Further Computer Programming Coursework assignments

Introduction

Sudoku is a number placement puzzle- the objective is to fill a 9x9 grid with digits such that:

- Each column contains all the digits from 1-9
- Each row contains all the digits from 1-9
- Each 3x3 sub grid (aka "box") contains the digits from 1-9

Typically, players are given a partially completed grid and asked to fill in the remaining spaces. For example, players may be given the grid in Figure 1A. Players would then complete the puzzle by adding digits until all squares are filled and the three constraints above are satisfied. The solution to the puzzle in Figure 1A in show in Figure 1B.

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

Figure 1A: An example sudoku problem

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

Figure 1B. The solution to the problem in Figure 1A

In this coursework, you will work as a group to develop a computer program that can solve Sudoku puzzles

This coursework will assess your ability to:

- Use Python to solve problems.
- Write Python code that adheres to a given specification.
- Work with a group to develop code collaboratively.
- Make effective use of version control tools such as Git.

Further Computer Programming Coursework assignments

Assignment (summative, group)

Due Date: May 5th

Submission: This is a group project to be completed in groups of 4 students. You should work together to produce a single codebase, but each should contribute some individual parts of the code. You will produce a joint submission consisting of:

- Code (submitted as a single .zip file on Blackboard)
- Reflection Report (submitted individually on Blackboard, including a link to a shared GitHub repo)

Your code for tasks 1-3 should be integrated into a single program which can be run from the terminal. You should include a README file which explains how to run your code.

Task 1 (30 points): Improve the provided recursive solver so that it can handle larger grids (3x2 and 3x3). We will provide a basic backtracking solver [available **on blackboard** after March 17th]. Broadly, there are two strategies you can use to improve the provided solver.

1. Currently, we try all values (from 1 to n) in our selected empty location. However, many of these values can be eliminated by looking at what is already in the corresponding row, column and square.
2. Our recursive search always tries to fill the first empty location it finds. However, it turns out that some spots are much easier to fill than others (because they have fewer possibilities). A better approach would be to select the empty location with the fewest number of viable options to fill.

Assessment: We will run your code against 5 3x2 grids and 5 3x3 grids (of increasing difficulty). You will receive 1.5 point per correctly solved Sudoku grid. A further 15 points are available for writing clear, well commented code, conforming to the unit style guide, and appropriate use of git to work collaboratively.

Task 2 (40 points): Extend your code to include:

1. Include a flag –explain which as well as printing out the answer, also provides a set of instructions for solving the puzzle (i.e as a list of commands such as "Put 2 in location (3, 4)".
2. Include a flag –file INPUT OUTPUT which reads a grid from a file (specified with the first argument), solves it, and then saves the solution to another file (specified as OUTPUT). If the –explain flag is included, your solution should also write the explanation to the output file.
3. Include a flag –hint N which rather than giving the full solution, instead returns a grid with N values filled in. This flag should also interact correctly with the –explain and –flag files from tasks 1 and 2 (i.e if I call your code with –explain -hint 3, it should return a grid with 3 squares filled and a set of instructions from getting from the original grid to the new grid).
4. Include a flag –profile which measures the performance of your solver(s) (in terms of time) for grids of different size (2x2, 3x2, 3x3) and difficulties (specified by number of unfilled locations). This should average the performance over a few solution attempts and starting conditions, summarising the results as a plot(s). Plots should be clearly labelled, with appropriate ranges, axes labels and legends.

You will also need to write a main function that acts as an entry point to your code, parsing these flags and selecting the appropriate behaviour.

Further Computer Programming Coursework assignments

Assessment: You will receive 5 points for completion of each task 1-4 (i.e a total of 20 for all 4). A further 20 points are available for writing clear, well commented code, conforming to the unit style guide, and appropriate use of git to work collaboratively.

Task 3 (30 points): So far, our solver has represented a Sudoku grid as a nested list. We have used the character '0' to represent an unfilled grid location, changing that to an integer when we place a value in the grid. An alternative way to represent an empty location would be to instead store a list of *possible* values for each grid location. For example, if a grid location is empty, it would be represented as [1, 2, 3, 4, 5, 6, 7, 8, 9] as any value could go there. As values are placed into that location's, row / column / square, we eliminate those values from the list of possibilities. Using this representation, a solver could work by iteratively finding locations with only a single possible value and then eliminating that value from that location's row / column / square. If no locations have only a single value (and the grid is not solved), then choose the location with the smallest number of possible values, select one randomly and begin eliminating again. This approach is known as "Wavefront propagation". Your task is to implement this alternative solver and compare its performance to the solver from task 1.

Assessment: You will receive 15 points for a successful implementation of the Wavefront propagation algorithm. A further 15 points are available for writing clear, well commented code, conforming to the unit style guide, and appropriate use of git to work collaboratively.

Task 4: You **must** also submit an individual report, reflecting on the project, your individual contribution, and your success working as a team. The template for the reflective report is available **on Blackboard**. Note that as part of your reflective report, you will be asked if you are happy to share the same mark among all members of your group. If everyone agrees, then everybody in the group will get the same mark. Otherwise, each student will get a different mark based on our assessment of individual contribution. This will be based on engagement during group meetings, commits recorded in git, and the contents of the reflection reports. **Note that failure to submit an individual report will result in a mark of zero for the project.**