Problem 1: Parse Tree and Leftmost Derivation

Using the grammar below, show a parse tree and a leftmost derivation for the following statement:
A = B * (C * (A + B))

Grammar:
Statement → Assignment | Expression
Assignment → Identifier = Expression
Expression → Expression + Term | Term
Term → Term * Factor | Factor
Factor → (Expression) | Identifier
Identifier → A | B | C

Problem 2: Scope Concepts

Considering the following program written in pseudocode:

```
int u = 42;
int v = 69;
int w = 17;

proc add( z:int )
  u := v + u + z;

proc bar( fun:proc )
  int u := w;
  fun(v);

proc foo( x:int, w:int )
  int v := x;
  bar(add);

main
  foo(u, 13);
  print(u);
end;
```

a. Using Static Scope, what is printed to the screen?

b. Using Dynamic Scope with Deep Binding, what is printed to the screen?

Hint: The sum for u is 126, but due to deep binding, it's foo's local v that gets involved.

c. Using Dynamic Scope with Shallow Binding, what is printed to the screen?

Hint: The sum for u is 101, but again it's foo's local v that matters.

Problem 3: Sudoku Solver

Write a program to solve a Sudoku puzzle by filling the empty cells.
A Sudoku solution must satisfy all of the following rules:
- Each of the digits 1-9 must occur exactly once in each row.
- Each of the digits 1-9 must occur exactly once in each column.
- Each of the digits 1-9 must occur exactly once in each of the 9 3x3 sub-boxes of the grid.

Example Board:
[["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"]]

Output: The only valid solution is shown below:

Example shown here: https://leetcode.com/problems/sudoku-solver/description/

Problem 4: Rubik's Cube Data Structure

Design a data structure to represent a Rubik's Cube and store its state in memory. The cube can have an arbitrary number of tiles per side (e.g., 3x3, 4x4, 5x5).

Things to Consider:

- Cube Size: The cube can be of any size (e.g., 3x3, 4x4, 5x5). Your solution should be flexible enough to handle cubes with different numbers of tiles per side.

- Layer Rotation: Since it's a Rubik's Cube, layers (rows, columns, or faces) need to be rotatable. Think about how rotating a layer affects adjacent sides and how to efficiently update the cube's state in memory.