# EE 658 – Fall 2012

## Assignment 1

**Due:** Thursday, September 6, by **8:00am**.
**Maximum points**: 100. You must complete this assignment by individual effort.

## Implications in Sudoku Puzzles

# 1 Background

### 1.1 Pedagogical objectives

Every time I teach EE 658, many students ask me about the level and amount of programming involved in this class. The first objective of this assignment is to give you a flavor of the type of programming background that is desired. The second objective is to expose you to many key concepts that we will cover in the first half of the semester, but using an example that requires no prior knowledge of these concepts. In this manner, this assignment will make it easier for you to understand many concepts that we will cover in a few weeks.

### 1.2 What is Sudoku?

Sudoku is a puzzle that originated in Japan and spread around the world a few years ago. I have noticed that practically every newspaper around the world (at least the ones I have seen in recent years) carries Sudoku. There are hundreds of web-site and perhaps thousands of books. I was introduced to this puzzle in an article in the on-line magazine Slate (Seth Stevenson, "My Days Are Numbered: I'm addicted to a Japanese logic puzzle. You will be, too," posted on May 31, 2005 at 6:19 AM ET, available at http://www.slate.com/id/2119796/).

The rules of Sudoku are extremely simple and you should be able to find and understand them in less than five minutes. I will use the following terms when I discuss Sudoku.

1. Array and cell: Let us view a standard Sudoku puzzle as a 9-by-9 **array**, where each element of the array is referred to as a **cell**.
2. Row: All nine cells along any horizontal straight-line collectively constitute a **row**. The puzzle has nine rows, numbered 1 to 9 starting at the top.
3. Column: All nine cells along any vertical straight-line collectively constitute a **column**. The puzzle has nine columns, numbered 1 to 9 starting on the left.
4. Each cell in an array is specified as (row-num, column-num), where row-num and column-num are respectively the number of the row and the number of the column to which the cell belongs.
5. Sub-array: The entire array is partitioned into nine non-overlapping **sub-arrays**. Each sub-array is of size 3-by-3 and constitutes $1/9^{th}$ of the entire array. The nine sub-arrays respectively start at: row-1 and col-1, row-1 and col-4, row-1 and col-7, row-4 and col-1, row-4 and col-4, row-4 and col-7, row-7 and col-1, row-7 and col-4, and row-7 and col-7. The first sub-array has cells at the top-left part of the array and includes cells (1, 1), (1, 2), (1, 3), (2, 1), (2, 2), (2, 3), (3, 1), (3, 2), and (3, 3). The sixth sub-array includes the cells (4, 7), (4, 8), (4, 9), (5, 7), (5, 8), (5, 9), (6, 7), (6, 8), and (6, 9). The ninth sub-array is the $1/9^{th}$ of the entire array at bottom-right and includes cells (7, 7), (7, 8), (7, 9), (8, 7), (8, 8), (8, 9), (9, 7), (9, 8), and (9, 9).

Sudoku is governed by the following simple rules.

1. Each cell must take one of the following nine possible **values**: 1, 2, 3, 4, 5, 6, 7, 8, or 9.
2. Each value must appear exactly once in each row.
3. Each value must appear exactly once in each column.
4. Each value must appear exactly once in each sub-array.

We will only consider the most common type of Sudoku puzzles, where each puzzle has a unique solution.

## 2 Your tasks for this assignment

1. Develop a number of **implication rules** that embody above rules. For example, rules 1 and 2 above, might lead to an implication rule: "If the value 4 is already assigned to one cell in row-7, then no other cell in row-7 can have value 4 as an option."
2. Develop an **implication algorithm** that uses above implication rules. In particular, decide the order in which you will apply the rules, the conditions under which you will repeat application of a rule, and so on.
3. Implement your algorithm.

   To streamline this task, we have provided a C program that has the procedures to read a given Sudoku puzzle (in a given format) and to create data-structures that are appropriate for your implementation. The given program also includes procedures for printing results.
4. Run your implementation on a set of Sudoku puzzles that we have provided, and gather results generated by your program.

## 3 Output format of Sudoku program

- Store all 16 answers in 1 file. (File name: Firstname_Lastname_StudentID_ans.txt)
  - e.g.) Sandeep_Gupta_123456789_ans.txt
- First line of the file: 1st problem file name (same as given file name)
- From second line, express Sudoku array **same** format as output format of "implication_sudoku.c"
- After printing 1st array, print 2nd problem file name.
- Print 2nd array.
- Print all (**solved/partially solved)** arrays.

e.g.)

```
websudoku-1022506735-easy.txt
  *1*   *7*   *3*  |  *5*   *4*   *2*  |  *8*   *6*   *9*
  *8*   *6*   *4*  |  *9*   *3*   *7*  |  *5*   *1*   *2*
  *9*   *5*   *2*  |  *8*   *1*   *6*  |  *4*   *3*   *7*
-------------------------------------------------------------------------------
  *6*   *1*   *7*  |  *3*   *9*   *4*  |  *2*   *8*   *5*
  *2*   *9*   *5*  |  *7*   *8*   *1*  |  *6*   *4*   *3*
  *3*   *4*   *8*  |  *2*   *6*   *5*  |  *7*   *9*   *1*
-------------------------------------------------------------------------------
  *5*   *8*   *1*  |  *6*   *2*   *9*  |  *3*   *7*   *4*
  *4*   *2*   *6*  |  *1*   *7*   *3*  |  *9*   *5*   *8*
  *7*   *3*   *9*  |  *4*   *5*   *8*  |  *1*   *2*   *6*
websudoku-506660453-evil.txt
…
```

- You can print any array first as long as the title and following array matches.
- You don't have to add array if you can't solve the problem at all.
- The file name printing on the answer file has to be **exactly** same as the one given in the problem.

## 4 How to submit

- Prepare a complete report that summarizes your implication rules, implication algorithm, key aspects of implementation, experimental results, analysis of results and observations, and thoughts about future enhancements and hand it to Professor in class by **Sep 6, 8:00am.**

- Make a folder “HW#1_ Firstname_Lastname_StudentID”
- In this folder you should have above answer file.
- You should also include executable file (a.out) and your source code (xxx.c).
- Do not include Sudoku problem files even though they should be in the same folder for executing.
- Insert “readme.txt” in the folder describing instruction of executing your program.
- Zip the folder (so the file name is same as folder name) and submit to “HW 01” under “Assignments” link in EE658_20123 den webpage at “https://www.uscden.net/”.