Alexandria University
Faculty of Engineering
Computer and Communication Engineering
Specialized Scientific Programs

Alexandria University Logo

CC272
Programming II
Fall 2025/2026

Lab 9: Sudoku Solution Verifier

Table of Contents

In this assignment you will implement Sudoku solution verifer in three modes of parallization.

Input

Your Java Program would be prompted by .csv file which is a Sudoku Solution.

NOTE you don’t have to sanitize the input file. It would be 9x9 ranging (1-9).

User would prompt also one of three modes **0**, **3**, **27**.

- **0**: means one thread (main thread) in other words a regular sequential program.
- **3**: means four threads, three added to the main, one per each type: one for rows, one for columns, and one for boxes.
- **27**: means twenty eight threads, twenty seven added to the main, one per each (row or column or box), we got 9 rows, 9 cols, and 9 boxes which sum up to 27 threads.

Requirments

So here is the task, your Java program would verify that the prompted 9x9 board which is filled with digits ranging of (1-9) that it is a valid Sudoku Solution.

TIP knowing the original game won’t affect the verification, our question is: Could this board be a solution to some Sudoku game (Sg) even though we don’t know this Sg?

Technically speaking, you are checking that all rows, columns, and boxes have no duplication of digits (1-9).

In case of having duplicates you have to locate them, therefore returning `INVALID` is not sufficient.

TIP Program have to process the whole board in the two cases (valid/invalid), no shortcuts.

TIP Start first with the sequential task, as it is the straightforward one, then move to 3-mode, lastly, 27-mode. Move by refactoring the previous one respectively.

Test Cases

Table 1. Valid solution that highlights centered row, centered column, and centered box

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

Table 2. Invalid Sudoku solution with 1 in every cell

1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1	1

For table Valid solution that highlights centered row, centered column, and centered box, your program would be expected to print `VALID`.

For table Invalid Sudoku solution with 1 in every cell, your program would be expected to print:

```
INVALID

ROW 1, #1, [1, 2, 3, 4, 5, 6, 7, 8, 9]
ROW 2, #1, [1, 2, 3, 4, 5, 6, 7, 8, 9]
.
.
.
ROW 9, #1, [1, 2, 3, 4, 5, 6, 7, 8, 9]
------------------------------------------
COL 1, #1, [1, 2, 3, 4, 5, 6, 7, 8, 9]
COL 2, #1, [1, 2, 3, 4, 5, 6, 7, 8, 9]
.
.
.
COL 9, #1, [1, 2, 3, 4, 5, 6, 7, 8, 9]
------------------------------------------
BOX 1, #1, [1, 2, 3, 4, 5, 6, 7, 8, 9]
BOX 2, #1, [1, 2, 3, 4, 5, 6, 7, 8, 9]
.
.
.
BOX 9, #1, [1, 2, 3, 4, 5, 6, 7, 8, 9]
```

NOTE In INVALID case, output represents all duplicated values in all ROWs, COLs, and BOXs showing their locations only. Unique or missing values are not printed.

Guide

> *Premature optimization is the root of **all** evil (or at least most of it in programming).*
>
> — *Donald Knuth*

- (`Optional`) This task doesn’t require concurrency; you implemented it sequentially, unlike, for example, competitive games. You also implemented parallel reduction in several modes here, which in real-world applications might be done for performance reasons. Which mode do you think best fits this task in terms of performance?

Bonus

You developed GUI apps in the previous labs, for this one we want a CLI one like commands you prompt on *git bash*.

1. Extract *.jar* from your app.
2. Run your program with `java -jar <app-name>.jar <.csv filepath> <mode>`.

Required

- You have to use one creational design pattern.
- You are required to obey OOP concepts (inheretence, abstraction, …).
- A discussion would be made with you at your lab next week on what you delivered.
- The deadline for delivery on the form is Monday 02/12/2025.

Deliverables

- One of the team members should submit to the form with zip file of .java files.
- File should be named e.g. 1111_2222_3333_4444_G2.

Policy

- 60% of the evaluation will be based on functionality and 40% on the discussion.
- Plagiarism will result in at least two full labs being canceled, so getting a zero is better than cheating.
- You may work in teams of up to **four** students.
- No late submissions are allowed.
- Each group must use GitHub throughout their lab work.

Dr. Layla Abou-Hadeed

Eng. Ahmed ElSayed
Eng. Ahmed Ashraf
Eng. Miar Mamdouh
Eng. AbdElaziz Mohamed

Eng. Mazen Sallam
Eng. Shams Zayan
Eng. Abdelrahman Wael
Eng. Mohamed Zaytoon
Eng. Muhannad Bashar