**Project Requirement and Specification**

**On**

**Image Processing**

Submitted to: Submitted by:

Ms. Meenakshi Mandola Hemlata Chuphal

Roll No.: 2014401

Guided by: Semester: 3rd

Ms. Meenakshi Mandola Session: 2022-2023

**DEPARTMENT OF COMPUTER SCIENCE AND ENGEENRING**

**GRAPHIC ERA DEEMED TO BE UNIVERSITY,**

**DEHRADUN**

About:

In this project I have created a Sudoku Solver for a 9\*9 sudoku in C++ language using recursive backtracking. Sudoku is a puzzle game which is a 9\*9 grid containing 81 cells having numbers filled from 1-9. Each row, column and square (9 spaces each) needs to be filled out with the numbers 1-9, without repeating any numbers within the row, column or square (3\*3 grid).

In this project user is required to enter a partially filled sudoku which they want to solve and then the sudoku solver will generate a correctly solved sudoku. If a user enters a filled sudoku or an invalid sudoku it recognises it and displays an error message. It also tells the user if the filled sudoku is correctly solved or not.

Technologies Used:

1. C++ Programming language to code the program. To solve the sudoku it uses concept of recursive backtracking. It uses recursive calling to find the solution by building a solution step by step and increasing values with time. It removes the solutions that does not give rise to the solution of the problem based on the constraints given to solve the problem.
2. Code-Blocks.

How to run this project?

To run this project on your system you must have a C++ compiler installed preferably- MinGW. You can download the MinGW compiler for C++ from their official site.

After having the setup ready, to use this program user will have to enter a partially filled Sudoku. In place of empty cells, you must enter 0. Remember, during entering the sudoku no number from 1-9 should be repeated in a row, column and sub grid (of order 3\*3). If a user enters an invalid or a filled sudoku it will show an error and it will also tell the user if the filled sudoku is correct or not. After entering the partially filled Sudoku it will display the correct solution.

Various Methods used for this project:

1. isFull (int [N][N]): This method checks if the Sudoku entered by the user is filled or not.
2. checkSudoku (int [N][N]): This method checks if the filled Sudoku entered by the user is valid or not by calling method isSafe(int [N][N], int, int, int). It checks the conditions required to solve a Sudoku.
3. checkInvalid (int [N][N]): This method tells if the partially filled sudoku entered by the user is valid or not. The invalid cases are:
4. If a number from 1-9 is repeating in a row, column or the grid.
5. findEmpty (int [N][N], int &, int &): This method checks for the empty cells in Sudoku.
6. SudokuSolver (int [N][N]): This method uses backtracking to solve the sudoku and calls isSafe(int [N][N],int ,int ,int) to check if a number from 1-9 is safe to enter in the sudoku, if not it checks for next number. If no number can satisfy the position it backtracks to the previously filled cell and tries to fill it with other numbers. It returns true is the sudoku is solved otherwise false.
7. isSafe (int [N][N], int, int, int): This method takes the number to be filled in the position also as a parameter and then checks if it is safe to be filled. Various conditions are:
8. Number to be filled should not be repeating the that same row.
9. Number to be filled should not be repeating in that same column.
10. Number to filled should not be repeating in the 3\*3 grid.
11. inRow (int [N][N], int, int): This method checks if the number to filled is not repeating in the that row.
12. inCol (int [N][N], int, int): This method checks if the number to be filled is not repeating the same column.
13. inMat (int [N][N], int, int): This method checks if the number to be filled is not repeating in the same 3\*3 grid.
14. print(int [N][N]): This method prints the correctly solved sudoku.