```
"""
Refactored version of badmain.py about tic-tac-toe game

:author: Chris Hegang Kim
:note: I affirm that I have carried out the attached academic endeavors with full academic honesty,
in accordance with the Union College Honor Code and the course syllabus.

:reflection: 1. Because it was on a global scope, some variables such as board and winner were used throughout the code
                without any orders and rules.
             2. I grouped the main-line into a single function called main(), and modified related codes when changing
                global variables to local variables such as removing globals and adding parameters and return statements.
"""
def print_board(board):
    """
    Prints a string for the board which is originally a list

    :param: the list for the board
    :return:
    """
    num_rows = len(board)
    num_cols = len(board[0])
    for row_num, row in enumerate(board):
        row_str = ''
        for col_num, marker in enumerate(row):
            row_str += marker
            if col_num < num_cols - 1:
                row_str += ' | '
        print(row_str)
        if row_num < num_rows - 1:
            print('---------')


def row_all_same(the_board, row):
    """
    Checks whether the row of the board is in all same
```

```
35 character
36
37     :param the_board: a list for the board
38     :param row: an integer for the index for each row
39     :return: True if the row is in all same character
40     """
41     return (the_board[row][0] == the_board[row][1] ==
   the_board[row][2])
42
43
44 def column_all_same(column):
45     """
46     Checks whether the column is in all same character
47
48     :param column: a string for each column
49     :return: True if the column is in all same character
50     """
51     return (column[0] == column[1] == column[2])
52
53
54 def diagonal_all_same(diagonal):
55     """
56     Checks whether the diagonal is in all same character
57
58     :param diagonal: a list with elements of the diagonal
59     :return: True if the diagonal is in all same character
60     """
61     return (diagonal[0] == diagonal[1] == diagonal[2])
62
63
64 def get_back_slash(board):
65     """
66     Returns a list with elements of the back-slash diagonal
67
68     :param: a list for the board
69     :return: a list with elements of the diagonal
70     """
71     return [board[i][i] for i in range(len(board))]
72
73
74 def get_forward_slash(board):
```

```
75      """
76      Returns a list with elements of the forward-slash
    diagonal
77
78      :param: a list for the board
79      :return: a list with elements of the diagonal
80      """
81      return [board[len(board)-i-1][i] for i in range(len(board
    ))]
82
83
84  def columns(board):
85      """
86      Returns a list with elements for columns on the board
87
88      :param board: a list for the board
89      :return: a list with elements for columns on the board
90      """
91      num_cols = len(board[0])
92      num_rows = len(board)
93
94      to_return = []
95
96      for i in range(num_cols):
97          col_str = ''
98          for j in range(num_rows):
99              col_str += board[j][i]
100         to_return.append(col_str)
101     return to_return
102
103
104 def check_winner(board):
105     """
106     Checks possible winning scenarios and determines the
    winner
107
108     :param: a list for the board
109     :return: a string for the winner based on possible
    scenarios
110     """
111     for row_num, row in enumerate(board):
```

```
112         if row_all_same(board, row_num):
113             winner = board[row_num][0]
114             return winner
115 
116     for col in columns(board):
117         if column_all_same(col):
118             winner = col[0]
119             return winner
120 
121     if diagonal_all_same(get_back_slash()):
122         winner = board[0][0]
123         return winner
124 
125     if diagonal_all_same(get_forward_slash()):
126         winner = board[2][0]
127         return winner
128 
129 
130 def get_board_from_file(filename):
131     """
132     Get a board from the file and creates a list for the
    board
133 
134     :param filename: a file for the board
135     :return: a list for the board
136     """
137     board_list = []
138     board_file = open(filename,"r")
139     for line in board_file:
140         board_list.append(line.strip())
141     board_file.close()
142     return board_list
143 
144 def main():
145     """
146     Starts the entire program and prints the winner
147 
148     :return:
149     """
150     inputfile = 'input.txt'
151     board = get_board_from_file(inputfile)
```

```
152
153     print_board(board)
154
155     winner = check_winner(board)
156
157     if winner ≠ '':
158         print(winner + ' WINS!!!!')
159     else:
160         print("TIE GAME!!!!")
161
162 if __name__ == "__main__":
163     main()
164
```