

# **Sudoku Solver and Generator**

To make a 'Sudoku' program using C++ was the preliminary idea to which all the team members and our JTA agreed to. Finally, after two labs and two meetings in the respective weeks, the final conceptualisation of the project has been completed and various parts have been taken up by team members.

## ***Problem Statement:***

In short, the program shall solve and generate a standard 9x9 Sudoku Grid using some basic logical and guessing (whenever required) techniques. Thus the problem is split into two linked but different parts –

### ***In case of solving a user-given incomplete Grid:***

1. If an invalid Sudoku grid, i.e. one which is unsolvable due to inconsistencies in the given digits, an appropriate error message explaining the user the problem will be displayed.
2. The program will feature an “activity box” which if activated by the user will slow down the algorithm, to show each step of solving to help a novice understand and learn the basic solving logic for a Sudoku.
3. A pencil marks scheme (if possible to be quickly and elegantly coded within the remaining time) may be implemented. Please see the attached gui\_scheme.pdf file for further details.
4. Saving and loading a puzzle from a file will also be implemented.
5. This functionality can be used for checking a Grid sourced from elsewhere or to get some hints into its solving.

### ***In case of generating a random, new and incomplete Grid for the user to solve:***

1. A difficulty level analysis shall be implemented based on the number of guesses required (higher weightage) and the number of times the logic is run (less weightage) to solve the Sudoku.
2. To make it more interesting and give it a slight twist, instead of a numerical Sudoku, various pictures – such as logos of popular brands, images of fruits, etc. – will replace numbers if the user wants so (such an option will be given).
3. A timer will also be present for the user to track his speed and proficiency in solving.
4. Again, hints will be provided for the user to get some further insight into the puzzle solving.

**Current Status** - Most of the solver is done (optimization left). A rudimentary generator algorithm has been conceived (though it produces either very easy puzzles or ones with a very unacceptably large number of possible solutions). The design of the GUI is feature-complete though a few features may not be implemented due to time constraints. As such, the coding of the GUI is in progress, delayed mainly due to problems faced installing EzWindows and getting its hands-on with its documentation.

## ***Details of the Team:***

Lab Batch: Friday (5)

Group Number: 31

Sub-groups:

### **1. Sub-group A**

Algorithm (i.e. the backend) shall be worked upon by this sub-group

**Jitendra Francis – 115090005**

Checking validity and possibility of multiple solutions for the Grid and making the generation algorithm (using the solving algorithm).

**Kandarp Khandwala – 110050005 -> Team and Sub-group Leader**

Creating the main solving algorithm using logic and guessing (by recursion) and linking grid to algorithm.

**Kaustubh Gupta – 110040005**

Conceptualising a difficulty level analysis, hints for a user and saving/loading options.

### **2. Sub-group B**

GUI (i.e. the frontend) shall be worked upon by this sub-group.

**Jagadeeshwar Goshika – 110010045**

Implementing pencil marks scheme, splash screen and activity box.

**Kheraj Ram – 110040046**

Implementing pencil marks scheme, splash screen and activity box.

**Pranit Iyengar – 11D170015**

Creating the main screen and toolbars in the interface.

**Varun Janga – 110050076 -> Sub-group Leader**

Creating the main screen and linking grid to algorithm.