MINISTRY OF HIGHER EDUCATION AND SCIENTIFIC RESEARCH

UNIVERSITY OF: Dgilali Bounaama -khmis miliana

FACULTY OF: Computer Science and matematics

DEPARTMENT OF: Computer Science

PROJECT TITLE: SUDOKU SOLVER USING C LANGUAGE

PREPARED BY:

NAME: Ihcene kerloufe

LEVEL: 1 LICENSE

SUPERVISED BY:

PROFESSOR: AMINA MOUDJAR

ACADEMIC YEAR:2025/2026

Project Introduction

The goal of this project is to develop a program in **C language** that can solve a **Sudoku puzzle** using the **Backtracking algorithm**. The program fills the empty cells while following all the rules of Sudoku:

- Each row must contain numbers 1–9 without repetition.
- Each column must contain numbers 1–9 without repetition.
- Each 3×3 subgrid must contain numbers 1–9 without repetition.

---

2. Solving Approach

The solution is based on the following steps:

1. Read a 9×9 Sudoku grid from the user.
2. Search for an empty cell (value 0).
3. Try placing numbers from 1 to 9 in the empty cell.
4. Check if the number placement is valid according to Sudoku rules.
5. If no number can be placed, perform **backtracking** by removing the previous number and trying the next possibility.
6. Continue this process until the puzzle is solved or it is determined that it is unsolvable.

---

3. Variables Explanation

- `grid[9][9]`: Array that stores the Sudoku puzzle.
- `row, col`: Store the coordinates of the current empty cell.
- `foundEmpty`: Flag to check if an empty cell exists.
- `valid`: Flag to check if a number is valid for the current cell.
- `solved`: Flag to determine if the puzzle can be solved.

---

4. Data Input

The program asks the user to enter the Sudoku puzzle values:

- Numbers 1–9 for filled cells.
- Number 0 for empty cells.

---

5. Displaying the Original Puzzle

The program prints the original grid as entered by the user to allow easy comparison after solving.

---

6. Searching for an Empty Cell

Two nested loops are used to find the first empty cell (value 0) in the grid.

---

7. Trying Numbers (1–9)

For each empty cell:

- Numbers 1 through 9 are tested sequentially.
- Each number is checked for conflicts in:
    - The row
    - The column
    - The 3×3 subgrid

---

8. Validating the Number

A number is considered valid if it does not already exist:

- In the same row
- In the same column
- In the same 3×3 subgrid

If valid, the number is placed in the cell.

---

9. Backtracking

If no number can be placed in the current cell:

- The cell is reset to 0.
- The algorithm backtracks to the previous cell to try the next number.

---

10. Handling Unsolvable Puzzles

If backtracking reaches the beginning and no solution is found:

- The program displays a message indicating that the Sudoku puzzle **cannot be solved**.

---

11. Displaying the Final Result

- If the puzzle is solved: The complete Sudoku grid is printed.
- If the puzzle is unsolvable: A message is displayed.

---

12. Program Features

- Implemented without using any functions.
- Uses only **conditions, loops, and arrays**.
- Follows all Sudoku rules strictly.
- Can handle unsolvable puzzles gracefully.

GitHub Repository:
https://github.com/ihsankerlouf-ops/sudoku-project

.