



# CALIFORNIA STATE UNIVERSITY, CHICO

CSCI 111-02,03,04: *Programming and Algorithms I*

EECE 135-02,03,04: *Algorithms and Programs for Engineers*

## Programming Assignment #7

Due: 12 noon, Friday, December 9 (system date/time at [tignon.ecst.csuchico.edu](http://tignon.ecst.csuchico.edu))

**Tic-Tac-Toe:** Write a C++ console program that will play a game of *Tic-Tac-Toe*. Your executable program can be invoked to either allow two humans to play, one human to play against the computer, or the computer play against itself (demo mode). The number of games played could also be set. Your solution must include a definition of a **TicTacToe** class with the following features (you can modify these to your liking):

- (private member variable) `board`, which must be a character vector (from the C++ STL classes) with nine slots representing the board.
- (private member variables) `autoP1` and `autoP2`, `bool` variables to indicate which players are computers; `true` means computer (non-human); assume P1 is 'X' and P2 is 'O'.
- (private member variable) `turn`, a character representing whose turn it is, 'X' or 'O'.
- (private member variable) `numFreeSpaces`, an unsigned value indicating the number of empty spaces on the board, 0 through 9.
- (public member function) `clear()`, that sets member variable `numFreeSpaces` to 9 and resets member variable `board` so when it is displayed it will show up as

```
1  2  3
4  5  6
7  8  9
```

- constructor with one character default argument `goesFirst` (default value is 'X' for the first turn) that sets member variable `turn` to `goesFirst`, and two `bool` default arguments: `p1auto` and `p2auto` (both defaulting to `true`, indicating both players are computers – none are human); uses the `clear()` member function to reset the board
- (public member function) `showBoard()`, that displays the board as a three by three board
- (public member function) `getWinner()`, that returns one of the following characters based on the configuration of the gameboard: 'X' if X is the winner, 'O' if O is the winner, 'T' if there is a tie, or 'N' if there is no winner yet.
- (private member function) `changeTurn()`, that changes member variable `turn` from 'X' to 'O', or vice versa.
- (public member function) `doMove()`, that places the mark in `turn` on a valid position in the board; reads console input if the player indicated by `turn` is human, and generates a valid random number if the player indicated by `turn` is a computer; calls `showBoard()` to show board after each move; guarantees all moves are valid.

Your executable program will be called **tictactoe** and may be invoked as follows:

```
tictactoe
tictactoe numberOfGames
tictactoe numberOfPlayers numberOfGames
```

The command-line argument, *numberOfPlayers*, indicates the number of “human” players, which could only be 0 (computer against itself), 1 (human player against the computer, or 2 (two humans playing). The command-line argument, *numberOfGames*, indicates the number times a game will be played between the indicated players. If *numberOfPlayers* is not specified, then the default is 0 human players (same as demo mode).

Additionally, you can have your driver program keep track of scores between 'X' and 'O'. The current score could be declared at the end of each game, and the overall winner could be declared at the end of all games played.

Here is part of a sample driver that will only play one game:

```
#include <iostream>
#include "TicTacToe.h"

using namespace std;
int main(int argc, char *argv[])
{
    TicTacToe game('O');    // 'O' makes first move; demo mode

    // Loop forever until we get a winner
    while (game.getWinner()=='N')
    {
        game.showBoard();
        game.doMove();
        cout << endl << endl;
    }

    // Display results of the game
    switch (game.getWinner())
    {
        case 'X':
            cout << "X is winner, goooo X!" << endl;
            break;
        case 'O':
            cout << "O is winner, goooo O!" << endl;
            break;
        case 'T':
            cout << "A tie!" << endl;
            break;
    }

    return 0;
}
```