```
 1 """
 2 Functions for a tic tac toe board and determining its
    winner
 3
 4 :author: Neil Daterao
 5 :note: I affirm that I have carried out the attached
   academic endeavors with full academic honesty,
 6 in accordance with the Union College Honor Code and
   the course syllabus.
 7 """
 8
 9 def print_board(board):
10     """
11     Prints a tic tac toe board
12     :param board: A valid tic-tac-toe board
13     :return: None
14     """
15
16     num_rows = len(board)
17     num_cols = len(board[0])
18     for row_num, row in enumerate(board):
19         row_str = ''
20         for col_num, marker in enumerate(row):
21             row_str += marker
22             if col_num < num_cols - 1:
23                 row_str += ' | '
24         print(row_str)
25         if row_num < num_rows - 1:
26             print('---------')
27
28
29 def row_all_same(board, row):
30     """
31     Checks the board to see if the row is all the
   same
32     :param board: A tic tac toe board
33     :param row: A specific row on tic tac toe board
34     :return: Boolean, True if all the same, false if
   not
35     """
36     return (board[row][0] == board[row][1] == board[
```

```
36 row][2])
37
38
39 def column_all_same(column):
40     """
41     Checks the board to see if the column is all the
   same
42     :param column: A column of the tic tac toe board
43     :return: Boolean, True if all the same, false if
   not
44     """
45     return (column[0] == column[1] == column[2])
46
47
48 def diagonal_all_same(diagonal):
49     """
50     Checks the board to see if the diagonals are all
   the same
51     :param diagonal: The diagonal of a tic tac toe
   board
52     :return: Boolean, True if all the same, false if
   not
53     """
54     return (diagonal[0] == diagonal[1] == diagonal[2
   ])
55
56
57 def get_back_slash(board):
58     """
59     Gets diagonal that looks like a back slash on a
   tic tac toe board
60
61     :param board: A valid tic tac toe board
62     :return: A list of the the characters of the
   diagonal back slash of the tic tac toe board
63     """
64     return [board[i][i] for i in range(len(board))]
65
66
67 def get_forward_slash(board):
68     """
```

```
 69     Gets diagonal that looks like a forward slash on
     a tic tac toe board
 70
 71     :param board: A valid tic tac toe board
 72     :return: A list of the the characters of the
    diagonal forward slash of the tic tac toe board
 73     """
 74     return [board[len(board)-i-1][i] for i in range(
    len(board))]
 75
 76
 77 def columns(board):
 78     """
 79     Gets the columns from the board
 80     :param board: A valid tic tac toe board
 81     :return: The columns from the board stored in a
    list
 82     """
 83     num_cols = len(board[0])
 84     num_rows = len(board)
 85
 86     columns_of_board = []
 87
 88     for i in range(num_cols):
 89         col_str = ''
 90         for j in range(num_rows):
 91             col_str += board[j][i]
 92         columns_of_board.append(col_str)
 93
 94     return columns_of_board
 95
 96
 97 def check_winner(board):
 98     """
 99     Determines if there's a winner given a tic tac
    toe board
100
101     :param board: A valid tic tac toe board
102     :return: The winner as a string
103     """
104     for row_num, row in enumerate(board):
```

```
105         if row_all_same(board, row_num):
106             winner = board[row_num][0]
107             return winner
108 
109     for col in columns(board):
110         if column_all_same(col):
111             winner = col[0]
112             return winner
113 
114     if diagonal_all_same(get_back_slash(board)):
115         winner = board[0][0]
116         return winner
117 
118     if diagonal_all_same(get_forward_slash(board)):
119         winner = board[2][0]
120         return winner
121     return "No winner"
122 
123 def get_board_from_file(filename):
124     """
125     Get a tic tac toe board from a file
126     :param filename: A text file holding a tic tac
    toe board
127     :return: The board as a list
128     """
129     board_list = []
130     board_file = open(filename,"r")
131     for line in board_file:
132         board_list.append(line.strip())
133     board_file.close()
134     return board_list
135 
136 
137 def main():
138     """
139     1. Why is the logic inside badmain's main-line
    code ambiguous and hard to follow? Although the code
     inside
140     badmain works, it's hard to follow because it
    uses a cluster of global variables. The most
    confusing was
```

```
141     declaring a global "winner" variable as an empty
     string. Additionally, the check winner function is
    updating the
142     global variable and then returning nothing,
    making the code hard to read. Additionally, most of
    the functions
143     take parameters and just access a global board
    variable. The columns function was especially
    confusing since it
144     was returning a list called "to_return". You
    would have to go throgh the lines of code to
    actually decipher what
145     the function was returning, making the code
    harder to follow.
146
147     2. How does your refactoring remove this
    ambiguity? I removed all the global variables and
    passed parameters to
148     all the functions. This made it clear in my main
     function that I was performing certain actions on a
     certain
149     board. This also made my functions more reusable
     since they weren't accessing a global board
    variable. I changed
150     check winnner by getting rid of the global
    winner variable and made it so that the function
    would return the
151     actual winner of the board. If there was no
    winner, the function will return "No winner". This
    also makes my main
152     function easier to read because instead of
    winner being an empty string, the if statement will
    check if winner !=
153     "No winner." The empty string in badmain made
    that conditional very confusing. I also changed the
     "to_return"
154     list in the columns function to a list named "
    columns_of_board." This makes it clear to the person
     readng the
155     code as to what that function is returning.
    Additionally, all the main code is placed in a main
```

```
155 function and then
156     that function is called, eliminating all
    variables from the global scope.
157
158
159     """
160
161
162     board = get_board_from_file('input.txt')
163     print_board(board)
164     winner = check_winner(board)
165
166     if winner != 'No winner':
167         print(winner + ' WINS!!!!')
168     else:
169         print("TIE GAME!!!!")
170
171
172 if __name__ == "__main__":
173     main()
174
175
```