Tic Tac Toe Project, Fall 2025

Tong Yi

Contents

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: **tttBoard_main.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.

```
#ifndef TicTacToeBoard_H
#define TicTacToeBoard_H
#include <iostream> //std
#include <vector>

class TicTacToeBoard {
public:
    //a default 3x3 TicTacToe board
    TicTacToeBoard();

    //If the provided size is less than 3, default to creating a 3x3 board.
    //Otherwise, initialize the board with the given size for both rows and
        columns.
    TicTacToeBoard(int givenSize);

    //Set each element of the board to be a space (' ').
    void clear();

    //Return the value at cell (row, col)
    char getValue(int row, int col) const;

    // Checks if the cell at (row, col) is available.
    //A cell is available if its value is a space character (' ').
    bool isAvailable(int row, int col) const;

    //Check whether the given parameter row is valid row index or not.
    //that is, whether row is in [0, board.size()-1]
    bool isValidRow(int row) const;

    //Check whether the given parameter col is a valid column index or not
    //that is, whether col is in [0, board.size()-1]
    bool isValidCol(int col) const;

    //Return the size of tic tac toe board
    int size() const;

    //Place playerId at board[row][col].
    void mark(int row, int col, char playerId);
```

```
38
39      //Returns a string that visually represents the current layout of the
            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 horizonal, 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 Digonal (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
            not).
19      //As long as the input row or col is not valid
20      //begin
21      //   prompt what error(s) are, for example,
22      //   * row is not in [0, size-1]
23      //   * col is not in [0, size-1]
24      //   * the corresponding cell in board is not available
25      //   prompt user to re-enter.
26      //end
27      //
28      //Once we exit the above repetition loop,
29      //row and col are valid,
30      //mark the corresponding cell in the board by HUMAN_ID.
31      void humanPlay();
32
33      //computer play
34      //Computer checks the board from the first row to the last row.
35      //In each row, the computer checks from the first column to the last
            column.
36      //A more sophisticated approach is to use "mark first; if unfit, then
            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.

```
# This is an example Makefile for tic tac toe project. This
# program uses TicTacToeBoard, TicTacToeGame, and TestTicTacToeGame modules
   .
# Typing 'make' or 'make run' will create the executable file.
#

# define some Makefile variables for the compiler and compiler flags
# to use Makefile variables later in the Makefile: $()
#
#  -g    adds debugging information to the executable file
#  -Wall turns on most, but not all, compiler warnings
#
# for C++ define  CC = g++
CC = g++ -std=c++11
#CFLAGS  = -g -Wall

# typing 'make' will invoke the first target entry in the file
# (in this case the default target entry)
# you can name this target entry anything, but "default" or "all"
# are the most commonly used names by convention
#
all: run

# To create the executable file run we need the object files
# TestTicTacToeGame.o, TicTacToeBoard.o
run:  TicTacToeBoard.o TestTicTacToeGame.o TicTacToeGame.o
    $(CC) -o run TestTicTacToeGame.o TicTacToeGame.o TicTacToeBoard.o

# To create the object file TestTicTacToeGame.o, we need the source
# files TestTicTacToeGame.cpp and TicTacToeBoard.hpp
#
TestTicTacToeGame.o:  TestTicTacToeGame.cpp
    $(CC) -c TestTicTacToeGame.cpp

TicTacToeGame.o:  TicTacToeGame.cpp
    $(CC) -c TicTacToeGame.cpp

# To create the object file TicTacToeBoard.o, we need the source
# files TicTacToeBoard.cpp, TicTacToeBoard.hpp
TicTacToeBoard.o:  TicTacToeBoard.cpp
    $(CC) -c TicTacToeBoard.cpp

# To start over from scratch, type 'make clean'.  This
# removes the executable file, as well as old .o object
```

```
# files and *~ backup files:
#
clean:
    $(RM) run *.o *~
```

To use makefile, just type

```
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

```
./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.

```
make clean
```

2.5 Sample Run

```
     0   1   2
   +---+---+---+
 0 |   |   |   |
   +---+---+---+
 1 |   |   |   |
   +---+---+---+
 2 |   |   |   |
   +---+---+---+

Round 1: User, enter row and col to place X: 0 0
     0   1   2
   +---+---+---+
 0 | X |   |   |
   +---+---+---+
 1 |   |   |   |
   +---+---+---+
 2 |   |   |   |
   +---+---+---+
Round 2: Computer places O at row 0 and col 1.
     0   1   2
   +---+---+---+
 0 | X | O |   |
   +---+---+---+
 1 |   |   |   |
   +---+---+---+
 2 |   |   |   |
   +---+---+---+
```

```
Round 3: User, enter row and col to place X: 1 1
     0   1   2
   +---+---+---+
 0 | X | O |   |
   +---+---+---+
 1 |   | X |   |
   +---+---+---+
 2 |   |   |   |
   +---+---+---+
Round 4: Computer places O at row 2 and col 2.
     0   1   2
   +---+---+---+
 0 | X | O |   |
   +---+---+---+
 1 |   | X |   |
   +---+---+---+
 2 |   |   | O |
   +---+---+---+
Round 5: User, enter row and col to place X: 2 0
     0   1   2
   +---+---+---+
 0 | X | O |   |
   +---+---+---+
 1 |   | X |   |
   +---+---+---+
 2 | X |   | O |
   +---+---+---+
Round 6: Computer places O at row 0 and col 2.
     0   1   2
   +---+---+---+
 0 | X | O | O |
   +---+---+---+
 1 |   | X |   |
   +---+---+---+
 2 | X |   | O |
   +---+---+---+
Round 7: User, enter row and col to place X: 1 0
     0   1   2
   +---+---+---+
 0 | X | O | O |
   +---+---+---+
 1 | X | X |   |
   +---+---+---+
 2 | X |   | O |
   +---+---+---+
Human wins. Yay!!!
```

```
Do you want to continue (yes/no): y
     0   1   2
   +---+---+---+
 0 |   |   |   |
   +---+---+---+
 1 |   |   |   |
   +---+---+---+
 2 |   |   |   |
   +---+---+---+

Round 1: User, enter row and col to place X: 0 0
     0   1   2
   +---+---+---+
 0 | X |   |   |
   +---+---+---+
 1 |   |   |   |
   +---+---+---+
 2 |   |   |   |
   +---+---+---+
Round 2: Computer places O at row 0 and col 1.
     0   1   2
   +---+---+---+
 0 | X | O |   |
   +---+---+---+
 1 |   |   |   |
   +---+---+---+
 2 |   |   |   |
   +---+---+---+
Round 3: User, enter row and col to place X: 1 0
     0   1   2
   +---+---+---+
 0 | X | O |   |
   +---+---+---+
 1 | X |   |   |
   +---+---+---+
 2 |   |   |   |
   +---+---+---+
Round 4: Computer places O at row 2 and col 0.
     0   1   2
   +---+---+---+
 0 | X | O |   |
   +---+---+---+
 1 | X |   |   |
   +---+---+---+
 2 | O |   |   |
   +---+---+---+
```

```
Round 5: User, enter row and col to place X: 0 2
     0   1   2
   +---+---+---+
 0 | X | O | X |
   +---+---+---+
 1 | X |   |   |
   +---+---+---+
 2 | O |   |   |
   +---+---+---+
Round 6: Computer places O at row 1 and col 1.
     0   1   2
   +---+---+---+
 0 | X | O | X |
   +---+---+---+
 1 | X | O |   |
   +---+---+---+
 2 | O |   |   |
   +---+---+---+
Round 7: User, enter row and col to place X: 1 2
     0   1   2
   +---+---+---+
 0 | X | O | X |
   +---+---+---+
 1 | X | O | X |
   +---+---+---+
 2 | O |   |   |
   +---+---+---+
Round 8: Computer places O at row 2 and col 1.
     0   1   2
   +---+---+---+
 0 | X | O | X |
   +---+---+---+
 1 | X | O | X |
   +---+---+---+
 2 | O | O |   |
   +---+---+---+
Computer wins. Yuck.
Do you want to continue (yes/no): y
     0   1   2
   +---+---+---+
 0 |   |   |   |
   +---+---+---+
 1 |   |   |   |
   +---+---+---+
 2 |   |   |   |
   +---+---+---+
```

```
Round 1: User, enter row and col to place X: 1 1
     0   1   2
   +---+---+---+
 0 |   |   |   |
   +---+---+---+
 1 |   | X |   |
   +---+---+---+
 2 |   |   |   |
   +---+---+---+
Round 2: Computer places O at row 0 and col 0.
     0   1   2
   +---+---+---+
 0 | O |   |   |
   +---+---+---+
 1 |   | X |   |
   +---+---+---+
 2 |   |   |   |
   +---+---+---+
Round 3: User, enter row and col to place X: 0 2
     0   1   2
   +---+---+---+
 0 | O |   | X |
   +---+---+---+
 1 |   | X |   |
   +---+---+---+
 2 |   |   |   |
   +---+---+---+
Round 4: Computer places O at row 2 and col 0.
     0   1   2
   +---+---+---+
 0 | O |   | X |
   +---+---+---+
 1 |   | X |   |
   +---+---+---+
 2 | O |   |   |
   +---+---+---+
Round 5: User, enter row and col to place X: 1 0
     0   1   2
   +---+---+---+
 0 | O |   | X |
   +---+---+---+
 1 | X | X |   |
   +---+---+---+
 2 | O |   |   |
   +---+---+---+
```

```
Round 6: Computer places O at row 1 and col 2.
     0   1   2
   +---+---+---+
 0 | O |   | X |
   +---+---+---+
 1 | X | X | O |
   +---+---+---+
 2 | O |   |   |
   +---+---+---+
Round 7: User, enter row and col to place X: 0 1
     0   1   2
   +---+---+---+
 0 | O | X | X |
   +---+---+---+
 1 | X | X | O |
   +---+---+---+
 2 | O |   |   |
   +---+---+---+
Round 8: Computer places O at row 2 and col 1.
     0   1   2
   +---+---+---+
 0 | O | X | X |
   +---+---+---+
 1 | X | X | O |
   +---+---+---+
 2 | O | O |   |
   +---+---+---+
Round 9: User, enter row and col to place X: 0 1
The square you pick up is not available.User, re-enter row and col to place
    X: 1 0
The square you pick up is not available.User, re-enter row and col to place
    X: 2 1
The square you pick up is not available.User, re-enter row and col to place
    X: 2 2
     0   1   2
   +---+---+---+
 0 | O | X | X |
   +---+---+---+
 1 | X | X | O |
   +---+---+---+
 2 | O | O | X |
   +---+---+---+
It is a tie.
Do you want to continue (yes/no): no
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 code locally using provided `tttBoard_main.cpp`.
4. Submit `TicTacToeBoard.cpp` to gradescope for grading.

3.1 Implement TicTacToeBoard.cpp

Put `TicTacToeBoard.hpp`, `TicTacToeBoard.cpp`, and `tttBoard_main.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

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

3.3 Test TicTacToeBoard.cpp Locally

Use the following `tttBoard_main.cpp` to test your TicTacToeBoard.cpp.

```
#include <iostream>
#include <vector>
#include "TicTacToeBoard.hpp"
//g++ -o tBoard TicTacToeBoard.cpp tttBoard_main.cpp

//test default constructor using
//./tBoard A

//test non-default constructor using
//./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            bool isWrong = false;
88            for (int row = 0; row < tttBoard->size() && !isWrong; row++) {
89                for (int col = 0; col < tttBoard->size() && !isWrong; col++)
                       {
90                    if (tttBoard->getValue(row, col) != ' ') {
91                       std::cout << "clear method is not correct. Each
                              element should be a space character\n";
92                       isWrong = true; //set a tag
93                       //break; //break only can break the inner loop, the
                              outer loop still runs
94                    }
95                }
96            }
97
```

```
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 | O | X |   |
111  //    +---+---+---+---+
112  // 1 | O | X |   |   |
113  //    +---+---+---+---+
114  // 2 |   | O | O |   |
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,
                        col) << ',';
140              }
141              std::cout << std::endl;
142          }
```

```
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,
                       col) << ',';
170             }
171             std::cout << std::endl;
172         }
173 //expected output:
174 //,true,,,
175 //,true,,,
176 //,true,,,
177 //,true,,,
178     }
179     else if (type == 'G') {
180         //test winByDiagonal(int row, int col)
181         char data[][NUM_COLUMNS] = {
182            {'X', 'X', 'O', ' '},
183            {'O', 'X', 'O', 'O'},
184            {'X', 'O', 'X', ' '},
185            {'O', 'X', ' ', 'X'},
186         };
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(
                       row, col) << ',';
200             }
201             std::cout << std::endl;
202         }
203 //expected output:
204 //true,,,,
205 //,true,,,
206 //,,true,,
207 //,,,true,
208     }
209     else if (type == 'H') {
210         //test winByDiagonal(int row, int col)
211         char data[][NUM_COLUMNS] = {
212            {'X', 'X', 'O', 'O'},
213            {'O', 'X', 'O', 'O'},
214            {'X', 'O', ' ', 'X'},
215            {'O', 'X', ' ', 'X'},
216         };
217 
218         tttBoard = assignData(data);
219 
220         bool result;
221         for (int row = 0; row < tttBoard->size(); row++) {
222             for (int col = 0; col < tttBoard->size(); col++) {
223                 result = tttBoard->winByDiagonal(row, col);
224                 if (result) { //result == true
225                    std::cout << std::boolalpha << result;
226                 }
227                 std::cout << ',';
228             }
229             std::cout << std::endl;
230         }
231 //expected output:
232 //,,,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 TicTacToeBoard* assignData(char data[][NUM_COLUMNS]) {
290    TicTacToeBoard *tttBoard = new TicTacToeBoard(NUM_COLUMNS);
291
292    for (int row = 0; row < tttBoard->size(); row++) {
293        for (int col = 0; col < tttBoard->size(); col++) {
294            tttBoard->mark(row, col, data[row][col]); //set board[row][col]
                   of *tttBoard -- a TicTacToeBoard object -- to be data[row][
                   col]
295        }
296    }
297
298    return tttBoard;
299 }
```