board[1][1])
126:       return false;
127:     if(ttt_board[1][1] != ttt_board[2][2])
128:       return false;
```

```
129:   } else {
130:     if(ttt_board[0][2] != ttt_board[1][1])
131:       return false;
132:     if(ttt_board[1][1] != ttt_board[2][0])
133:       return false;
134:   }
135:   if(ttt_board[1][1] == BLANK)
136:     return false;
137:   return true;
138: }
139:
140: bool all_full() {
141:   for(int i = 0; i < DIM; i++)
142:     for(int j = 0; j < DIM; j++)
143:       if(ttt_board[i][j] == BLANK)
144:         return false;
145:   return true;
146: }
```