**Do Part I and (Part II or Part III)**

**Part I**

Modify our Tic Tac Toe game to play 1000 games and determine who wins using a one dimensional array. You can start with our classwork. You will need to add checks for all rows, columns and diagonals.

**Part II**

Modify the Tic-Tac-Tac game program to use a two dimensional array, and simulates playing a game. The basic flow for populating the board is:

For Player 1, pick a random space to move, if empty place 1 in that space.

Now its Player 2 turn.

For Player 2, pick a random space to move, if empty place 2 in that space.

Now its Player 1 turn again.

Keep moving until the board is full or a player wins.

Then play 1000 iterations of the game, within 3 modes. After the thousand games in each mode, display the number of times X won, O won and ties. The three modes are:  
  
1)  X and O move randomly as described above. If a random square chosen has been played, have the player choose again.  
2) X always moves to the middle square in the first move.  
3) O always moves to the middle square in the first move, is X has not moved there.

Your program should be structured like the following:

For mode from 1 to 3 {

For each game 1 to 1000 {

Play game – considering the current mode

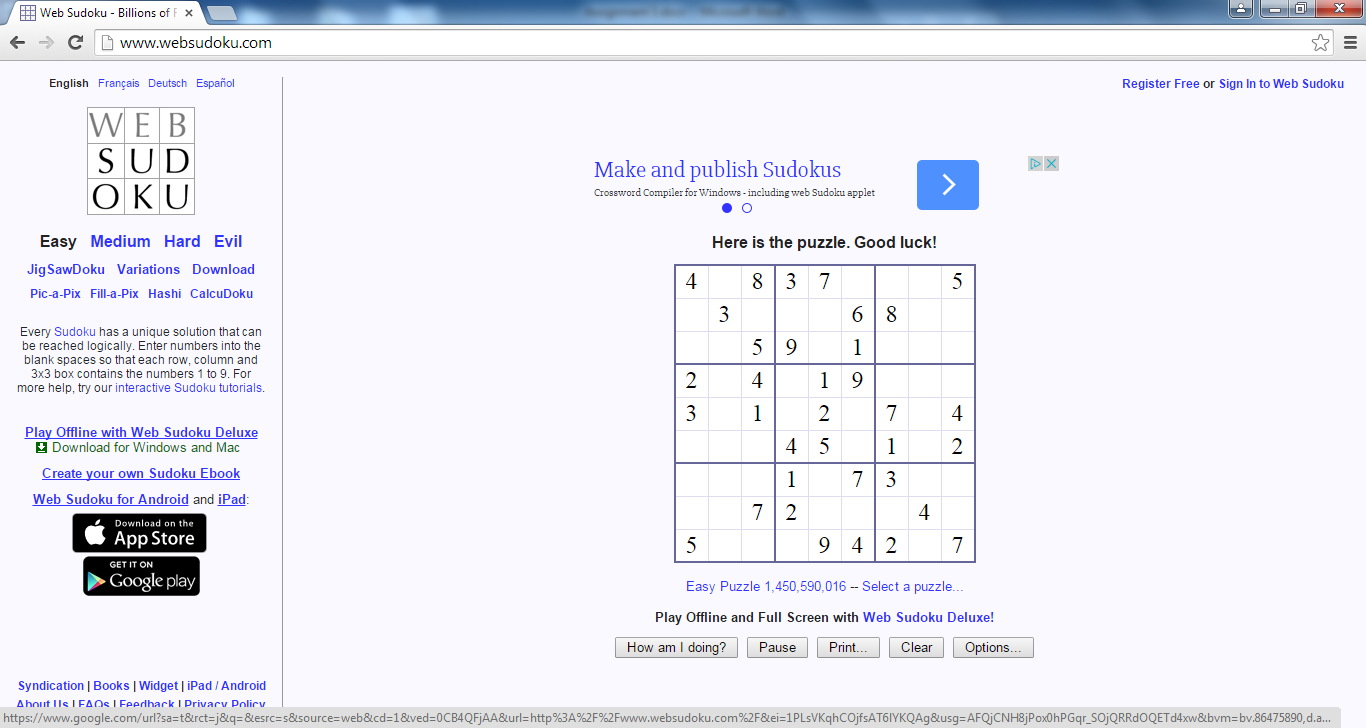
Add result to totals

}

Print totals (#Xwins, #Owins, #Ties)

}

**Part III**



This part of the assignment will generate a series of SuDoKu Boards (see below) and randomly attempt to fill the squares . The goal is to try to randomly populate remaining square to achieve a solution.

For one game, start with a SuDoKu board

1. Generate a 9 x 9 set of random numbers , each from 1-9, representing a SuDoKu board and store them in an 2 dimensional array.

Then for 1000 iterations:

1. Remove duplicates in :
   1. Rows
   2. Columns
   3. 9 3x3 boxes as shown above (OPTIONAL)
   4. A duplicate is the existence of the same number in a row, column or optionally a box
2. Populate squares on the Board that now have blanks (previous duplicate locations)

Finally determine the number filled in boxes, to find the seed that yields the best result (most filled in squares after 1000 populations.

Now Play 100 games, each starting with a different seed in your random number generator. The goal is to find the seed that will play a game and achieve the most populated board. Your only output is the seed that can produce the most populated board and the number of squares populated by that seed.

Your solution should include a FindDuplicates method and a Populate Board method.