CS 2123-001 Data Structures

Instructor Dr. Turgay Korkmaz

Homework 1
Due date: check BB
!!!! NO LATE HOMEWORK WILL BE ACCEPTED !!!

Background (from our textbook)
In the last several years, a new logic puzzle called Sudoku has become quite popular throughout the world. In Sudoku, we start with a 9 x 9 grid of integers in which some of the cells have been filled in with digits between 1 and 9. Our job in the puzzle is to fill in each of the empty spaces with a digit between 1 and 9 so that each digit appears exactly once in each row, each column, and each of the smaller 3 x 3 squares. Each Sudoku puzzle is carefully constructed so that there is only one solution. For example, given the **puzzle** shown on the left of the following diagram, the unique **solution** is shown on the right:

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

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

HW- Sudoku:
Currently you may not have learned the algorithmic strategies that you need to solve Sudoku puzzles. But you can easily

- declare a data structure to represent a Sudoko puzzle,
- write some functions to read Sudoko puzzle from keyboard/file,
- check if a proposed solution follows the Sudoku rules against duplicating values in a row, column, or outlined 3 x 3 square (if needed, we will discuss Sudoko rules in the class).
- also write a function that can list the possible values for each empty cell according to Sudoku rules.

So instead of asking you to write a program that can solve a Sudoko puzzle, which will be hard, **you are asked to write a program that can do some of the simple things mentioned above.** Specifically, your program must do the followings:

1. Declare a minimum size data structure to store the values of a Sudoko puzzle (suppose empty cells will have the value of 0 while the other cells will have values from 1 to 9).

2. Then ask user if he/she wants to enter data for a sudoku puzzle or a sudoku solution. Suppose user enters 1 to enter a puzzle; or 2 to enter a solution to be checked.
3. Write/call a function that asks user to enter the values for a Sudoku puzzle or solution and saves these values into the data structure you declared in the previous item. User is expected to enter a number between 0 and 9 for each cell in case of a puzzle. Zero is for empty cells, 1-9 are for other cells. In case of a solution, user is expected to enter only the values between 1 and 9. Your program should reject invalid values.

   [Note: I am sure you are wondering if you have to enter 82 values by hand. Unfortunately, yes :(until we learn how to read data files in ch3. But there is another option to avoid entering all these numbers.
   Basically develop your program as if you will read the values from keyboard. But when you are running the program, the operating system's redirection mechanism will help you to read these values from a **file** so you don't need to enter them by hand every time. Here how that option works: suppose you implement your program called hw1

   If you run it as follows,

   ```
   elk03>  hw1
   ```

   it will ask you to enter the option (1 for puzzle data or 2 for solution data) and then enter 81 integer values by hand. After you enter the values, the program will work.

   But if you first save puzzle data or solution data into a file (say `puzzle1.txt`) and then run the program as follows using redirection,

   ```
   elk03>  hw1  < puzzle1.txt
   ```

   then the operating system will read the numbers from the file `puzzle1.txt` and pass the values to the program when it executes `scanf("%d", &value);` so you won't need to enter anything by hand. But you need to create `puzzle1.txt` and put 82 values in it, for example here is the file for the puzzle data given above.

   ```
   1
   0 0 2 4 0 5 8 0 0
   0 4 1 8 0 0 0 2 0
   6 0 0 0 7 0 0 3 9
   2 0 0 0 3 0 0 9 6
   0 0 9 6 0 7 1 0 0
   1 7 0 0 5 0 0 0 3
   9 6 0 0 8 0 0 0 1
   0 2 0 0 0 9 5 6 0
   0 0 8 3 0 6 9 0 0
   ```

   I post this file and other files on the class web page and BB Learn so you can download them and save under the same directory where you will develop your hw1.c. You can create other files too. We will check your programs using other files as well... **]**

4. If the given file contains a sudoku puzzle (i.e., option 1 is selected), your program should print possible values for each cell containing 0. In case of the above puzzle data, the output would be as follows

   ```
   [0][0]: 3, 7
   ```

```
[0][1]: 3, 9
[0][4]: 1, 6, 9
.....
```

5. If the given file contains a Sudoku solution (i.e., option 2 is selected), then your program should check if the solution is a valid one according to Sudoku rules. Accordingly prints Yes or No.
6. When implementing your program use modular programming as much as possible. For example, you may develop some utility functions that will be helpful for performing the tasks in items 4 and 5 (try to avoid the same codes that are doing the same things in different parts of your program).

What to return: !!!! NO LATE HOMEWORK WILL BE ACCEPTED !!!

1. Follow the problem solving methodology, and solve the problem(s). Then convert your solution(s) to a C program. Include /* comments */ so that we can understand your solution(s). You can name your program as hw01.c
2. Compile and run it. Copy/paste the result in an output file, say out1.txt. (or you can use **script** command in linux as follows)

> script out1.txt # saves everything into out1.txt until you hit **ctrl-d**
>hw01
…

3. Zip your files hw01.c and out01.txt as a Lastname-hw01.zip file
4. Go to BB Learn, and submit your .zip file as an attachment before the deadline.

/* Don't forget to include comments about the problem, yourself and each major step in your program! */

You must submit your work using Blackboard Learn and respect the following rules:

1) All assignments must be submitted as either a zip or tar archive file unless it is a single pdf file.
2) Assignments must include all source code.
3) Assignments must include an output.txt file which demonstrates the final test output run by the student.
4) If your assignment does not run/compile, the output.txt file should include an explanation of what was accomplished, what the error message was that prevented the student from finishing the assignment and what the student BELIEVES to be the underlying cause of the error.

HINT:

To minimize the size of the structure that holds Sudoku puzzle, you can use `char` as the smallest type to hold integer values between 0 and 9. So you can define your puzzle as `char puzzle[9][9];` But then be careful when reading "an integer value" into a `char` variable with `scanf`; for example

```
char ch;
scanf("%d", &ch); /* will give segmentation fault, why? */
```

here is some hint to get around this problem. I assume you will have the following declarations in this hw. If not, declare them as follows...

```
int UserChoice, tmp;
char puzzle[9][9];
/* ... */
scanf("%d",&UserChoice);
if (UserChoice == 1 || UserChoice == 2 ) ....
```

when you are reading values for the puzzle or solution data into 9x9 char array, do the following:

```
for(i=0; i<9; i++){
   for(j=0; j< 9; j++){
      scanf("%d", &tmp);
      puzzle[i][j] = tmp;
     /* just  scanf("%d", &puzzle[i][j]);
        would be wrong!  why? */
   }
}
```

BTW, the above problem does not happen with `printf`, why? So when you need to print an integer value stored in `puzzle[i][j],` you can simply say

```
printf("%d ", puzzle[i][j]);
```