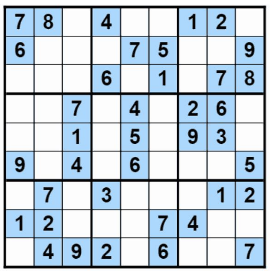
Lab 4: Constraint satisfaction

The task in this assignment is to create a classic sudoku solver. If you are not familiar with the rules of classic sudoku, visit this website and become an expert before you start: http://www.conceptispuzzles.com/?uri=puzzle/sudoku/rules

Your program will receive a 9x9 board with numbers from 0 to 9 and should return the number of permutations performed to solve the puzzle. During the execution it also should replace the entries with value 0 with number from 1 to 9 according to the rules of the game. **Remember your program can only change the cells with value 0** (it represents a blank square). The board:



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

Will be represented by:

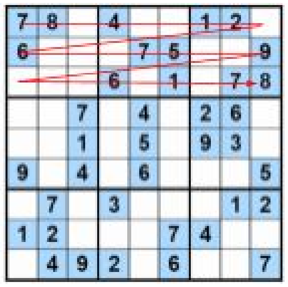
```
sudoku[0]=[7,8,0,4,0,0,1,2,0]
sudoku[1]=[6,0,0,0,7,5,0,0,9]
sudoku[2]=[0,0,0,6,0,1,0,7,8]
sudoku[3]=[0,0,7,0,4,0,2,6,0]
sudoku[4]=[0,0,1,0,5,0,9,3,0]
sudoku[5]=[9,0,4,0,6,0,0,0,5]
sudoku[6]=[0,7,0,3,0,0,0,1,2]
sudoku[7]=[1,2,0,0,0,7,4,0,0]
sudoku[8]=[0,4,9,2,0,6,0,0,7]
```

Part 1: Sequential solution

For this first part you are to complete two functions:

- sudoku_backtracking that return an integer representing the number of permutations performed to solve the board using a simple backtracking algorithm.
    - For C: int sudoku_backtracking(int **sudoku)
    - For python: def sudoku_backtracking(sudoku)
- Sudoku_forwardchecking that return an integer representing the number of permutations performed to solve the board using a forward checking algorithm.
    - For C: int sudoku_forwardchecking(int **sudoku)
    - For python: def sudoku_forwardchecking(sudoku)

Remember to keep the solution space under control, the value you return will be upper and lower bounded to evaluate if the solution matches the expected footprint. To match the expected results, remember to explore the solution following this sequence:



Part 2: MRV

For part two you are to complete one function:

- sudoku_mrv that return an integer representing the number of permutations performed to solve the board using a mrv algorithm.
  - For C: int sudoku_mrv(int **sudoku)
  - For python: sudoku_mrv(sudoku)

In case your program finds two or more variables equally restricted it should pick the one that appears first using the sequential order of part one.

Considerations:

- We provide some boards to test your solution but grading will be done with another set.
- The number of permutations refers to the number of different nodes analyzed (size of the solution space explored), no matter if they were or not discarded before making changes to the sudoku array, it has to reflect the whole size of the solution space explored**.**
- You cannot change cell in the board that start with a value different than 0.
- The results of the functions has only an upper bound in the lab but in the grading will have a lower bound too to make sure it matches the expected for the algorithm.
- The running time of your functions cannot be longer than 5 seconds for any board, otherwise it will fail the grading tests.
- All functions will be tested independently, so you will get credit for each one that returns the right results, but you have to make sure your program compiles and run the testLab properly.
- Make sure you follow the academic honesty and plagiarism rules given on the first day of class and in the syllabus on canvas.
- Be sure to submit your lab 4 code!