LAB 4: Sudoku - Constraint Satisfaction Problems

In Constraint Satisfaction Problems (CSPs), the goal is to find a complete, consistent assignment of values to a set of variables X (taken from their domains D) satisfying a set of constraints C that limit the valid combinations of variable values. In this assignment, you will have an opportunity to develop a program using CSP solution techniques to solve Sudoku puzzles.

Sudoku (Japanese meaning number place) puzzle is a 9x9 grid (81 variables) where each cell in the grid can take on the integer values 1-9 (the domain of each variable). A solution to a Sudoku puzzle is an assignment of values for each cell in the grid such that no two cells in the same row, column, or 3x3 square have the same value.

For example, for an initial configuration of a Sudoku puzzle, you might be given:

```
003|020|600
900|305|001
001|806|400
-----------
008|102|900
700|000|008
006|708|200
-----------
002|609|500
800|203|009
005|010|300
```

which has the solution:

```
483|921|657
967|345|821
251|876|493
-----------
548|132|976
729|564|138
136|798|245
-----------
372|689|514
814|253|769
695|417|382
```

The code you will be using can be downloaded as a zip archive on Blackboard, namely `sudoku`.

Your assignment is to write a program in Python that can take a set of Sudoku puzzles as input from a file, models each puzzle as a CSP, and outputs solutions to each puzzle.

The provided `.zip` folder contains several files, in which two files wihtin `/data` sub-folder are Sudoku puzzles:

1. data/euler.txt, a set of Sudoku puzzles from Project Euler https://projecteuler.net/problem=96 (https://projecteuler.net/problem=96)
2. data/magictour.txt, a more difficult set of Sudoku puzzles from http://magictour.free.fr/top95 (http://magictour.free.fr/top95)

Each file contains a multiple Sudoku puzzles (one per line), in the following format:

- Each line is a string of 81 characters, where characters in positions 0-8 correspond to the first row of the puzzle, characters in positions 9-17 correspond to the second row of the puzzle, etc.
- Known values are represented by the digits 1-9.
- Initially unknown values are represented by digit 0.

And other files that you can and cannot modify as below:

Files you'll edit:

File Name	Description
`search.py`	Where all your search algorithms will reside.
`csp.py`	Class description for constraint satisfaction problem.

Files you should look at but NOT edit:

File Name	Description
`util.py`	Useful data structures for implementing search algorithms.
`sudoku.py`	The main file that runs to solve all Sudoku problems.

Exercise 1 (30 points): Implement the constraint satisfaction problem in the initializing function in the csp.py.

- How did you represent the Sudoku puzzle a CSP?
- What design options did you consider, and how did you decide on this implementation?

Exercise 2 (35 points): Implement Backtracking Search algorithm in the search.py.

Exercise 3 (35 points): Implement AC-3 search algorithm.

Your program should be able to read in these puzzles, solve them, then output the solutions in the same format (a string of 81 digits, followed by a newline character) in the same order they were read in from file.

Commands to run code:

```
python sudoku.py --inputFile ./data/euler.txt --outputFile euler_output.txt
```

```
python sudoku.py --inputFile ./data/magictour.txt --outputFile magictour_out
put.txt
```

What to submit

1. The solutions to all 145 puzzles in the same format as the input files
2. A short paragraph describing your experience during the assignment (what did you enjoy, what was difficult, etc.)
3. Source code.
4. Please create a folder called "yourname_StudentID_Lab4" that includes all the required files and generate a zip file called "yourname_StudentID_Lab45.zip".
5. Please submit your work (.zip) to Moodle.