COSC2396 (UNIX and Systems Programming in C)
Assignment 2 (Tic Tac Toe Game)
Maximum Marks: 10

Game Description:
Tic-tac-toe is a classic two-payer game played on a 3 by 3 grid by two players, who alternately place the marks X and O in one of the nine spaces in the grid. A player wins when they mark all three spaces of a row, column, or diagonal of the grid. When 9 boxes are filled, it is a draw.
In our advanced variation of Tic Tac Toe game, both players can play as X's and O's and the first player to get a tic-tac-toe (3 X's or 3 O's) will win the game.

https://en.wikipedia.org/wiki/Wild_tic-tac-toe

Problem Specification:
Write C code to implement Tic Tac Toe using Process creation and Signal mechanism. For process creation can be implemented using fork() system call. Signal mechanism will need sigaction() and kill() system calls. You may utilize the concepts and code examples provided in Lecture notes. As the objective of this assignment is to learn system programming in C, it is fine to adjust the game strategy of parent and child to some extent (e.g. switching to basic Tic-tac-toe where parent always marks O & child marks X and vice versa) as long as overall it plays like a tic-tac-toc game.

Game Strategy of Parent Process:
P0) Create a 3x3 (i.e. 9 boxes) board and initiate with empty spaces.
P1) Call a random number r between 0..8 to determine board(i, j) where i=r/3 and j =r%3
P2) Find the current machine time (seconds elapsed since epoch January 1, 1970). Place X at board(i, j) if the current time in seconds is an even number and O if the current time in seconds is an odd number.
P3) Parent sends SIGUSR1 to child process requesting child's move.
P4) If it receives SIGUSR1 (or SIGUSR2) from child process, then it knows that child wants to place X (or O) as next move. Call a random number r between 0..8 to determine box for child's move board(i, j) where i=r/3 and j =r%3. If the board(i, j) is already filled, find an empty box by inspecting the next boxes sequentially i.e. r+1, r+2, so on.
P5) Parent then needs to decide its own next move, essentially repeat from Step P1.
P6) Whenever there is a winner or there is a draw, parent sends SIGUSR2 to child process announcing the game over so that child process can gracefully exit.

Game Strategy of Child Process:
C1) Upon receiving SIGUSR1 from parent process, child tries to decide its next move.
C2) Find the current machine time (seconds elapsed since epoch January 1, 1970). Decides to place X if the current time in seconds is an even number and O if the current time in seconds is an odd number. Accordingly, sends SIGUSR1 for X and SIGUSR2 for O.
C3) Upon receiving SIGUSR2 from parent process, child tries to determine whether it was a win, loss or draw to plan for its celebration. Accordingly, it prints a message and terminate itself sending SIGTERM signal to itself. [Hint: parent makes 1st move, so it will make the 9th move]

Bonus Task:
Do not let child process terminate upon receiving SIGUSR2, especially if child lost the game. Instead, child may request the parent to restart the game from the beginning. [Hint: check whether

current seconds is even or odd and accordingly let child decide terminate or replay by sending a signal SIGUSR1 (or SIGUSR2) to parent requesting a replay with X (or O) as the first mover.]

Submission Instructions:

1) All submissions must be on LMS for evaluation and grading.
2) Prepare a document explaining problem and solution (Pseudocode).
3) Write C code to implement Tic Tac Toe Game.
4) Create a few testcases and use the testcases to validate Tic-Tac-Toe program on UNIX system.
5) Add the screenshot to show the results from run of your program.
6) Record a video demonstrating execution of the game on the UNIX machine.
7) Submit three files following the naming convention below:
   7a) COSC2396-<Section>-Assignment2-<StudentName>.c
   7b) COSC2396-<Section>-Assignment2-<StudentName>.pdf
   7c) COSC2396-<Section>-Assignment2-<StudentName>.mp4

Marking Scheme:

Description	1
Pseudocode	1
Testcases	1
Demo	2
Complete program with comments, no compile/logical error and correct output	5