EE231002 Introduction to Programming

Lab06. Finding Sudoku Solutions

Due: Nov. 4, 2017

Sudoku is a popular logic puzzle. The purpose of the game is to fill the 9 by 9 matrix with numbers from 1 to 9 and meeting the following constraints:

1. Each row has no duplicated numbers,
2. Each column has no duplicated numbers,
3. Each 3 by 3 submatrix has no duplicated numbers.

The game starts with a partially filled matrix, and the player is asked to fill all the blanks meeting the three constraints above. For example, the left matrix below is an unfilled matrix and the right one is a solution.

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

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

Your assignment is to write a C program to read in a partially filled matrix and to find all solutions meeting the constraints. Though a properly designed Sudoku matrix should have only one solution, your program should not make this assumption, instead it should try to find **all** solutions possible. Nine puzzles are given for you to test your program. They are `s1.dat` to `s9.dat`. Note that the blanks in these files are denoted by a `'.'`. Your program should be able to read it in and convert it to a digit other than 1–9. To read input from an existing file, one can use `unix` redirection as

```
$ a.out < s1.dat
```

Then, the content of the file `s1.dat` is treated as the standard input, and no keyboard input is expected from the program. The output of your program should look like the following.

```
$ ./a.out < s9.dat
Solution 1:
  3 6 1 | 7 9 4 | 2 8 5
  9 2 8 | 3 5 1 | 4 7 6
  5 7 4 | 2 8 6 | 1 3 9
  ------|-------|------
  4 8 2 | 6 1 9 | 3 5 7
  1 3 9 | 8 7 5 | 6 4 2
  7 5 6 | 4 2 3 | 8 9 1
  ------|-------|------
  6 9 3 | 5 4 2 | 7 1 8
  8 4 5 | 1 6 7 | 9 2 3
  2 1 7 | 9 3 8 | 5 6 4
Total number of solutions found: 1.
```

Notes.

1. Create a directory **`lab06`** and use it as the working directory.
2. Name your program source file **`lab06.c`**.
3. You should try more input matrices to ensure that your program functions correctly.
4. After you finish verifying your program, you can submit your source code by

   ```
   $ ~ee2310/bin/submit lab06 lab06.c
   ```

   If you see a "submitted successfully" message, then you are done. In case you want to check which file and at what time you have submitted your labs, you can type in the following command:

   ```
   $ ~ee2310/bin/subrec lab06
   ```

   It will show the last few submission records.