```
"""
Ian Sulley

A module for receiving a completed tic-tac-toe board in a txt file and evaluating the winner

Honor Code Statement:
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.

Comparing goodmain vs badmain:

badmain was written in a way that made the order of steps taken to solve a board confusing. because it was using global
variables the functions being called did not clearly reference the variables they were using. This made it hard to
follow what variables were being accessed in each function and how changing a global variable might affect the output of
the function. In goodmain, main() follows a clear and specific order. it takes a file and makes a board from it, then it
takes that board and evaluates a winner, then it prints the results. It much easier to see how the different variables
rely on each other when it is written this way. by changing the variables from a global scope and instead passing them
to each function as an argument, it is clear exactly what variable is being used in each function.
"""


def print_board(board):
    """
    given a board in list form, iterates though each part and prints it in a readable way

    :param board: list of 3 strings, each string a row in the board
    :return: prints the given board to the terminal
    """
```

```
29      num_rows = len(board)
30      num_cols = len(board[0])
31      for row_num, row in enumerate(board):
32          row_str = ''
33          for col_num, marker in enumerate(row):
34              row_str += marker
35              if col_num < num_cols - 1:
36                  row_str += ' | '
37          print(row_str)
38          if row_num < num_rows - 1:
39              print('---------')
40
41
42  def row_all_same(board, row):
43      """
44      given a board and a row, determines if each value in a
     row are equal or not and returns True or False
45
46      :param board: the board whos row is being tested
47      :param row: the row being tested
48      :return: True if all values are equal, False otherwise
49      """
50      return board[row][0] == board[row][1] == board[row][2]
51
52
53  def column_all_same(column):
54      """
55      given a column, checks if each value in a column are
    equal or not, returns True or False
56
57      :param column: list of strings, each string containing
     the contents of a board column
58      :return: True if all column values are equal, false
    otherwise
59      """
60      return column[0] == column[1] == column[2]
61
62
63  def diagonal_all_same(diagonal):
64      """
65      given a diagonal, checks if each value in a diagonal
    are equal or not, returns True or False
66
67      :param diagonal: list of strings, each string
    containing a diagonal value
68      :return: True if all diagonal values are equal, false
```

```
 68 otherwise
 69     """
 70     return diagonal[0] == diagonal[1] == diagonal[2]
 71
 72
 73 def get_back_slash(board):
 74     """
 75     given a board, gets diagonal values from upper left
    to bottom right and returns each value
 76
 77     :param board: board from which diagonal is being
    retrieved
 78     :return: value of each diagonal tile
 79     """
 80     return [board[i][i] for i in range(len(board))]
 81
 82
 83 def get_forward_slash(board):
 84     """
 85     gets diagonal values from board from bottom left to
    upper right
 86
 87     :param board: board which diagonal is being retrieved
     from
 88     :return: value of each diagonal tile
 89     """
 90
 91     return [board[len(board) - i - 1][i] for i in range(
    len(board))]
 92
 93
 94 def columns(board):
 95     """
 96     given a board, separates each column into a list
 97
 98     :param board: board columns are being retrieved from
 99     :return: list of values in a column for each column
100     """
101     num_cols = len(board[0])
102     num_rows = len(board)
103
104     to_return = []
105
106     for i in range(num_cols):
107         col_str = ''
108         for j in range(num_rows):
```

```
109             col_str += board[j][i]
110         to_return.append(col_str)
111     return to_return
112 
113 
114 def check_winner(board):
115     """
116     given a board, determines the winner and returns the
    winner
117 
118     :param board: board having winner evaluated
119     :return: winner (X, O, or '')
120     """
121     for row_num, row in enumerate(board):
122         if row_all_same(board, row_num):
123             winner = board[row_num][0]
124             return winner
125 
126     for col in columns(board):
127         if column_all_same(col):
128             winner = col[0]
129             return winner
130 
131     if diagonal_all_same(get_back_slash(board)):
132         winner = board[0][0]
133         return winner
134 
135     if diagonal_all_same(get_forward_slash(board)):
136         winner = board[2][0]
137         return winner
138 
139     else:
140         winner = ''
141         return winner
142 
143 
144 def get_board_from_file(filename):
145     """
146     given a txt file, generates a board, represents board
     as a list of strings and returns it
147 
148     :param filename: file being converted
149     :return: board (list of strings)
150     """
151 
152     board_list = []
```

```
153     board_file = open(filename, "r")
154     for line in board_file:
155         board_list.append(line.strip())
156     board_file.close()
157     return board_list
158 
159 
160 def print_winner(winner):
161     """
162     given a winner, prints the winner with proper
    formatting
163 
164     :param winner: winner being printed
165     :return: if there is a winner prints '(name of winner
    ) wins!' else prints 'tie game'
166     """
167     if winner != '':
168         print(winner + ' WINS!!!!')
169     else:
170         print("TIE GAME!!!!")
171 
172 
173 def main():
174     """
175     main calls all processes necessary for converting a
    txt file into a board.
176     Evaluates and prints board and its winner
177 
178     :return: Prints board and the winner
179     """
180 
181     inputfile = 'input.txt'  # assign txt file to a
    variable
182 
183     board = get_board_from_file(inputfile)  # create a
    board from that txt file
184 
185     print_board(board)  # print the board
186 
187     winner = check_winner(board)  # determine the winner
    of the board
188 
189     print_winner(winner)  # print the winner
190 
191 
192 if __name__ == '__main__':
```

```
193     main()
194
```