Basic knowledge of classes, functions, loops, strings, vectors, references, and unit tests.  
  
Problem Domain:

The classic TicTacToe game alternatively place Xs and Os on a 3x3 grid. The winner is the first player to place 3 consecutive marks in a horizontal, vertical or diagonal row.

Understanding the Problem:  
1) Determine a winner by row, column, diagonally, or a tie.  
2) Display winner of game in main.  
3) Create main.cpp game logic to:  
     a) start game   
     b) play game  
     c) output board after each mark board  
     d) display winner of game

**Perequisite:**

Homework 5 **Completed**

Homework 6 **Completed**  
  
Understanding the Problem:  
A tic tac toe board is usually as follows :  
  
How users see the board

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| 4 | 5 | 6 |
| 7 | 8 | 9 |

How a programmer sees the board

|  |  |  |
| --- | --- | --- |
| 0 | 1 | 2 |
| 3 | 4 | 5 |
| 6 | 7 | 8 |

In this program, use a vector of strings to represent the board as follows where value can be " ", O, or X:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Value |  |  |  |  |  |  |  |  |  |
| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

**How to determine if a player has won?**A column wins with marked values 1,4,7 or 2,5,8, or 3,6,9 with all Os or Xs  (**Remember a vector index starts at 0**)  
  
Example: Win by first column where X goes first  
  
TicTacToe board

|  |  |  |
| --- | --- | --- |
| X | O |  |
| X | O |  |
| X |  |  |

Vector view (Remember a user will see 1,4 and 7 as a column win.  But the vector will store the values at 0,3, and 6):

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Values | X | O |  | X | O |  | X |  |  |
| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

A row wins with marked values 1,2,3 or 4,5,6 or 7,8,9 with all Os or Xs  
  
A diagonal wins with marked values 1,5,9 or 7,5,3 with all Os or Xs  
  
***Class Member Function and Member(variable) Specifications  
+ = public  
- = private***

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Public /private** | **New /Update /**NoUpdate | **Function/Data Member** | | **Functionality of function or data member** |
| + | **Update** | bool game\_over | | No parameters  1) return check\_board\_full function return value  2) update class function to determine a win by checking for row win or column win or diagonal win.  check board full indicaes a tie occurs only if  X or O didn’t win.     If no winner by row, column, or diagonal set winner to C |
| + |  | void start\_game(string first\_player) | | 1) first\_player function argument value must be X or O; otherwise, throw an Error exception when value is not X or O.   Error Message: Player must be X or O.  2)In function set player(private variable) to first\_player function argument.  3) After the conditional branch statement, call the clear\_board function |
| + |  | void mark\_board(int position) | | 1) Value of int must be in the range 1 to 9; otherwise, throw an Error exception if value not in this range.  Error Message: Position must be 1 to 9.  2) Private data player can’t be empty “”, throw an Error exception if player variable is “”.  Error Message: Must start game first.  3) Update the vector of pegs with user provided position.  Call set\_next\_player private function |
| + |  | string get\_player() const | | Return the next\_player value |
| + |  | void display\_board const | | No parameters Iterate vector of strings pegs to Display a tic tac toe board in 3x 3 format |
| + | **New** | string get\_winner | | return winner |
|  |  |  | |  |
|  | | **private functions** | |  |
|  | **New** | bool check\_column\_win | | A column wins with marked values 1,4,7 or 2,5,8, or 3,6,9 with all Os or Xs  (**Remember a vector index starts at 0**) |
|  | **New** | bool check\_row\_win | | A row wins with marked values 1,2,3 or 4,5,6 or 7,8,9 with all Os or Xs |
|  | **New** | bool check\_diagonal\_win | | A diagonal wins with marked values 1,5,9 or 7,5,3 with all Os or Xs |
|  | **New** | void set\_winner | | If player is X set winner to O otherwise set winner to X |
| - |  | void set\_next\_player() | | Set next\_player. If private variable player X,  player is O else player is X |
| - |  | bool check\_board\_full | | No parameters 1) return false if vector of strings pegs has available slot by checking for a “ “(space)in each element. Otherwise return true |
|  |  |  | |  |
| - |  | void clear\_board const | | No parameters 1) Set all 9 elements to a “ “ (space) |
|  |  |  | |  |
|  |  | **Class private Data** | |  |
| - |  | string player | | **Class member variable** |
|  |  | vector of string pegs | | **Class member variable 1) (initialize to 9 “ “(spaces)** |
|  | New | string winner | | Stores the winner of the game X, O or C. |
|  |  |  |  |  |
|  |  |  |  |  |

**Sample Unit Test  Case for win by first column**

TicTacToe board

|  |  |  |
| --- | --- | --- |
| X | O |  |
| X | O |  |
| X |  |  |

Vector view:

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Values | X | O |  | X | O |  | X |  |  |
| Index | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

TEST\_CASE("Test win by first column", "[X wins first column]")  
 { /\* Tic Tac Toe Board  
         123  
         456  
         789   
  
        First column win are user positions 1,4, and 7  
vector view: 0, 3, and 6  
       \*/  
        TicTacToe board;  
         board.start\_game("X");            
         REQUIRE(board.game\_over() == false);  
         board.mark\_board(1);//X          
         REQUIRE(board.game\_over() == false);  
         board.mark\_board(2);//O            
         REQUIRE(board.game\_over() == false);  
         board.mark\_board(4);//X            
         REQUIRE(board.game\_over() == false);  
         board.mark\_board(5);//O            
         REQUIRE(board.game\_over() == false);  
         board.mark\_board(7);//X  
         //X wins  
         REQUIRE(board.game\_over() == true);  
}

**Required Test Cases for Assignment (write the code in /homeworks/tic\_tac\_toe\_test/ tic\_tac\_toe\_test.cpp)**  
  
Green equals new test case

|  |  |
| --- | --- |
| **Test Case Name** | **Steps to Test** |
| Test game over | Create an instance of TicTacToe, verify that class function game\_over() returns false |
| Test set first player X | Create an instance of TicTacToe, call start\_game() function with argument X, verify that get\_player() returns X |
| Test set first player O | Create an instance of TicTacToe, call start\_game() function with argument O, verify that get\_player() returns O |
| Test Mark Position accepts values from  1 to 9 only | 1) Create an instance of TicTacToe game  2) Call the start game function   3) Use REQUIRE\_THROWS\_AS to verify game.mark\_position(0) throws an Error  4) Use REQUIRE\_THROWS\_AS to verify game.mark\_position(10) throws an Error  5) Call mark board 5 to show that an Error is not thrown. |
| **Modify this test case to determine if there is a tie** Test game over if 9 slots are selected. | 1)Create an instance of TicTacToe game 2)Call the start game function 3)Call mark board 9 times using numbers 1 to 9 in the following order(this test case will eventually be the test for CAT tie case) |
| Test win by first column-**code see above** | Create an instance of TicTacToe, call start\_game() function with argument X, set positions for first player X to 1,4,7.  Pick other positions for O. Verify that game\_over() is true. **See my example above** |
| Test win by second column | Create an instance of TicTacToe, call start\_game() function with argument X, set positions for first player X to 2,5,8.  Pick other positions for O. Verify that game\_over() is true. |
| Test win by third column | Create an instance of TicTacToe, call start\_game() function with argument X, set positions for first player X to 3,6,9.  Pick other positions for O. Verify that game\_over() is true. |
| Test win by first row | Create an instance of TicTacToe, call start\_game() function with argument X, set positions for first player X to 1,2,3.  Pick other positions for O. Verify that game\_over() is true. |
| Test win by second row | Create an instance of TicTacToe, call start\_game() function with argument X, set positions for first player X to 4,5,6.  Pick other positions for O. Verify that game\_over() is true. |
| Test win by third row | Create an instance of TicTacToe, call start\_game() function with argument X, set positions for first player X to  7,8,9.  Pick other positions for O. Verify that game\_over() is true. |
| Test win diagonally from top left | Create an instance of TicTacToe, call start\_game() function with argument X, set positions for first player X to 1,5,9.  Pick other positions for O. Verify that game\_over() is true. |
| Test win diagonally from bottom left | Create an instance of TicTacToe, call start\_game() function with argument X, set positions for first player X to 7,5,3.  Pick other positions for O. Verify that game\_over() is true. |
| Test for no winner | Create an instance of TicTacToe, call start\_game() function with argument X, set positions to values where no one wins.  Verify that game\_over() is true when all positions have been filled. |
|  |  |
|  |  |

**Implementation:**

1) Open Visual Studio Project acc-cosc-1337-spring-2020-YourUserName

2) Pull changes from Github (sync and pull)  
  
3) In the **homeworks** folder   
    In the tic\_tac\_toe folder:  
    In file named tic\_tac\_toe.h, write the class interface code.  
    In file named tic\_tac\_toe.cpp, write the class implementation.  
    In file named tic\_tac\_toe\_test.cpp, write the class test cases.  
    In file named main.cpp, run code to play the tic tac to game.

Running the program from main (write the code in /homeworks/tic\_tac\_toe/main.cpp)  
1) Create a program that will play the TicTacToe game until the user opts to quit (outer loop).  
2) Start the game with X or O. Loop here while user doesn’t provide an X or O.  
3) Create another loop(inner loop) that iterates until a winner is determined  
4) In the loop, prompt the user for a position from 1 through 9 and update board position.(Assume user never chooses a taken position)  
5) Display the current status of the board.  
6) After a winner is determined prompt user if they want to play another game.

**Upload to Github:**

 Commit your changes using the Visual Studio Git Integration (Team Explorer).  
 Push your changes to your repository.

**Submission:**

Add your Github username to  the comment section of this assignment and submit.