Combinatorial Problem Solving (CPS)

Laboratory. Constraint Programming. Latin Square Completion Problem.

In this session we will attack the *Latin square completion problem*. A *Latin square* is a square matrix $n \times n$ of integer numbers between 1 and $n$ such that every row and every column contain all numbers between 1 and $n$. The Latin square completion problem consists in, given a square matrix $n \times n$ which is partially filled, to determine if it is possible (and how) to fill the empty cells so that the resulting matrix is a Latin square.

For example, below you can see an instance of a Latin square completion problem on the left, and a possible solution on the right.

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

1. Write a program with Gecode that solves the Latin square completion problem. You are free to use the variable and value selection strategies that you think will yield the best performance.

The instance should be read from the standard input. The input format consists in $n$, followed by $n$ lines with $n$ blank-separated strings of numbers or a dot . representing an empty cell.

The output should start with the input data, and then the filled Latin square should follow.

For example, according to this format the above instance would be written as shown on the left, and a possible output is given on the right:

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

```
9
. . . . . 2 3 . 7
. . . . 6 4 5 .
1 . . 9 3 . . .
. . . . 6 1 8 .
. . . . 4 8 . . .
. . . . 6 4 2 .
. . . . 7 5 . . .
. 2 9 1 . . . .
4 . 5 6 . . . .
```
You can find instances at the website of CPS (http://www.cs.upc.edu/~erodri/cps.html). For all these instances, it is guaranteed that there exists a solution. You will also find there a check program that you can use to validate your solutions. For example, assuming that your program is named `program`:

```
$ program < instance-4.txt | check
OK!
```

2. Evaluate the performance of your program on the instances. You can call `time bash run.sh` where `run.sh` is the script below, available at the website of CPS:

```
for instance in instances/instance-*.txt; do
    time program < $instance | check
    echo "................"
done
```

3. Now implement a program that solves the Latin square completion problem with a backtracking algorithm on your own, without using Gecode. Evaluate its performance and compare it with the constraint programming-based program that uses Gecode.