```
/*
 * hw10.c
 *
 * Created on: Dec. 3, 2014
 *     Author: Scot Matson
 *     Course: CS49C - 01
 *
 *     HW10 Crooked Tic-Tac-Toe
 */
#include <stdio.h>
#include <stdlib.h>
#include "hw10.h"

//Grid Locations
#define TLEFT    cp->grid[0][0]
#define TOP      cp->grid[0][1]
#define TRIGHT   cp->grid[0][2]
#define LEFT     cp->grid[1][0]
#define MIDDLE   cp->grid[1][1]
#define RIGHT    cp->grid[1][2]
#define BLEFT    cp->grid[2][0]
#define BOTTOM   cp->grid[2][1]
#define BRIGHT   cp->grid[2][2]

void printGrid(struct config *cp) {

  printf("+---+-%c-+---+\n", '-');
  printf("| %c | %c | %c |\n", TLEFT, TOP, TRIGHT);

  printf("+---+-%c-+---+\n", '-');
  printf("| %c | %c | %c |\n", LEFT, MIDDLE, RIGHT)
;
  printf("+---+-%c-+---+\n", '-');
  printf("| %c | %c | %c |\n", BLEFT, BOTTOM, BRIGH
T);
  printf("+---+-%c-+---+\n", '-');
}

void Xmove(struct config *cp){
  //make next move by X
  if (cp->playsLeft == 9) { MIDDLE = 'X'; } //'X' t
akes center
```

```
  else {
          //Make moves which oppose 'O'
          if (TLEFT  == 'O' && BRIGHT == ' ') { BRIG
HT = 'X'; }
    else if (TOP    == 'O' && BOTTOM == ' ') { BOTT
OM = 'X'; }
    else if (TRIGHT == 'O' && BLEFT  == ' ') { BLEF
T  = 'X'; }
    else if (LEFT   == 'O' && RIGHT  == ' ') { RIGH
T  = 'X'; }
    else if (RIGHT  == 'O' && LEFT   == ' ') { LEFT
   = 'X'; }
    else if (BLEFT  == 'O' && TRIGHT == ' ') { TRIG
HT = 'X'; }
    else if (BOTTOM == 'O' && TOP    == ' ') { TOP
   = 'X'; }
    else if (BRIGHT == 'O' && TLEFT  == ' ') { TLEF
T  = 'X'; }
  }
  return;
}

void Omove(struct config *cp){
  int row, col;

  if(MIDDLE == ' ') { MIDDLE = 'O'; }
  else
          //Make attempt to take central opposing po
sitions.
          //If all else fails, take the remaining co
rners which 'X'
          //needs for a win.
          if (LEFT   == ' ') { LEFT   = 'O'; }
    else if (RIGHT  == ' ') { RIGHT  = 'O'; }
    else if (TOP    == ' ') { TOP    = 'O'; }
    else if (BOTTOM == ' ') { BOTTOM = 'O'; }
    else if (BLEFT  == ' ') { BLEFT  = 'O'; }
    else if (TRIGHT == ' ') { TRIGHT = 'O'; }

    else {
      row = rand() % 3;
      col = rand() % 3;
```

```
      while (cp->grid[row][col] != ' ') {
        col++;
        if (col == 3){
          col = 0;
          row = (row + 1) % 3;
        }
      }
      cp->grid[row][col] = 'O';
    }
  return;
}

enum player eval(struct config *cp) {
  //evaluate winner of game
  if (TLEFT  == TOP    && TOP    == MIDDLE && MIDDL
E == RIGHT)  return X;
  if (TLEFT  == LEFT   && LEFT   == MIDDLE && MIDDL
E == BOTTOM) return X;
  if (LEFT   == MIDDLE && MIDDLE == TOP    && TOP
  == TRIGHT) return X;
  if (LEFT   == MIDDLE && MIDDLE == BOTTOM && BOTTO
M == BRIGHT) return X;
  if (TOP    == MIDDLE && MIDDLE == RIGHT  && RIGHT
  == BRIGHT) return X;
  if (TOP    == MIDDLE && MIDDLE == LEFT   && LEFT
  == BLEFT)  return X;
  if (TRIGHT == RIGHT  && RIGHT  == MIDDLE && MIDDL
E == BOTTOM) return X;
  if (BLEFT  == BOTTOM && BOTTOM == MIDDLE && MIDDL
E == RIGHT)  return X;
  //Otherwise
  return O;
}
```