# Tic Tac Trie

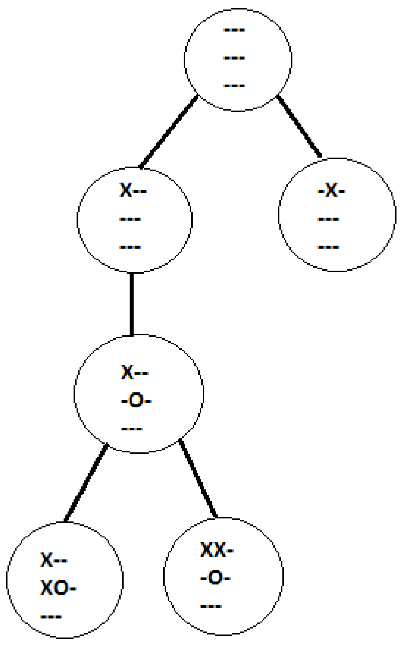
**Due**: Check due date on BlackBoard

## Introduction:

A trie is a type of tree that allows for quick lookups of string based keys. The way it works is each node has 26 children, each representing a letter of the alphabet. Each path through the tree spells out a word. For more information: https://en.wikipedia.org/wiki/Trie

We're going to create something similar for storing games of Tic Tac Toe. Each node will represent a board state, and each link will represent a single move.

Visualization:



## Classes:

There are three classes required for this project:

- TicTacNode – The node class that will be used in the other classes.
- TicTacToe – A single tic tac toe game in a linked list format
- TicTacTrie – The trie data structure made up of TicTacNodes.

## TicTacNode:

This class will hold the information about the current state of the game board.

**Required variables:**

- m_children - A vector containing pointers to all of the children of this node.
- m_board - A **map** containing the game board.

**Required functions:**

- Overloaded comparison (==) operator
- Overloaded assignment (=) operator
- Overloaded out (<<) operator

## TicTacTree:

This class will essentially be a linked list of TicTacNodes that contains the information about the game you just read in.

**Required Variables:**

- m_isOver – a Boolean for whether or not the game in the file reached an end
- m_results – an integer for the outcome of the game
    - 0 – game did not end
    - 1 – Player X won
    - 2 – Player O won
    - 3 – Game ended in a draw

**Required Functions:**

- Getters for all the member variables
- void readGame(string fileName) – This is the main function of the TicTacToe class. It will parse through the passed in file name, create nodes for each play made in the game, and determine the result of the game once it's reached the end.

Feel free to add any other helper functions.

## TicTacTrie:

This class will contain the trie data structure that will hold all paths for the TicTacToe games.

**Required Variables:**

- m_xWins – the number of times Player X has won
- m_oWins – the number of times Player O has won
- m_draws – the number of times the game resulted in a draw
- m_size – the current number of nodes in the tree

**Required Functions:**

- Getters for all of the member variables
- void addGame(TicTacToe t) – This is the main function of the TicTacTrie class. It will iterate through the TicTacToe game that is passed into it. If it finds a node that isn't already in the trie, it will add it in the appropriate spot. If it's finding moves that are already in the trie, it will follow through and add any new moves.

## Input Files:

All input files will be consistent in format and each move will be valid.

Example:

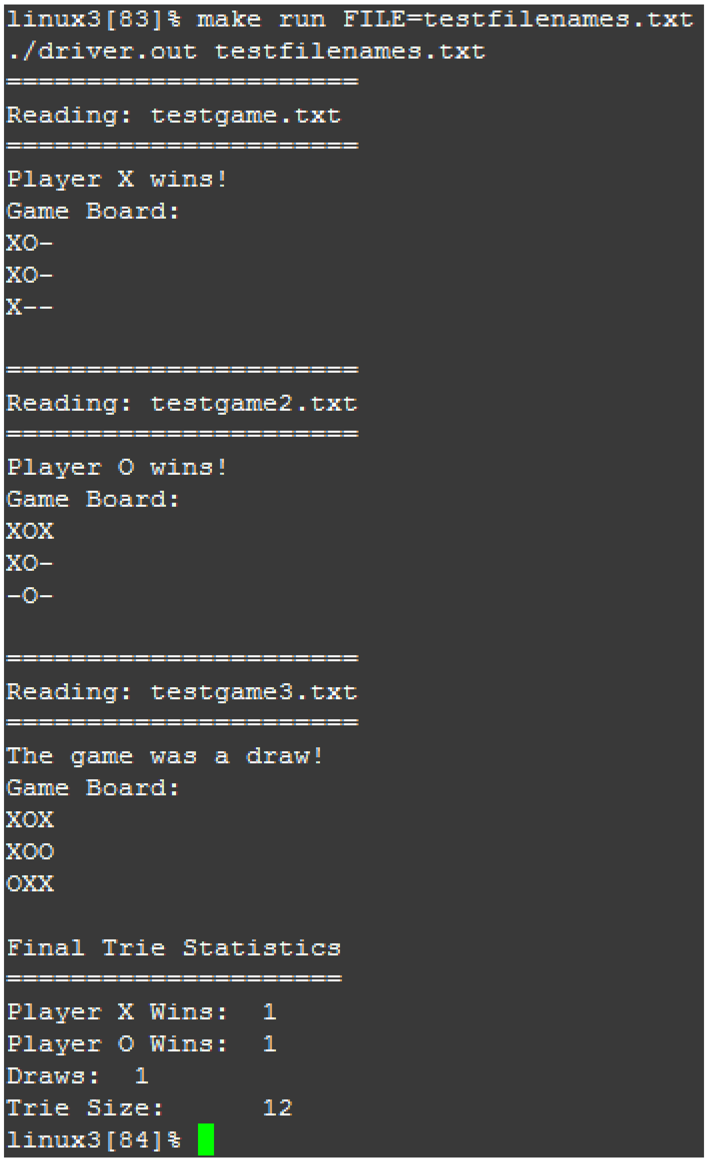
```
---

---

---


X--

---

---


X--

-O-

---
```

## Driver:

The driver will take in a file as a command line argument. The file it takes in will be a list of file names that should be read in and added into the TicTacTrie. You **must** validate the command line arguments (whether there are enough and that the file actually exists). It is safe to assume that the files listed inside will exist, and that the games inside those files will be correct and valid. All output will be done inside the driver.

Whenever the TicTacToe game finds a finished board, the driver should print out the results of that game and the final board. After all of the games have been read and added to the trie, the driver should print the final statistics of the trie.

## Sample Output:



```
linux3[83]% make run FILE=testfilenames.txt
./driver.out testfilenames.txt
======================
Reading: testgame.txt
======================
Player X wins!
Game Board:
XO-
XO-
X--

======================
Reading: testgame2.txt
======================
Player O wins!
Game Board:
XOX
XO-
-O-

======================
Reading: testgame3.txt
======================
The game was a draw!
Game Board:
XOX
XOO
OXX

Final Trie Statistics
=====================
Player X Wins:  1
Player O Wins:  1
Draws:  1
Trie Size:      12
linux3[84]%
```

## What to Submit:

Follow the course project submission procedures. You should copy over all of your C++ source code with .cpp/.h files under the src directory. **You do not need to submit the text files.** You must also supply a Makefile build file.

Make sure that your code is in the ~/cs341class/projo/ directory and not in a subdirectory of ~/cs341class/proj2/. In particular, the following Unix commands should work.

The command "make run FILE=<a file name>" should run the file. The "FILE" variable is for passing in the command line argument to your program. You **must** add $(FILE) after your normal make run command for this to work.

Don't forget the Project Submission requirements shown online!! One hint, **after you submit**, if you type:

```
  ls ~/cs341class/proj2/
```

and you see a bunch of .cpp and .h files, this is WRONG. You should see:

```
  src
```

instead. The C++ programs must be in directories under the src directory. Your submissions will be compiled by a script. The script will go to your proj0 directory and run your makefile.  This is required. You will be severely penalized if you do not follow the submission instructions.