CS 4613 **E. K. Wong**
Project 2: SUDOKU (*Revised*) **Spring 2019**

Total number of points = 100.

Project Description: Design and implement a program to solve SUDOKU puzzles. The rules of the game are:

- The game board consists of 9 x 9 cells divided into smaller blocks of 3 x 3 cells as shown in the diagram below (See Figure 1.)
- Some of the cells already have numbers (1 through 9) assigned to them.
- The goal is to find assignments (1 though 9) for the empty cells such that every row, every column, and every 3 x 3 block contains each of the digits from 1 to 9 (See Figure 2.) To satisfy this requirement, a digit can only appear once in any single row, column or block.

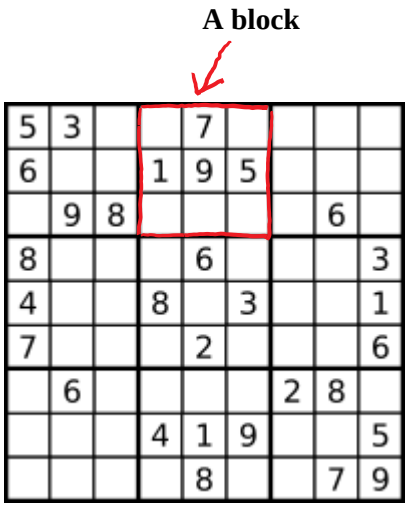

Figure 1. Initial cell values.

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.

As the first step in your program, use *Forward Checking* (slide # 18 of lectures slides for CSP) to reduce the domain of empty cells, based on the values of cells that already have a number. Note that if an empty cell has only one value left in its domain after domain reduction, you can further apply forward checking on the cell's neighbors. If any empty cell has an empty domain after applying forward checking, then the puzzle does not have a solution and the program can stop here. Next, use the *Backtracking Algorithm* (slide # 12 of the lecture slides for CSP) you have learned for solving constraints satisfaction problems. Implement the function *SELECT-UNASSIGNED-VARIABLE* in the algorithm by using the *minimum remaining values (MRV)* heuristic, and in case of a tie, use the *degree heuristic* as tie breaker. There is no need to implement the *least constraining value* heuristic in the *ORDER-DOMAIN-VALUES* function; instead, simply order the domain values in increasing order (from lowest to highest.) For the INFERENCE function, use *Forward Checking* again to reduce the domain of variables and to detect early failures. (As mentioned in class, you only have to use *Forward Checking* before you run the *Backtracking Algorithm.* You do not have to implement the INFERENCE function with *Forward Checking* inside the *Backtracking Algorithm, but if you do, it is also good.*)

Your program will read in values from an input text file and produce an output text file that contains the solution. The input file contains nine rows (or lines) of integers, with each row containing nine integers, ranging from 0 to 9 and separated by one or more blanks. The nine rows correspond to the values of the cells on the SUDOKU game board. A value of 0 indicates a blank cell. The output

file has the same format, except the 0's are replaced with values from 1 through 9 after the program finds the solution.

Testing your program: I will provide a few test input files on NYU Classes for you to test your program but you can also create your own test files. A 15 min time slot will also be set up during class time for you to run your program on test files that will be provided to you in class at that time (Details to be announced later.)

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

Hand in by due date on NYU Classes:

1. A text file that contains the source code. 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. Output text files for the test files that will be provided on NYU Classes.
3. An MS Words or PDF file that contains instructions on how to run your program. Also, copy and paste your output files and your source code onto the Words or PDF file.

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