# **Artificial Intelligence Assignment#2**

## Muhammad Kashif

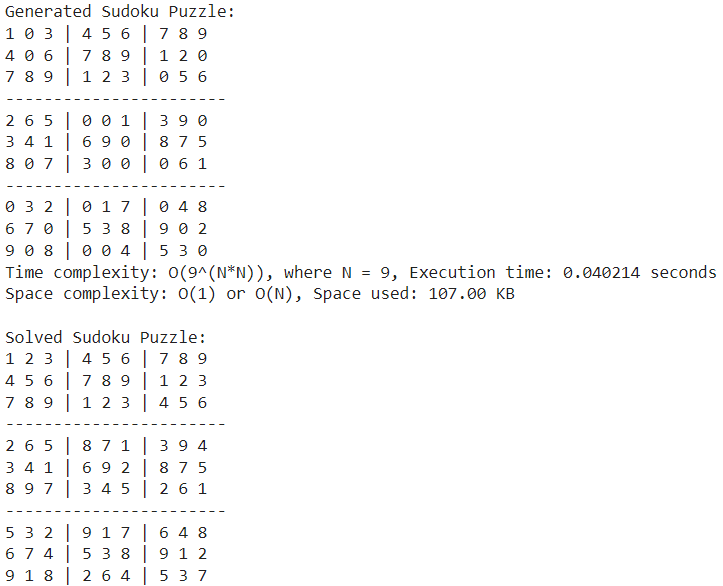
## I21-0851

## Section Y

### **Question-1**

1. **Display Grid Function**:
   * This function is responsible for displaying the Sudoku grid in a readable format with horizontal and vertical lines to separate the 3x3 sub grids.
2. **Generate Sudoku Function**:
   * Generates a random Sudoku puzzle by first creating a complete puzzle using the **solve\_sudoku** function and then removing a specified number of numbers to create a puzzle.
3. **Remove Numbers Function**:
   * Removes numbers from the Sudoku grid to create a puzzle by randomly selecting cells and setting their values to zero.
4. **Is Valid Move Function**:
   * Checks if placing a number at a specific position in the grid is valid by ensuring the number does not violate Sudoku rules in the same row, column, or 3x3 sub grid.
5. **Solve Sudoku Function**:
   * Uses backtracking with MRV (Minimum Remaining Values), Degree Heuristic, and LCV (Least Constraining Value) to solve the Sudoku puzzle recursively. It prioritizes cells with fewer remaining valid moves and explores options based on the least constraining values.
6. **Find Empty Cells Function**:
   * Finds and returns a list of empty cells (cells with value zero) in the Sudoku grid.
7. **Get Valid Moves Function**:
   * Returns a list of valid numbers that can be placed at a specific position in the grid without violating Sudoku rules.
8. **Count Least Constraining Values Function**:
   * Counts the least constraining values for a given number at a specific position in the grid. It considers the impact of placing the number on other empty cells in the same row and column.
9. **Arc Consistency Function**:
   * Implements the Arc Consistency (AC3) algorithm to ensure consistency in the Sudoku grid by propagating constraints and eliminating invalid options iteratively.
10. **Main Function**:
    * Generates a Sudoku puzzle, displays it, and then applies arc consistency and solves the puzzle. It also calculates and prints the time and space complexity of the solving process.

### **Results:**



### **Question-2**

The program implements a genetic algorithm to find a 3x3 magic square whose rows, columns, and diagonals all sum up to a constant value (MAGIC\_CONSTANT = 15).

1. **Fitness Function**:
   * Calculates the fitness of an individual magic square by counting the number of rows, columns, and diagonals that sum up to the magic constant.
2. **Generate Individual Function**:
   * Generates a random valid individual (3x3 magic square) with numbers from 1 to 9 shuffled.
3. **Crossover Function**:
   * Performs crossover between two parent magic squares to create two new children by splitting and swapping rows at a random crossover point.
4. **Mutate Function**:
   * Mutates an individual magic square by swapping two random elements within the square.
5. **Evolve Population Function**:
   * Evolves the population by selecting parents based on fitness, performing crossover, and applying mutation with a certain probability.
6. **Main Function**:
   * Initializes a population of magic squares and evolves them over generations until a solution (a magic square with maximum fitness) is found or until reaching the maximum number of generations (MAX\_GENERATIONS = 1000).
   * Prints the solution if found, along with the generation number.
   * Calculates and prints the time elapsed and memory used during the execution of the genetic algorithm.

### **Results:**

