

```
1 """Tic tac toe helper functions
2
3 Author: James Lin
4 Honor Code: I affirm that I have carried out the
5 attached academic endeavors with full academic honesty,
6 in accordance with the Union College Honor Code and the
7 course syllabus.
8 """
9
10 def print_board(board):
11     """
12         Prints the board on the console
13         :param board: a list of X and O boards
14     """
15     num_rows = len(board)
16     num_cols = len(board[0])
17     for row_num, row in enumerate(board):
18         row_str = ''
19         for col_num, marker in enumerate(row):
20             row_str += marker
21             if col_num < num_cols - 1:
22                 row_str += ' | '
23             print(row_str)
24             if row_num < num_rows - 1:
25                 print('-----')
26
27 def row_all_same(board, row):
28     """
29         Checks if the rows are all the same
30         :param board: a list of X and O boards
31         :param row: a list of rows in the board
32         :return: a boolean value
33     """
34     return (board[row][0] == board[row][1] == board[row][2])
35
```

```
36 def column_all_same(column):
37     """
38     Checks if the column is the same
39     :param column: a list of column vals
40     :return: a boolean value
41     """
42     return (column[0] == column[1] == column[2])
43
44
45 def diagonal_all_same(diagonal):
46     """
47     Check if the diagonal is the same
48     :param diagonal: a list of diagnol vals
49     :return: a boolean value
50     """
51     return (diagonal[0] == diagonal[1] == diagonal[2])
52
53
54 def get_back_slash(board):
55     """
56     Gets the back slashes from the board
57     :param board: a list of X and O boards
58     :return: return the list of back slash only
59     """
60     return [board[i][i] for i in range(len(board))]
61
62
63 def get_forward_slash(board):
64     """
65     Gets the forward slash from the board
66     :param board: a list of X and O boards
67     :return: return the list of forward slash only
68     """
69     return [board[len(board)-i-1][i] for i in range(len(board))]
70
71
72 def columns(board):
```

```
73     """
74     Gets the columns of the board
75     :param board: a list of X and O boards
76     :return: a list of columns of the board
77     """
78     num_cols = len(board[0])
79     num_rows = len(board)
80
81     to_return = []
82
83     for i in range(num_cols):
84         col_str = ''
85         for j in range(num_rows):
86             col_str += board[j][i]
87         to_return.append(col_str)
88     return to_return
89
90
91 def check_winner(board):
92     """
93     Check for the winner of the board
94     :param board: a list of X and O boards
95     :return: the winner of the board
96     """
97     for row_num, row in enumerate(board):
98         if row_all_same(board, row_num):
99             winner = board[row_num][0]
100            return winner
101
102     for col in columns(board):
103         if column_all_same(col):
104             winner = col[0]
105             return winner
106
107     if diagonal_all_same(get_back_slash(board)):
108         winner = board[0][0]
109         return winner
110
```

```
111     if diagonal_all_same(get_forward_slash(board)):
112         winner = board[2][0]
113     return winner
114
115
116 def get_board_from_file(filename):
117     """
118     Create a list of boards from a file
119     :param filename: the name of the file
120     :return: a list of boards
121     """
122     board_list = []
123     board_file = open(filename, "r")
124     for line in board_file:
125         board_list.append(line.strip())
126     board_file.close()
127     return board_list
128
129 def main():
130     inputfile = 'input.txt'
131     board = get_board_from_file(inputfile)
132
133     print_board(board)
134
135     winner = check_winner(board)
136
137     if winner != '':
138         print(winner + ' WINS!!!!')
139     else:
140         print("TIE GAME!!!!")
141
142 if __name__ == '__main__':
143     main()
144
145
146
147 """
148 Reflection:
```

149 1. Why is the logic inside badmain's main-line code ambiguous and hard to follow?

150 The logic inside badmain's main-line code is hard to follow because of the helper functions not actually utilizing their parameters. Instead they are using global variables,

151 this is not a good way to write code, especially in the industry, when you have to work as a team on a project. The global variables will cause many confusions and hard to track.

152

153 2) How does my refactoring remove this ambiguity?

154 Now all my helper functions uses its parameters instead of global variables. The main function is also way easier to track and understand after removing all the messy variables.

155

156 """