Project 2
CSC 2053 - Platform Based Computing
Grading: 100 points
Due Date: Nov. 15, 2017

Part 1 - Solve Puzzle using Recursive Depth First Search (80 points)

We continue our implementation of graphs and trees using another game example, Sudoku. The objective of Sudoku is to fill a 9x9 grid with digits so that each column, each row, and each of the nine 3x3 subgrids that compose the grid (also called "boxes", "blocks", or "regions") contains all of the digits from 1 to 9. The puzzle setter provides a partially completed grid, which for a well-posed puzzle has a single solution. As we saw in Project 1, graphs and trees have can represent a multitude of problems. We will create a game tree that solves sudoku using a recursive depth-first search in Javascript. This project is meant to reinforce the basic data structures, variables, and functions that you have learned as well as complex notions of recursion, graphs, and the usage of Javascript, HTML, and CSS. Use the following template code below as a starting point.

	7		2	3	8			
			7	4		8		9
	6	8	1		9			2
	3	5	4					8
6		7	8		2	5		1
8					5	7	6	
2			6		3	1	9	
7		9		2	1			
			9	7	4		8	

```
File: index.html
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" href="sudoku.css"/>
  </head>
  <body>
    <table>
      <tr>
        <td id='r11'></td>
        <td id='r12'></td>
        <td id='r13'></td>
        <td id='r14'></td>
        <td id='r15'></td>
```

```
        <td id='r16'></td>
        <td id='r17'></td>
        <td id='r18'></td>
        <td id='r19'></td>
      </tr>
      <tr>
        <td id='r21'></td>
        <td id='r22'></td>
        <td id='r23'></td>
        <td id='r24'></td>
        <td id='r25'></td>
        <td id='r26'></td>
        <td id='r27'></td>
        <td id='r28'></td>
        <td id='r29'></td>
      </tr>
      .
      .
      .
      <tr>
        <td id='r91'></td>
        <td id='r92'></td>
        <td id='r93'></td>
        <td id='r94'></td>
        <td id='r95'></td>
        <td id='r96'></td>
        <td id='r97'></td>
        <td id='r98'></td>
        <td id='r99'></td>
      </tr>
    </table>
  </body>
  <script src="sudoku.js"></script>
</html>


File: sudoku.js
var sol = [[0,7,0,2,3,8,0,0,0],
  [0,0,0,7,4,0,8,0,9],
  [0,6,8,1,0,9,0,0,2],
  [0,3,5,4,0,0,0,0,8],
  [6,0,7,8,0,2,5,0,1],
  [8,0,0,0,0,5,7,6,0],
  [2,0,0,6,0,3,1,9,0],
  [7,0,9,0,2,1,0,0,0],
  [0,0,0,9,7,4,0,8,0]];

var printBoard = function(s) {
  //display the beginning board
};
var printSolution = function(s) {
  //display the final solution
};
var solve = function(s) {
  //solve the puzzle s using a recursive depth first search
};
var solved = function(s) {
```

```
72    //check to see if the puzzle s is solved.  Return true or false.
73  };
74  printBoard(sol);
75  solve(sol);
76
77
78  File: sudoku.css
79  table{
80    border-collapse: collapse;
81  }
82  td {
83    border: 1px solid black;
84    padding: 10px;
85    margin: 0px;
86  }
```

Part 2 - PDF write up (20 points) Create a 1-2 page write up that summarizes the interesting parts of your program. Include any problems or insights that you encountered.

Sample solution of the given problem. Your program should output the solution in HTML in a similar fashion.

9	7	1	2	3	8	4	5	6
5	2	3	7	4	6	8	1	9
4	6	8	1	5	9	3	7	2
1	3	5	4	6	7	9	2	8
6	4	7	8	9	2	5	3	1
8	9	2	3	1	5	7	6	4
2	5	4	6	8	3	1	9	7
7	8	9	5	2	1	6	4	3
3	1	6	9	7	4	2	8	5

Deliverables: Submit on Blackboard.