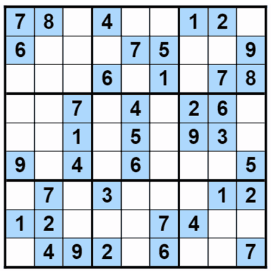
<u>Academic honesty and plagiarism</u>

- Plagiarism is unacceptable.
  - Code you submit must be your own. No copying, adapting, or submitting code you did not create is allowed. Working together and presenting variants of the same file is not acceptable. Here are some specific guidelines to make sure you don’t cross the line:
    - Do not exchange programs or program fragments in any form on paper, via e-mail, photos, or by other means.
    - Do not copy solutions from any source, including the web or previous quarters’ students.
    - Do not discuss code with other students at the level of detail that will lead to identical programs or program fragments.
  - Ask me if you are uncertain.
  - **All violations of the academic honesty will be immediately referred to the dean's office.**
- **Plagiarism detection will be applied to all code. It is very good and will catch you!! (20% of a class referred to dean!!)**

Lab 5: Constraint satisfaction Due May 12, 7pm

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:

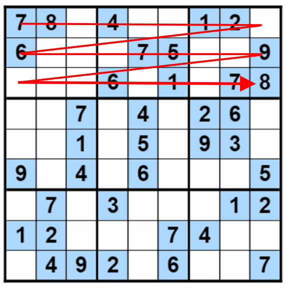
```
sudoku[0]=[7,8,0,4,0,0,1,2,0]
sudoku[1]=[6,0,0,0,7,5,0,0,9]
sudoku[2]=[0,0,0,6,0,1,0,7,8]
sudoku[3]=[0,0,7,0,4,0,2,6,0]
sudoku[4]=[0,0,1,0,5,0,9,3,0]
sudoku[5]=[9,0,4,0,6,0,0,0,5]
sudoku[6]=[0,7,0,3,0,0,0,1,2]
sudoku[7]=[1,2,0,0,0,7,4,0,0]
sudoku[8]=[0,4,9,2,0,6,0,0,7]
```

Part 1: Sequential solution

For this first part you are to complete two functions:

- sudoku_backtracking that returns an integer representing the number of recursive function calls performed to solve the board using a simple backtracking algorithm. Your implementation should not make any recursive calls for assignments that violate constraints.
- Sudoku_forwardchecking that returns an integer representing the number of recursive function calls performed to solve the board using a forward checking algorithm. Your implementation should not make a recursive call unless the new assignment under consideration does not leave any domains empty in the forward check, and does not violate constraints.

To match the expected results, remember to explore values low to high, and variables left to right, top to bottom, as shown here:



Your code should be run in Python3 and not include any extra modules other than common.

Considerations:

- We provide some boards to test your solution but grading will be done with another set. All boards will have a solution.
- We provided a function for checking constraints in “common”: can_yx_be_z(), you can use if you like, but it is not required.
- You cannot change cells in the board that start with a value different than 0.
- The running time of your functions cannot be longer than 1 min 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 runs properly with python3.

- Make sure you follow the academic honesty and plagiarism rules given on the first day of class and in the syllabus on canvas.