```cpp
#include <iostream>
#include <limits>
#include <math.h> //exploring this lib for fmod and floor

using namespace std;

enum TicTacToeCell {
	None = ' ',
	Nought = 'O',
	Cross = 'X'
};


int main()
{
	//BUG: if a user inputs new line before any invalid or valid character,
	//		it will mess with indended control codes!

	//    Cell names:
	//                          |           |
	//            00   |  01  |     02
	//           ______|______|______
	//            10   |  11  |     12
	//           ______|______|______
	//            20   |  21  |     22
	//                          |           |

	// Number map:
	//                          |           |
	//            1             |  2   |     3
	//           ______|______|______
	//            4             |  5   |     6
	//           ______|______|______
	//            7             |  8   |     9
	//                          |           |


	//Random
	srand(time(nullptr));


	//Cells of Tic Tac Toe
	TicTacToeCell	cell00, cell01, cell02,
				cell10, cell11, cell12,
				cell20, cell21, cell22;

	//Game Flags
	bool isTurnNought = (bool)(rand()%2); //Random start
	bool isGameOver = false;
	bool isDecisiveWin = false;

	int eloNought	= 100;
	int eloCross = 100;

	//Game Counter
	int turn = 0;
	const int MAX_TURNS = 9;

	__int64 gameTimeStart;
	float gameTime;
	const float TIME_MINUTE = 60.0f;
```

```
	//Session loop
	while (!isGameOver)
	{
		//Setup
		cell00 = None;
		cell01 = None;
		cell02 = None;
		cell10 = None;
		cell11 = None;
		cell12 = None;
		cell20 = None;
		cell21 = None;
		cell22 = None;

		gameTimeStart = time(nullptr);

		//Game loop
		for (int turn = 0; turn < MAX_TURNS; turn++)
		{
			//Reset game state
			isDecisiveWin = false;

			//Clear clear game and input
			cout << "\x1b[7F\x1b[0J"; //start of 7 lines up, clear until
end of screen.

			//Display current
			cout << endl
				<< (char)cell00 << '|' << (char)cell01 << '|' <<
(char)cell02 << endl
				<< "_____" << endl
				<< (char)cell10 << '|' << (char)cell11 << '|' <<
(char)cell12 << endl
				<< "_____" << endl
				<< (char)cell20 << '|' << (char)cell21 << '|' <<
(char)cell22 << endl;


			//Do input
			int input;
			bool isValid = false;
			do
			{
				cout << (char)(isTurnNought ? Nought : Cross) << " -
Input a cell 1-9: ";
				cin >> input;

				//If invalid clear error and continue.
				if (cin.fail()) {
					//Clear error
					cin.clear();
					cin.ignore(numeric_limits<streamsize>::max(),
'\n');
				}
				else
				{
					//This is very unusual without arrays. (this
doubles as a sentinal)
					switch (input)
					{
						case 1:
						{
							if (cell00 != None) break;
```

```
                cell00 = (isTurnNought ? Nought :
Cross);
                isValid = true;
                break;
            }
            case 2:
            {
                if (cell01 != None) break;
                cell01 = (isTurnNought ? Nought :
Cross);
                isValid = true;
                break;
            }
            case 3:
            {
                if (cell02 != None) break;
                cell02 = (isTurnNought ? Nought :
Cross);
                isValid = true;
                break;
            }
            case 4:
            {
                if (cell10 != None) break;
                cell10 = (isTurnNought ? Nought :
Cross);
                isValid = true;
                break;
            }
            case 5:
            {
                if (cell11 != None) break;
                cell11 = (isTurnNought ? Nought :
Cross);
                isValid = true;
                break;
            }
            case 6:
            {
                if (cell12 != None) break;
                cell12 = (isTurnNought ? Nought :
Cross);
                isValid = true;
                break;
            }
            case 7:
            {
                if (cell20 != None) break;
                cell20 = (isTurnNought ? Nought :
Cross);
                isValid = true;
                break;
            }
            case 8:
            {
                if (cell21 != None) break;
                cell21 = (isTurnNought ? Nought :
Cross);
                isValid = true;
                break;
            }
            case 9:
            {
```

```
                                        if (cell22 != None) break;
                                        cell22 = (isTurnNought ? Nought :
Cross);
                                        isValid = true;
                                        break;
                                }
                                default:
                                {
                                        isValid = false;
                                        break;
                                }
                        }
                }

                //I feel like better flow could improve this
                if (!isValid) cout << "\x07\x1b[1F\x1b[0J"; //bell,
clear last line with invalid input
        } while (!isValid);


        //Check win conditions and break if won (VERY UGLY!!!):


        //Top horizontal (De morgans law)
        if (!(!(cell00 == cell01) || !(cell00 == cell02)) && cell00
!= None)     { isDecisiveWin = true; break; }
        //Middle horizontal

        else if (cell10 == cell11 && cell10 == cell12 && cell10 !=
None)        { isDecisiveWin = true; break; }
        //Bottom horizontal

        else if (cell20 == cell21 && cell20 == cell22 && cell20 !=
None)        { isDecisiveWin = true; break; }
        //Decending diagonal

        else if (cell00 == cell11 && cell00 == cell22 && cell00 !=
None)        { isDecisiveWin = true; break; }
        //Ascending diagonal

        else if (cell20 == cell11 && cell20 == cell02 && cell20 !=
None)        { isDecisiveWin = true; break; }
        //Left vertical

        else if (cell00 == cell10 && cell00 == cell20 && cell00 !=
None)        { isDecisiveWin = true; break; }
        //Middle vertical

        else if (cell01 == cell11 && cell01 == cell21 && cell01 !=
None)        { isDecisiveWin = true; break; }
        //Right vertical

        else if (cell02 == cell12 && cell02 == cell22 && cell02 !=
None)        { isDecisiveWin = true; break; }


        //Swap turns if continuing.
        isTurnNought = !isTurnNought;
    }

    gameTime = (float)(time(nullptr) - gameTimeStart);
    float gameTimeSeconds = fmod(gameTime, TIME_MINUTE);
    float gameTimeMinutes = floor(gameTime / TIME_MINUTE);
```

```
		//Display results
		if (isDecisiveWin)
		{
			//Do ELO
			int elo = isTurnNought ? 10 : -10; //if nought wins, add 10,
else subtract 10
			eloNought    += elo;
			eloCross     -= elo;

			//Display
			cout << (char)(isTurnNought ? Nought : Cross) << " WINS!!!"
<< endl
				<< "Game took " << gameTimeMinutes << " minutes and "
<< gameTimeSeconds << " seconds." << endl
				<< "O's ELO: " << eloNought << endl
				<< "X's ELO: " << eloCross << endl;
		}
		else
		{
			cout << "TIE!" << endl;
		}


		//Prompt to play again
		bool isValid;
		do
		{
			cout << "Would you like to play again? (y/n)?  ";
			char select;
			cin >> select;

			if (cin.fail()) {
				//Clear error
				cin.clear();
				cin.ignore(numeric_limits<streamsize>::max(), '\n');
			}
			else
			{
				switch (select)
				{
					case 'y':
					case 'Y':
					{
						isValid = true;
						break;
					}
					case 'n':
					case 'N':
					{
						isValid = true;
						isGameOver = true;
						break;
					}
					default:
					{
						isValid = false;
						break;
					}
				}
			}
```

```
                    if (!isValid) cout << "\x07\x1b[1F\x1b[0J"; //bell, clear
last line with invalid input
                }
                while (!isValid);

                if (!isGameOver) cout << "\x1b[5F\x1b[0J"; //clear display before
next game
        }

        cout << endl << "Thanks for playing!";

        //Sorry, I could not think of a way to integrate decrement into the
program!
        int integerForChecklist = 0;
        integerForChecklist--;

        return integerForChecklist;
}
```