ARTIFICIAL INTELLIGENCE

Assignment # 1

Deadline: 21 Feb, 2023

Instructions:

1. Assignments are to be done individually. You must complete this assignment by yourself. You cannot work with anyone else in the class or with someone outside of the class.
2. You have to use python programming language.
3. **Plagiarism of any kind (copying from others and copying from the internet, etc.,) is not allowed and can result in zero marks in whole assignment category.**
4. Your code must be properly commented.
5. Any assignment marked late by google will be considered late.
6. No marks will be assigned if any of the following deliverables are missing.
   a. The source code of the program.
   b. A pdf or word report containing a brief explanation of the steps involved in the program (each question) and the results obtained.
7. Put both source code and report in one folder, ZIP it and submit it. Your folder must be named as ROLLNO_NAME.ZIP

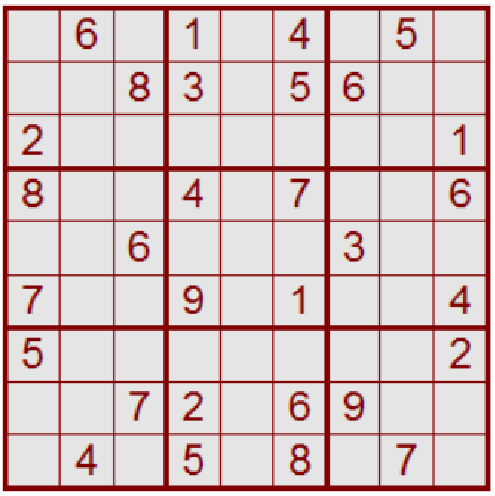
Question 1 (60 marks)

Sudoku is the Japanese word for “single numbers”, and refers to numerical puzzle game that has become popular in newspapers and game publications all over the world. Although the rules for this puzzle are simple to understand, the solutions can range from the very simple to the agonizingly difficult. This assignment will require you to develop a Sudoku solver by applying search algorithms with heuristics to solve a puzzle.

Consider the classic 9-by-9 Sudoku puzzle in Figure 1. The goal is to fill in the empty cells such that every row, every column, and every 3-by-3 box contains exactly the digits 1 through 9, each digit occurring once. The solution to the puzzle is in Figure 2, and it satisfies these constraints:

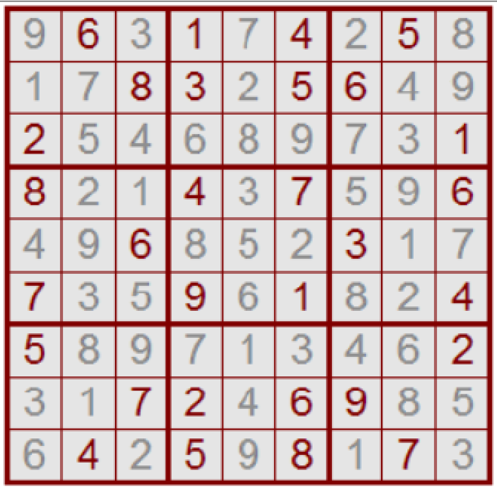
1. The digits to be entered are 1, 2, 3, 4, 5, 6, 7, 8, and 9.
2. A row is 9 cells wide. A filled-in row must have exactly one of each digit. There is 9 rows in the grid, and the same constraint applies to each of them.
3. A column is 9 cells tall. A filled-in column must have exactly one of each digit. There are 9 columns in the grid, and the same constraint applies to each of them.
4. A box contains 9 cells in a 3-by-3 layout. A filled-in box must have exactly one of each digit. There are 9 boxes in the grid, and the same constraint applies to each of them

You are to write programs to generate and solve classic 9x9 Sudoku puzzles. Note that every correct Sudoku puzzle has only one correct solution (see Figure 2). However, because you will generate random puzzle instances, they may have zero, one, or more solutions.



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

Figure 1: Sudoku puzzle



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

Figure 2: Solution

Question 2 (40 marks)

The magic square is a square matrix, whose order is odd and where the sum of the elements for each row or each column or each diagonal is same. The sum of each row or each column or each diagonal can be found using this formula. n(n2+ 1)/2.

Consider the following example puzzle called "Moving Magic Square". It is played on a 3 × 3 table containing each of the numbers 1 to 9. The number 9 is the "movable number". You can move 9 in four directions (up/down/left/right), and swap 9 with the number in that direction. The initial state is shown in Table 1. As the player, we want to move 9 to reach a final state such that the sum of the three numbers on every row, column, and diagonal is 15. There are multiple states that satisfy this condition, and you can stop your answer when you find the first satisfied state.

6	9	8
7	1	3
2	5	4

Table 1: Initial state

You are to write programs to generate and solve classic 3x3 magic square puzzles. However, because you will generate random puzzle instances, they may have zero, one, or more solutions.

For both questions your solutions should be generic and have to use the fastest possible search algorithms more than one and also calculate the running time complexity and space complexity of your programs.