C++ for Finance - Assignment 1 - 10th December 2019 (due: 12 noon 21st January 2020)

Laurence A. Hurst - University of Birmingham - Academic year 2019-20

Your assignment

For your assignment you need to complete the following coding project, and then write a short report on it. Both the code and your report will be marked and both must be submitted before the deadline.

Coding task

Your task is to write, in C++ utilising any language features available in C++14 (as taught in this module) and its standard library, a 9x9 grid (with 3x3 regions) Sudoku puzzle solver. You may not use any libraries that are not in the C++14 standard library in your solution (everything shown to you, so far, in the module is part of the C++ standard library).

The Sudoku puzzle is described below, if you are not familiar with it already.

There are numerous different approaches to computationally solving this problem, including brute-force, stochastic methods, constraint/logic solvers and exact cover. Choosing an approach is part of your task (see the report, below).

Things to consider:

- Not all Sudoku puzzles given to your program may be solvable.
- There maybe multiple solutions to the Sudoku puzzle given to your program.

(both of these cases are considered not to be "proper" Sudoku puzzles, but that does not prevent them from being fed into your program and your program should be able to manage these scenarios.)

Your solution must read in the Sudoku puzzle from a seperate file (i.e. the puzzle must not be hard-coded into your program, your program must be able to be repeatedly run with differente puzzles without re-compilation). The format of this file is up to you to devise (but must be documented - see report below).

This task can be completed using only what you have been taught on this course so far.

Report task

Alongside your code you must submit a short report about your solution. At a minimum it must address each of the following:

1. Briefly explain your approach to the problem and why you have chosen to implement that as your solution over any others you may have thought of or researched.
2. A summary of the algorithm as implemented in your code (you might find it helpful to include a mathematical description and/or a flow diagram).
3. An explanation of how efficient you think the **algorithm** you have used is and any efficiency considerations.
4. A reflection on the efficiency of your implementation (specifically your code/implementation, not the efficiency of the algorithm you have used) and any considerations, including any investigations/tests you may have conducted to asses this.

Please make sure to address the last two points individually.

The Sudoku puzzle

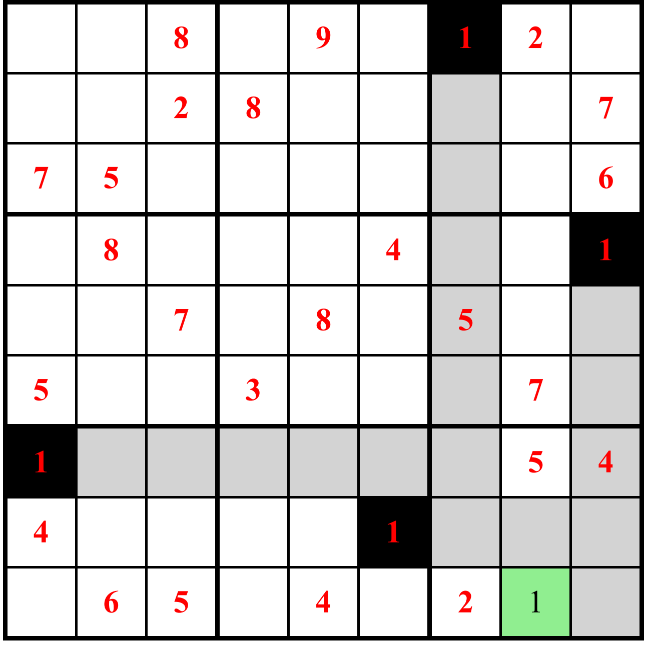
The Sudoku puzzle is a logic-based puzzle, the objective of which is to fill in a 9x9 grid made up of 9 3x3 grids:

You are given a starting set of numbers (no fewer than 17 as it has been proven that no Sudoku with less than 17 starting digits is logically solvable(McGuire, Tugemann and Civario, 2014)) positioned on the grid:

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

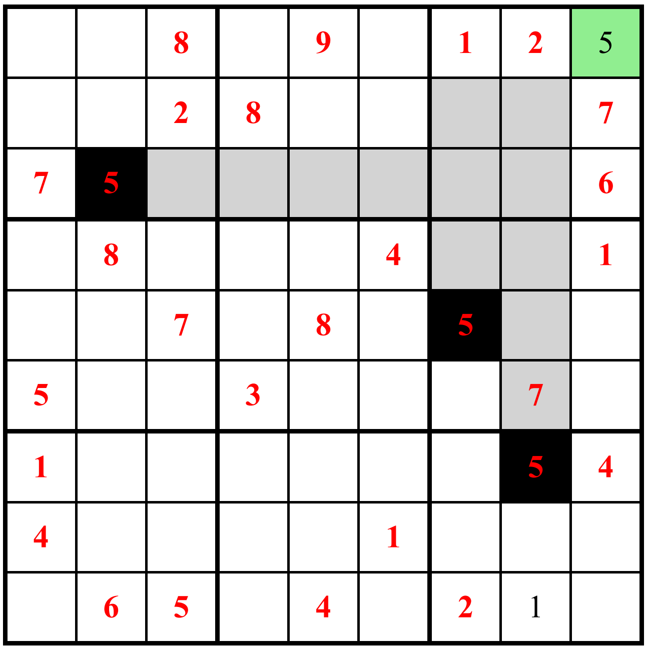
From this starting position, the puzzle is to complete the grid by putting a single digit in each empty square such that each row, column and marked 3x3 region contains exactly the digits 1-9, in any order. No digit maybe repeated in any row, column or marked 3x3 region.

For example, in the above starting grid, one possible initial step is to complete 1 in the highlighted cell, which must be a 1 because the other highlighted 1s mean none of the other spaces in that 3x3 region can contain a 1 (as every 3x3 region, column and row must contain exactly one of each digit):



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

and, for the same reasoning, the top right space must be a 5:



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

When solving these puzzles by hand it is often easiest to note down the positilities for each space and eliminate them (cross or rub them out) as the puzzle progresses until only one possibility remains in each space.

Assessment

This project is worth 40% of your overall mark for this module. Of that 40%, 70% of the marks (28% of your overall mark for the module) will come from your code and 30% (12% of your overall mark for the module) from your report.

What you must hand in, by upload to Canvas

You must upload two files to Canvas by the project deadline:

1. A zip file containing all of the code for your project - refer to the Exercise 1 from week 5's exercise sheet for what you need to include with your submission (n.b. do not share your project code with your classmates).
2. A Microsoft Word (doc or docx) or PDF file containing your report.

The marks for your code will be split between functionality (it solving the problem set and the efficiency of the algorithm used) and the quality of the code. Code quality will be measured against what you have been taught on this course; judging style, understandability, efficiency of your implementation, maintainability and layout (this is not an exhaustive list of critera, nor is it an indication of the importance of any one aspect, but to give you an idea of what is being looked for).

Additional Important Information

Your report, code and any comments in your code must be written in English.

The code you upload must run unmodified in the CLion IDE as installed in the teaching lab computers in G04 - this is the environment we have been using throughout this course.

Your report file must be print-ready, it is your responsibility to ensure the report can be printed correctly without modification.

Your report must have a title page giving the assessment title and your university number.

In your report you may refer to your code by filename and line number(s), for example "line 30 in main.cpp" or "lines 30-35 in main.cpp", or include small code snippets if you wish. Make sure the line numbers are correct before you submit, especially if you have changed any code since writing your report.

Your report body must be no longer than three A4 pages (including any and all diagrams, tables and code snippets but not including the title page), using either Calibri or Times New Roman font and a minimum font size of 11pt.

The name of the report, and the zip file must be your university number. Rename the internal C++ project name to be your ID number. Your university number must also appear on the title page of the report and in a comment at the top of each .hpp and .cpp file. (marking is anonymous so your name must not appear anywhere).

The code in the zip file will be compiled and run to confirm it functions and correctly solves sample Sudoku puzzles.

To reiterate: **your student ID must appear in every source code file**.

Group work is not permitted

You are encouraged to discuss the assignment and ideas for approaches with other students but all code you submit must be yours alone. Clear similarities in code or in the report will be taken as evidence of group work (for instance if identical or very similar results for your assessment of your code are given, or if code is clearly shared).

Code closely modelled on that from other sources is not permitted

Code to perform this exercise, in C++ or other languages, may be found in various places and might, for instance, be downloadable from the internet. Use of any code taken, or cosmetically altered, from such sources (or elsewhere) is not permitted (with or without attribution). You must devise your own code from scratch.

Specifically excluded from this prohibition is code and examples in the lecture materials (including the exercise sheets and solutions), which you may refer to and model parts of your solution on without attribution.

You may, and are encouraged to, research general approaches to this task - but you must not use, or model your work on, any existing code.

References

McGuire, G., Tugemann, B. and Civario, G. (2014). There Is No 16-Clue Sudoku: Solving the Sudoku Minimum Number of Clues Problem via Hitting Set Enumeration. Experimental Mathematics, 23(2), pp.190-217.