

# Tic Tac Toe Project, Fall 2025

Tong Yi

## Contents

<b>1</b>	<b>CopyRight Warning</b>	<b>1</b>
<b>2</b>	<b>File Structures</b>	<b>1</b>
2.1	TicTacToeBoard.hpp . . . . .	2
2.2	Header File of TicTacToeGame Class: TicTacToeGame.hpp . . . . .	4
2.3	Source Code to Test Tic Tac Toe Game: TestTicTacToeGame.cpp . . . . .	5
2.4	makefile . . . . .	5
2.5	Sample Run . . . . .	7
<b>3</b>	<b>Task A: Define Constructors, clear, and to_string methods of TicTacToeBoard.cpp</b>	<b>13</b>
3.1	Implement TicTacToeBoard.cpp . . . . .	13
3.2	Hint: help function to Draw Separate Line in to_string method . . . . .	13
3.3	Test TicTacToeBoard.cpp Locally . . . . .	13

## 1 CopyRight Warning

1. This is copyrighted materials; you are not allowed to upload to the Internet.
2. Our project is different from similar products in Internet.
  - (a) Ask help only from teaching staff of this course.
  - (b) Use solutions from artificial intelligence like ChatGPT or online tutoring websites like, but not limited to, chegg.com, violates academic integrity and is not allowed.

## 2 File Structures

1. Provide the following files
  - (a) Header file of TicTacToeBoard: [TicTacToeBoard.hpp](#)
  - (b) Header file of TicTacToeGame: [TicTacToeGame.hpp](#)
  - (c) Automate the process of building executable programs and other files from source code: [makefile](#)
  - (d) Source code to test TicTacToeGame class: [TestTicTacToeGame.cpp](#)
  - (e) Source code to test important methods of TicTacToeBoard class: [TestTicTacToeBoard.cpp](#)

2. You implement TicTacToeBoard and TicTacToeGame classes, that is,

(a) Define TicTacToeBoard.cpp.

(b) Define TicTacToeGame.cpp.

## 2.1 TicTacToeBoard.hpp

The contents of header file TicTacToeBoard is as follows.

```
1 #ifndef TicTacToeBoard_H
2 #define TicTacToeBoard_H
3 #include <iostream> //std
4 #include <vector>
5
6 class TicTacToeBoard {
7 public:
8     //a default 3x3 TicTacToe board
9     TicTacToeBoard();
10
11     //If the provided size is less than 3, default to creating a 3x3 board.
12     //Otherwise, initialize the board with the given size for both rows and
13     //columns.
14     TicTacToeBoard(int givenSize);
15
16     //Set each element of the board to be a space (' ').
17     void clear();
18
19     //Return the value at cell (row, col)
20     char getValue(int row, int col) const;
21
22     // Checks if the cell at (row, col) is available.
23     //A cell is available if its value is a space character (' ').
24     bool isAvailable(int row, int col) const;
25
26     //Check whether the given parameter row is valid row index or not.
27     //that is, whether row is in [0, board.size()-1]
28     bool isValidRow(int row) const;
29
30     //Check whether the given parameter col is a valid column index or not
31     //that is, whether col is in [0, board.size()-1]
32     bool isValidCol(int col) const;
33
34     //Return the size of tic tac toe board
35     int size() const;
36
37     //Place playerId at board[row][col].
38     void mark(int row, int col, char playerId);
```

```

38 //Returns a string that visually represents the current layout of the
39 board's elements.
40 std::string to_string() const;
41
42 //If every single row, column, main diagonal, AND anti-diagonal
43 //each contain both 'X' and 'O' symbols,
44 //then the board is in a tie state (return true).
45 //Otherwise, if any of those lines (row/col/diagonals) is missing
46 //an 'X' or an 'O', it means a win is still possible (return false).
47 bool tie() const;
48
49 //Check whether the player at row and col wins.
50 //If there is any win in horizontal, vertical,
51 //diagonal, or anti-diagonal direction, return true,
52 //otherwise, return false.
53 bool win(int row, int col) const;
54
55 //Check Horizontal Win:
56 //Evaluate the row containing the cell (row, col).
57 //If the current player's symbol forms a continuous,
58 //unbroken sequence of the required length within this row,
59 //the function returns true (win found); otherwise, it returns false.
60 //Check whether the player at (row, col) can win that row or not.
61 bool winByRow(int row, int col) const;
62 //Check Vertical Win:
63 //Evaluate the column containing the cell (row, col).
64 //If the current player's symbol forms a continuous,
65 //unbroken sequence of the required length within this column,
66 //the function returns true (win found); otherwise, it returns false.
67 bool winByCol(int row, int col) const;
68 //Check Dagonal (including both diagonal and anti-diagonal) Win:
69 //If the cell (row, col) is not in diagonal or anti-diagonal, return
    false.
70 //If the cell (row, col) is in the diagonal,
71 //evaluate the diagonal containing the cell (row, col).
72 //If the current player's symbol forms a continuous,
73 //unbroken sequence of the required length within this diagonal,
74 //the function returns true (win found);
75 //otherwise, evaluate the anti-diagonal containing the cell (row, col).
76 //If the current player's symbol forms a continuous,
77 //unbroken sequence of the required length within this anti-diagonal,
78 //the function returns true (win found); otherwise, return false.
79 bool winByDiagonal(int row, int col) const;
80
81 private:

```

```

82     std::vector<std::vector<char>> board; //board is a 2d array of chars
83 };
84 #endif

```

## 2.2 Header File of TicTacToeGame Class: TicTacToeGame.hpp

```

1  #ifndef TicTacToeGame_H
2  #define TicTacToeGame_H
3  #include "TicTacToeBoard.hpp"
4  class TicTacToeGame {
5  public:
6      TicTacToeGame();
7      TicTacToeGame(int size);
8      void runRepeat();
9      //start to play the game
10     void start();
11     bool isGameOver() const;
12
13     //This is how the user plays:
14     //Enter row and col such that
15     //1. row is in [0, size-1]
16     //2. col is in [0, size-1]
17     //3. the corresponding cell in the board is available
18     // (hint: call board.getValue(row, col) to check the return is 0 or
19     not).
20     //As long as the input row or col is not valid
21     //begin
22     // prompt what error(s) are, for example,
23     // * row is not in [0, size-1]
24     // * col is not in [0, size-1]
25     // * the corresponding cell in board is not available
26     // prompt user to re-enter.
27     //end
28     //
29     //Once we exit the above repetition loop,
30     //row and col are valid,
31     //mark the corresponding cell in the board by HUMAN_ID.
32     void humanPlay();
33
34     //computer play
35     //Computer checks the board from the first row to the last row.
36     //In each row, the computer checks from the first column to the last
37     column.
38     //A more sophisticated approach is to use "mark first; if unfit, then
39     remove mark"

```

```

37 //1. Try to win first.
38 // Mark an available cell by computerId,
39 // if this leads to win by computer,
40 // take this cell and return,
41 // otherwise, do not take this cell (that is, set this cell to be
   available).
42 //2. Try to block the opponent from winning.
43 // This approach is adopted after the try-to-win approach fails.
44 // Mark an available cell by userId (that is, suppose this cell is
   taken by userId),
45 // if this leads to win by user,
46 // mark this cell by computerId, and return.
47 // otherwise, do not take this cell (that is, set this cell to be
   available).
48 //3. If neither one of the above two approaches works
49 // (that is, the computer does not take a cell yet),
50 // then mark the first available cell.
51 void computerPlay();
52
53 private:
54     TicTacToeBoard tttBoard;
55     int currRow;
56     int currCol; //the row and col the current player chooses.
57     static const char HUMAN_ID = 'X';
58     static const char COMPUTER_ID = 'O';
59 };
60 #endif

```

## 2.3 Source Code to Test Tic Tac Toe Game: TestTicTacToeGame.cpp

```

1 #include "TicTacToeGame.hpp"
2
3 int main() {
4     TicTacToeGame game; //3 x 3 tic tac toe board
5     //TicTacToeGame game(4); //4 x 4 tic tac toe board
6     game.runRepeat();
7 }

```

## 2.4 makefile

When working with several source files for this project, it is highly recommended to use a **makefile** and the **make** utility. This automates the build process, eliminating the need to manually remember and type complex compilation commands for every file. A key advantage of using a Makefile is its efficient dependency tracking: if a change is made to a single source file, the **make** utility intelligently recompiles

only that specific file and then re-links the final executable, significantly reducing build times by avoiding unnecessary recompilation of unmodified files.

```
1 # This is an example Makefile for tic tac toe project. This
2 # program uses TicTacToeBoard, TicTacToeGame, and TestTicTacToeGame modules
3 # Typing 'make' or 'make run' will create the executable file.
4 #
5
6 # define some Makefile variables for the compiler and compiler flags
7 # to use Makefile variables later in the Makefile: $( )
8 #
9 # -g adds debugging information to the executable file
10 # -Wall turns on most, but not all, compiler warnings
11 #
12 # for C++ define CC = g++
13 CC = g++ -std=c++11
14 #CFLAGS = -g -Wall
15
16 # typing 'make' will invoke the first target entry in the file
17 # (in this case the default target entry)
18 # you can name this target entry anything, but "default" or "all"
19 # are the most commonly used names by convention
20 #
21 all: run
22
23 # To create the executable file run we need the object files
24 # TestTicTacToeGame.o, TicTacToeBoard.o
25 run: TicTacToeBoard.o TestTicTacToeGame.o TicTacToeGame.o
26     $(CC) -o run TestTicTacToeGame.o TicTacToeGame.o TicTacToeBoard.o
27
28 # To create the object file TestTicTacToeGame.o, we need the source
29 # files TestTicTacToeGame.cpp and TicTacToeBoard.hpp
30 #
31 TestTicTacToeGame.o: TestTicTacToeGame.cpp
32     $(CC) -c TestTicTacToeGame.cpp
33
34 TicTacToeGame.o: TicTacToeGame.cpp
35     $(CC) -c TicTacToeGame.cpp
36
37 # To create the object file TicTacToeBoard.o, we need the source
38 # files TicTacToeBoard.cpp, TicTacToeBoard.hpp
39 TicTacToeBoard.o: TicTacToeBoard.cpp
40     $(CC) -c TicTacToeBoard.cpp
41
42 # To start over from scratch, type 'make clean'. This
43 # removes the executable file, as well as old .o object
```

```

44 # files and *~ backup files:
45 #
46 clean:
47     $(RM) run *.o *~

```

To use makefile, just type

```

1 make

```

From `-o run` of Line 26 of the above make file, we notice that the runnable file is called `run`. Then run the project using

```

1 ./run

```

When you want to delete all the temporary files generated during the build process (such as object files `.o`, intermediate data files, and the final executable itself). Run the following command.

```

1 make clean

```

## 2.5 Sample Run

```

1      0    1    2
2      +---+---+---+
3  0 |   |   |   |
4      +---+---+---+
5  1 |   |   |   |
6      +---+---+---+
7  2 |   |   |   |
8      +---+---+---+
9
10 Round 1: User, enter row and col to place X: 0 0
11      0    1    2
12      +---+---+---+
13  0 | X |   |   |
14      +---+---+---+
15  1 |   |   |   |
16      +---+---+---+
17  2 |   |   |   |
18      +---+---+---+
19 Round 2: Computer places 0 at row 0 and col 1.
20      0    1    2
21      +---+---+---+
22  0 | X | 0 |   |
23      +---+---+---+
24  1 |   |   |   |
25      +---+---+---+
26  2 |   |   |   |
27      +---+---+---+

```

```

28 Round 3: User, enter row and col to place X: 1 1
29     0   1   2
30     +---+---+---+
31     0 | X | 0 |   |
32     +---+---+---+
33     1 |   | X |   |
34     +---+---+---+
35     2 |   |   |   |
36     +---+---+---+
37 Round 4: Computer places 0 at row 2 and col 2.
38     0   1   2
39     +---+---+---+
40     0 | X | 0 |   |
41     +---+---+---+
42     1 |   | X |   |
43     +---+---+---+
44     2 |   |   | 0 |
45     +---+---+---+
46 Round 5: User, enter row and col to place X: 2 0
47     0   1   2
48     +---+---+---+
49     0 | X | 0 |   |
50     +---+---+---+
51     1 |   | X |   |
52     +---+---+---+
53     2 | X |   | 0 |
54     +---+---+---+
55 Round 6: Computer places 0 at row 0 and col 2.
56     0   1   2
57     +---+---+---+
58     0 | X | 0 | 0 |
59     +---+---+---+
60     1 |   | X |   |
61     +---+---+---+
62     2 | X |   | 0 |
63     +---+---+---+
64 Round 7: User, enter row and col to place X: 1 0
65     0   1   2
66     +---+---+---+
67     0 | X | 0 | 0 |
68     +---+---+---+
69     1 | X | X |   |
70     +---+---+---+
71     2 | X |   | 0 |
72     +---+---+---+
73 Human wins. Yay!!!

```



```

74 Do you want to continue (yes/no): y
75     0     1     2
76 +---+---+---+
77 0 |   |   |   |
78 +---+---+---+
79 1 |   |   |   |
80 +---+---+---+
81 2 |   |   |   |
82 +---+---+---+
83
84 Round 1: User, enter row and col to place X: 0 0
85     0     1     2
86 +---+---+---+
87 0 | X |   |   |
88 +---+---+---+
89 1 |   |   |   |
90 +---+---+---+
91 2 |   |   |   |
92 +---+---+---+
93 Round 2: Computer places 0 at row 0 and col 1.
94     0     1     2
95 +---+---+---+
96 0 | X | 0 |   |
97 +---+---+---+
98 1 |   |   |   |
99 +---+---+---+
100 2 |   |   |   |
101 +---+---+---+
102 Round 3: User, enter row and col to place X: 1 0
103     0     1     2
104 +---+---+---+
105 0 | X | 0 |   |
106 +---+---+---+
107 1 | X |   |   |
108 +---+---+---+
109 2 |   |   |   |
110 +---+---+---+
111 Round 4: Computer places 0 at row 2 and col 0.
112     0     1     2
113 +---+---+---+
114 0 | X | 0 |   |
115 +---+---+---+
116 1 | X |   |   |
117 +---+---+---+
118 2 | 0 |   |   |
119 +---+---+---+

```

```

120 Round 5: User, enter row and col to place X: 0 2
121     0   1   2
122     +---+---+---+
123     0 | X | 0 | X |
124     +---+---+---+
125     1 | X |   |   |
126     +---+---+---+
127     2 | 0 |   |   |
128     +---+---+---+
129 Round 6: Computer places 0 at row 1 and col 1.
130     0   1   2
131     +---+---+---+
132     0 | X | 0 | X |
133     +---+---+---+
134     1 | X | 0 |   |
135     +---+---+---+
136     2 | 0 |   |   |
137     +---+---+---+
138 Round 7: User, enter row and col to place X: 1 2
139     0   1   2
140     +---+---+---+
141     0 | X | 0 | X |
142     +---+---+---+
143     1 | X | 0 | X |
144     +---+---+---+
145     2 | 0 |   |   |
146     +---+---+---+
147 Round 8: Computer places 0 at row 2 and col 1.
148     0   1   2
149     +---+---+---+
150     0 | X | 0 | X |
151     +---+---+---+
152     1 | X | 0 | X |
153     +---+---+---+
154     2 | 0 | 0 |   |
155     +---+---+---+
156 Computer wins. Yuck.
157 Do you want to continue (yes/no): y
158     0   1   2
159     +---+---+---+
160     0 |   |   |   |
161     +---+---+---+
162     1 |   |   |   |
163     +---+---+---+
164     2 |   |   |   |
165     +---+---+---+

```

```

166
167 Round 1: User, enter row and col to place X: 1 1
168     0   1   2
169     +---+---+---+
170 0 |   |   |   |
171     +---+---+---+
172 1 |   | X |   |
173     +---+---+---+
174 2 |   |   |   |
175     +---+---+---+
176 Round 2: Computer places O at row 0 and col 0.
177     0   1   2
178     +---+---+---+
179 0 | O |   |   |
180     +---+---+---+
181 1 |   | X |   |
182     +---+---+---+
183 2 |   |   |   |
184     +---+---+---+
185 Round 3: User, enter row and col to place X: 0 2
186     0   1   2
187     +---+---+---+
188 0 | O |   | X |
189     +---+---+---+
190 1 |   | X |   |
191     +---+---+---+
192 2 |   |   |   |
193     +---+---+---+
194 Round 4: Computer places O at row 2 and col 0.
195     0   1   2
196     +---+---+---+
197 0 | O |   | X |
198     +---+---+---+
199 1 |   | X |   |
200     +---+---+---+
201 2 | O |   |   |
202     +---+---+---+
203 Round 5: User, enter row and col to place X: 1 0
204     0   1   2
205     +---+---+---+
206 0 | O |   | X |
207     +---+---+---+
208 1 | X | X |   |
209     +---+---+---+
210 2 | O |   |   |
211     +---+---+---+

```

```

212 Round 6: Computer places 0 at row 1 and col 2.
213     0   1   2
214     +---+---+---+
215 0 | 0 |   | X |
216     +---+---+---+
217 1 | X | X | 0 |
218     +---+---+---+
219 2 | 0 |   |   |
220     +---+---+---+
221 Round 7: User, enter row and col to place X: 0 1
222     0   1   2
223     +---+---+---+
224 0 | 0 | X | X |
225     +---+---+---+
226 1 | X | X | 0 |
227     +---+---+---+
228 2 | 0 |   |   |
229     +---+---+---+
230 Round 8: Computer places 0 at row 2 and col 1.
231     0   1   2
232     +---+---+---+
233 0 | 0 | X | X |
234     +---+---+---+
235 1 | X | X | 0 |
236     +---+---+---+
237 2 | 0 | 0 |   |
238     +---+---+---+
239 Round 9: User, enter row and col to place X: 0 1
240 The square you pick up is not available.User, re-enter row and col to place
    X: 1 0
241 The square you pick up is not available.User, re-enter row and col to place
    X: 2 1
242 The square you pick up is not available.User, re-enter row and col to place
    X: 2 2
243     0   1   2
244     +---+---+---+
245 0 | 0 | X | X |
246     +---+---+---+
247 1 | X | X | 0 |
248     +---+---+---+
249 2 | 0 | 0 | X |
250     +---+---+---+
251 It is a tie.
252 Do you want to continue (yes/no): no
253 Bye

```

## 3 Task A: Define Constructors, clear, and to\_string methods of TicTacToeBoard.cpp

1. Create a file named TicTacToeBoard.cpp.
2. Implement methods defined in TicTacToeBoard.hpp.
3. Test TicTacToeBoard.cpp locally using provided TestTicTacToeBoard.cpp.
4. Submit TicTacToeBoard.cpp to gradescope for grading.

### 3.1 Implement TicTacToeBoard.cpp

Put TicTacToeBoard.hpp, TicTacToeBoard.cpp, and TestTicTacToeBoard.cpp in the same directory.

You need to implement the corresponding source code, that is, define TicTacToeBoard.cpp.

### 3.2 Hint: help function to Draw Separate Line in to\_string method

```
1 //separate lines of data in board
2 //This function is NOT a member function,
3 //since clients of TicTacToeBoard class do not need to use it.
4 //Only method to_string needs to use it.
5 //Since this is not a method of TicTacToeBoard class,
6 //we need to pass parameter size.
7 std::string separateLine(int size) {
8     std::string str = "  +";
9     for (int i = 0; i < size; i++) {
10         str += "---+";
11     }
12     str += "\n";
13     return str;
14 }
```

### 3.3 Test TicTacToeBoard.cpp Locally

Use the following TestTicTacToeBoard.cpp to test your TicTacToeBoard.cpp.

```
1 #include <iostream>
2 #include <vector>
3 #include "TicTacToeBoard.hpp"
4 //g++ -o tBoard TicTacToeBoard.cpp TestTicTacToeBoard.cpp
5
6 //test default constructor using
7 //./tBoard A
8
9 //test non-default constructor using
10 //./tBoard B
```

```

11
12 //test clear method using
13 //./tBoard C
14 //and so on.
15
16 const int NUM_COLUMNS = 4;
17 TicTacToeBoard* assignData(char data[][NUM_COLUMNS]);
18
19 int main(int argc, const char *argv[]) {
20     if (argc != 2) {
21         std::cout << "Need 'A'-'J' in command line parameter" << std::endl;
22         return -1;
23     }
24
25     //unit-testing for constructors and the destructor
26     char type = *argv[1];
27     std::string prompt;
28     TicTacToeBoard *tttBoard;
29
30     if (type == 'A') {
31         prompt = "default constructor TicTacToeBoard board;";
32         tttBoard = new TicTacToeBoard;
33
34         //expected output:
35         //Call default constructor TicTacToeBoard board;
36         //contents of board is
37         // , , ,
38         // , , ,
39         // , , ,
40         //
41     }
42     else if (type == 'B') {
43         std::cout << "Enter size of the board: ";
44         int size;
45         std::cin >> size;
46         prompt = "non-default constructor TicTacToeBoard board(" + std::
            to_string(size) + ");";
47         tttBoard = new TicTacToeBoard(size);
48
49         //sample input/output:
50         //Enter size of the board: 5
51         //Call non-default constructor TicTacToeBoard board(5);
52         //contents of board is
53         // , , , , ,
54         // , , , , ,
55         // , , , , ,

```

```

56 // , , , , ,
57 // , , , , ,
58 //
59
60 //do not take the parameter as it is,
61 //need to make sure that parameter size is >= 3
62 //another sample input/output:
63 //Enter size of the board: 2
64 //Call non-default constructor TicTacToeBoard board(2);
65 //contents of board is
66 // , , ,
67 // , , ,
68 // , , ,
69 //
70 }
71 else if (type == 'C' || type == 'D') {
72     //test clear method,
73     char data[][NUM_COLUMNS] = {
74         {'X', 'O', 'X', ' '},
75         {'O', 'X', ' ', ' '},
76         {' ', 'O', 'O', ' '},
77         {'X', ' ', ' ', 'X'}},
78 };
79
80 tttBoard = assignData(data);
81
82 if (type == 'C') {
83     //test clear method
84     tttBoard->clear();
85
86     //after calling clearing method, each element of board should be
87     ' ',
88
89     bool isWrong = false;
90     for (int row = 0; row < tttBoard->size() && !isWrong; row++) {
91         for (int col = 0; col < tttBoard->size() && !isWrong; col++)
92         {
93             if (tttBoard->getValue(row, col) != ' ') {
94                 std::cout << "clear method is not correct. Each
95                 element should be a space character\n";
96                 isWrong = true; //set a tag
97                 //break; //break only can break the inner loop, the
98                 outer loop still runs
99             }
100         }
101     }
102 }

```

```

98         if (!isWrong) {
99             std::cout << "clear method is correct\n";
100         }
101         //expected output:
102         //clear method is correct
103     }
104     else if (type == 'D') {
105         //test to_string method
106         std::cout << tttBoard->to_string();
107         //expected output:
108         //      0   1   2   3
109         //  +---+---+---+---+
110         // 0 | X | 0 | X |   |
111         //  +---+---+---+---+
112         // 1 | 0 | X |   |   |
113         //  +---+---+---+---+
114         // 2 |   | 0 | 0 |   |
115         //  +---+---+---+---+
116         // 3 | X |   |   | X |
117         //  +---+---+---+---+
118     }
119 }
120 else if (type == 'E') {
121     //test winByRow(int row, int col)
122     char data[][NUM_COLUMNS] = {
123         {'X', 'O', 'X', ' '},
124         {'X', 'O', 'O', ' '},
125         {'O', 'O', 'O', 'O'},
126         {'X', 'X', 'X', ' '},
127     };
128
129     tttBoard = assignData(data);
130
131     bool result;
132     for (int row = 0; row < tttBoard->size(); row++) {
133         for (int col = 0; col < tttBoard->size(); col++) {
134             result = tttBoard->winByRow(row, col);
135             if (result) { //result == true
136                 std::cout << std::boolalpha << result;
137             }
138             std::cout << ',';
139             //std::cout << std::boolalpha << tttBoard->winByRow(row,
140                 col) << ',';
141         }
142         std::cout << std::endl;
143     }

```



```

143 //expected output:
144 //,,,
145 //,,,
146 //true,true,true,true,
147 //,,,
148 }
149 else if (type == 'F') {
150     //test winByCol(int row, int col)
151     char data[][NUM_COLUMNS] = {
152         {'O', 'X', 'O', ' '},
153         {'O', 'X', 'X', 'O'},
154         {'O', 'X', 'O', ' '},
155         {'X', 'X', ' ', 'X'},
156     };
157
158     tttBoard = assignData(data);
159
160     bool result;
161     for (int row = 0; row < tttBoard->size(); row++) {
162         for (int col = 0; col < tttBoard->size(); col++) {
163             result = tttBoard->winByCol(row, col);
164             if (result) { //result == true
165                 std::cout << std::boolalpha << result;
166             }
167             std::cout << ',';
168
169             //std::cout << std::boolalpha << tttBoard->winByCol(row,
170                 col) << ',';
171         }
172         std::cout << std::endl;
173     }
174     //expected output:
175     //,true,,,
176     //,true,,,
177     //,true,,,
178     //,true,,,
179     }
180     else if (type == 'G') {
181         //test winByDiagonal(int row, int col)
182         char data[][NUM_COLUMNS] = {
183             {'X', 'X', 'O', ' '},
184             {'O', 'X', 'O', 'O'},
185             {'X', 'O', 'X', ' '},
186             {'O', 'X', ' ', 'X'},
187         };

```

```

188     tttBoard = assignData(data);
189
190     bool result;
191     for (int row = 0; row < tttBoard->size(); row++) {
192         for (int col = 0; col < tttBoard->size(); col++) {
193             result = tttBoard->winByDiagonal(row, col);
194             if (result) { //result == true
195                 std::cout << std::boolalpha << result;
196             }
197             std::cout << ' ';
198
199             //std::cout << std::boolalpha << tttBoard->winByDiagonal(
200                 row, col) << ' ';
201         }
202         std::cout << std::endl;
203     }
204     //expected output:
205     //true,,,
206     //,true,,,
207     //,,true,,
208     //,,,true,
209     }
210     else if (type == 'H') {
211         //test winByDiagonal(int row, int col)
212         char data[][NUM_COLUMNS] = {
213             {'X', 'X', 'O', 'O'},
214             {'O', 'X', 'O', 'O'},
215             {'X', 'O', ' ', 'X'},
216             {'O', 'X', ' ', 'X'},
217         };
218
219         tttBoard = assignData(data);
220
221         bool result;
222         for (int row = 0; row < tttBoard->size(); row++) {
223             for (int col = 0; col < tttBoard->size(); col++) {
224                 result = tttBoard->winByDiagonal(row, col);
225                 if (result) { //result == true
226                     std::cout << std::boolalpha << result;
227                 }
228                 std::cout << ' ';
229             }
230             std::cout << std::endl;
231         }
232     }
233     //expected output:
234     //,,,true,

```

```

233 //,,true,,
234 //,true,,,
235 //true,,,,
236     }
237     else if (type == 'I') {
238         //test tie()
239         char data[][NUM_COLUMNS] = {
240             {'X', 'X', 'O', ' '},
241             {'O', 'X', 'O', 'O'},
242             {'X', 'O', 'X', ' '},
243             {'O', 'X', ' ', 'X'},
244         };
245
246         tttBoard = assignData(data);
247
248         std::cout << std::boolalpha << tttBoard->tie() << '\n';
249
250         //expected output:
251         //false
252     }
253     else if (type == 'J') {
254         //test tie()
255         char data[][NUM_COLUMNS] = {
256             {'X', 'X', 'O', ' '},
257             {'O', 'X', 'X', 'O'},
258             {'X', 'O', 'O', ' '},
259             {'O', 'X', ' ', 'X'},
260         };
261
262         tttBoard = assignData(data);
263
264         std::cout << std::boolalpha << tttBoard->tie() << '\n';
265
266         //expected output:
267         //true
268     }
269
270     // 'A' for default constructor and
271     // 'B' for non-default constructor
272     if (type == 'A' || type == 'B') {
273         std::cout << "Call " << prompt << '\n';
274         std::cout << "contents of board is\n";
275         for (int row = 0; row < tttBoard->size(); row++) {
276             for (int col = 0; col < tttBoard->size(); col++) {
277                 std::cout << tttBoard->getValue(row, col) << ',';
278             }

```

```

279         std::cout << '\n';
280     }
281 }
282
283     delete tttBoard; //release dynamic allocated memory
284     tttBoard = nullptr; //handle dangling pointer problem
285
286     return 0;
287 }
288
289 //TODO: need to define mark method of TicTacToeBoard.cpp before using
assignData function
290 TicTacToeBoard* assignData(char data[][NUM_COLUMNS]) {
291     TicTacToeBoard *tttBoard = new TicTacToeBoard(NUM_COLUMNS);
292
293     for (int row = 0; row < tttBoard->size(); row++) {
294         for (int col = 0; col < tttBoard->size(); col++) {
295             tttBoard->mark(row, col, data[row][col]); //set board[row][col]
of *tttBoard -- a TicTacToeBoard object -- to be data[row][
col]
296         }
297     }
298
299     return tttBoard;
300 }

```