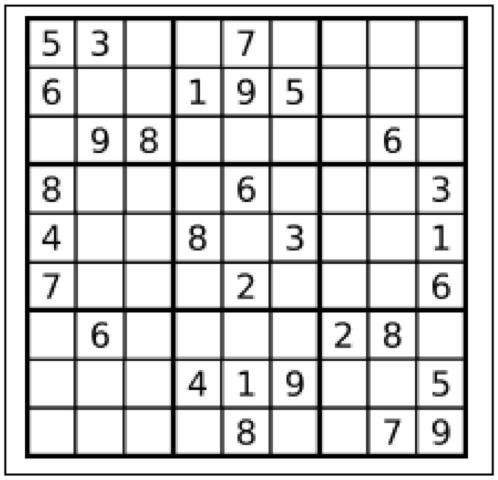
CS 4613 **E. K. Wong**
Project 2: Sudoku **Spring 2022**

Total number of points = 100. You can work on the project by yourself or you can work in a team of two.

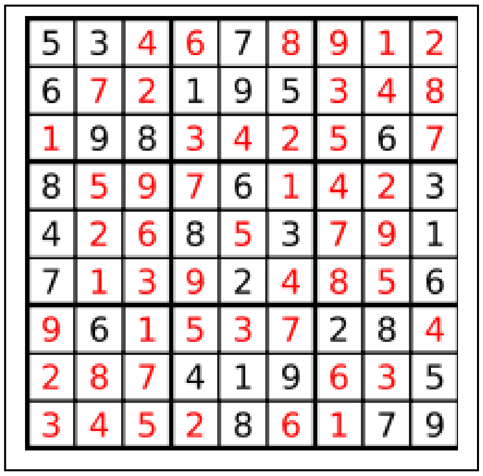
Project Description: Implement the *Backtracking Algorithm* to solve Sudoku puzzles. The rules of the game are:

- The game board consists of 9×9 cells divided into 3×3 non-overlapping blocks as shown in Figure 1 below. Some of the cells already have numbers (1 to 9) assigned to them initially.
- The goal is to find assignments (1 to 9) for the empty cells so that every row, column, and 3×3 block contains all the digits from 1 to 9. Each of the 9 digits, therefore, can only appear once in every row, column, and block. Figure 2 shows the solution for the puzzle in Figure 1.



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

Figure 1. Initial game board



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

Figure 2. Solution

Implement the *Backtracking Algorithm* in Figure 3 below to solve Sudoku puzzles. Use the *minimum remaining value* and *degree* heuristics to implement the *SELECT-UNASSIGNED-VARIABLE* function. For the *ORDER-DOMAIN-VALUES* function, simply order the domain values in increasing order from 1 to 9 instead of using heuristics. You do not have to implement the INFERENCE function inside the Backtracking Algorithm. (Reminder: when implementing the *minimum remaining value* and *degree* heuristics, two or more variables are neighbors if they share a common constraint.)

Input and output files: Your program will read in the initial game board cell values from an input text file and produce an output text file that contains the solution. The input file contains 9 rows (or lines) of integers, as shown in Figure 4 below. Each row contains 9 integers ranging from 0 to 9, separated by blank spaces. Digits 1-9 represent the cell values and 0's represent blank cells. Similarly, the output file contains 9 rows of integers, as shown in Figure 5 below. Each row contains 9 integers ranging from 1 to 9 (without 0,) separated by blank spaces.

Testing your program: Three input test files will be provided on NYU Brightspace for you to test your program.

CS 4613 **E. K. Wong**
Project 2: Sudoku **Spring 2022**

Recommended languages: Python, C++/C and Java. If you would like to use a different language, send me an email first.

What to submit: The following files on Brightspace by the due date. Submit as separate files. Do not combine them into a single ZIP file. If you work with a partner, only one of you needs to submit but put both partners' names on the PDF report (Please do not submit your project twice.)

1. Your source code file. Put comments in your source code to make it easier for someone else to read your program. Points will be taken off if you do not have comments in your source code.
2. The output text files generated by your program. Name your output files *Output1.txt*, *Output2.txt* and *Output3.txt*.
3. A PDF report that contains the following:

   a. Instructions on how to run your program. If your program requires compilation, instructions on how to compile your program should also be provided.
   b. A description of your formulation of Sudoku as a constraint satisfaction problem. This includes the set of *variables* you have defined, the *domains* for the variables, and the set of *constraints* you have set up.
   c. Also, copy and paste the output files and your source code onto the PDF report (to make it easier for us to grade your project.) This is in addition to the source code and output files that you have to submit separately (as described in 1 and 2 above.)

```
function BACKTRACKING-SEARCH(csp) returns a solution or failure
  return BACKTRACK(csp, { })

function BACKTRACK(csp, assignment) returns a solution or failure
  if assignment is complete then return assignment
  var ← SELECT-UNASSIGNED-VARIABLE(csp, assignment)
  for each value in ORDER-DOMAIN-VALUES(csp, var, assignment) do
     if value is consistent with assignment then
        add {var = value} to assignment
        ~~inferences ← INFERENCE(csp, var, assignment)~~
        ~~if inferences ≠ failure then~~
          ~~add inferences to csp~~
          result ← BACKTRACK(csp, assignment)
          if result ≠ failure then return result
          ~~remove inferences from csp~~
        remove {var = value} from assignment
  return failure
```

Figure 3. The Backtracking Algorithm for CSPs.

```
n n n n n n n n n
n n n n n n n n n
n n n n n n n n n
n n n n n n n n n
n n n n n n n n n
n n n n n n n n n
n n n n n n n n n
n n n n n n n n n
n n n n n n n n n
```

Figure 4. Input file format. *n* is an integer between 0 and 9, with 0s representing blank cells.

```
n n n n n n n n n
n n n n n n n n n
n n n n n n n n n
n n n n n n n n n
n n n n n n n n n
n n n n n n n n n
n n n n n n n n n
n n n n n n n n n
n n n n n n n n n
```

Figure 5. Output file format. *n* is an integer between 1 and 9.