CS 6613 **E. K. Wong**
Project 2: Cryptarithmetic Problem **Fall 2023**

Total number of points = 100.

Project Description: Design and implement a program to solve Cryptarithmetic problems as shown in the figure below:

$$\begin{array}{cccccc} & x_1 & x_2 & x_3 & x_4 \\ + & x_5 & x_6 & x_7 & x_8 \\ x_9 & x_{10} & x_{11} & x_{12} & x_{13} \end{array}$$

where x_1 to x_{13} can be any capital letter from A to Z; some letters may occur more than once. Each letter stands for a distinct digit; the aim is to find a substitution of digits for letters such that the resulting sum is arithmetically correct, with the added restriction that no leading zeros are allowed. The domain for x_9 is therefore {1}, the domain for x_1 and x_5 is {1,2, … , 9} and the domain for variables x_2 to x_4, x_6 to x_8, and x_{10} to x_{13} is {0,1,2, … , 9}. You can introduce auxiliary variables and specify their domains to represent carry overs from previous columns. After this, you can set up a set of constraints for the problem.

Implementation: Implement the *Backtracking Algorithm for CSPs* in Figure 1 below to solve this problem. Implement the function *SELECT-UNASSIGNED-VARIABLE* in the algorithm by using the *minimum remaining values* and *degree heuristics.* Instead of implementing the *least constraining value* heuristic in the *ORDER-DOMAIN-VALUES* function, simply order the domain values in increasing order (from lowest to highest.) You can skip the implementation of the INFERENCE function.

Input and output files: Your program will read in values from an input text file and produce an output text file that contains the solution. The input file contains three rows (or lines) of capital letters:

LLLL
LLLL
LLLLL

The first and second rows contain four capital letters and the third row contains five capital letters with no blank space between letters. The output file should follow the following format

DDDD
DDDD
DDDDD

where the Ds represent digits from 0 to 9 with no bank space between the digits.

Teammate: You can work on the project by yourself or form a team of two students to work on the project. You can discuss with your classmates on how to do the project, but every team is expected to write their own code and submit their own project.

Testing your program: Two test input files will be provided on Brightspace to test your program. You can also create your own test files. A 10-minute time slot will be set up during class time to test your program on additional test files that will be provided in class. (Details to be announced later.)

CS 6613 **E. K. Wong**
Project 2: Cryptarithmetic Problem **Fall 2023**

Recommended languages: Python, C++/C or Java. If you would like to use a different language, send me an email first.

Submit on Brightspace:

- Your source code file. Put comments in your source code to make it easier for someone else to read your program. Points will be taken off if you do not have comments in your source code.
- The output files generated by your program for the input test files.
- A PDF file that contains instructions on how to run your program. If your program requires compilation, instructions on how to compile your program should also be provided. Also, copy and paste your outputs and your source code onto the PDF file (to make it easier for us to grade your project.) This is in addition to the source code file and output files that you have to hand in separately, as described in items (1) and (2) above.

If you work in a team of two, only one partner needs to submit but please write both partners' names on the source code and the PDF report.

```
function BACKTRACKING-SEARCH(csp) returns a solution or failure
  return BACKTRACK(csp, { })

function BACKTRACK(csp, assignment) returns a solution or failure
  if assignment is complete then return assignment
  var ← SELECT-UNASSIGNED-VARIABLE(csp, assignment)
  for each value in ORDER-DOMAIN-VALUES(csp, var, assignment) do
    if value is consistent with assignment then
      add {var = value} to assignment
      inferences ← INFERENCE(csp, var, assignment)
      if inferences ≠ failure then
        add inferences to csp
        result ← BACKTRACK(csp, assignment)
        if result ≠ failure then return result
        remove inferences from csp
      remove {var = value} from assignment
  return failure
```

Figure 1. The Backtracking Algorithm for CSPs