CS 152 PROJECT

Name : Arijit Pramanik
Roll Number: 150020094

Name: Srijit Dutta
Roll Number: 150070046

Project Title : SUDOKU Solver

Problem Description :

We have implemented a Sudoku game in racket with an inbuilt solver for it using the graphics features. The problem instance is an unsolved Sudoku game which the user can play. Some features like 'New Game' and 'Undo' have been implemented. The 'Solve' option has been given by which the player can see the answer of the game for any given initial instance.

Implementation Idea :

The basic algorithm for Sudoku solving involving backtracking has been used. We solve the game by assigning numbers one by one to the empty cells and checking if that cell is 'safe' for that number. If the position is safe for that number,we assign that number there and recursively check if it leads to a solution. 'Undo' is handled by keeping track of the moves made by the player so far.

Backtracking is a technique that is used for constraint satisfaction problems, as in this case where we have to ensure that each cell has one of {1,2,3,4,5,6,7,8,9} and consequently, no number is being repeated across the same row, column or a 3 X 3 block. *The inspiration for this using lists was drawn from the eight queens problem as illustrated in class.* Functions for checking whether a number can be placed in a particular cell is done using inRow?, inCol? and inBox?, cumulatively, safe?. This generates a solution for the instance.

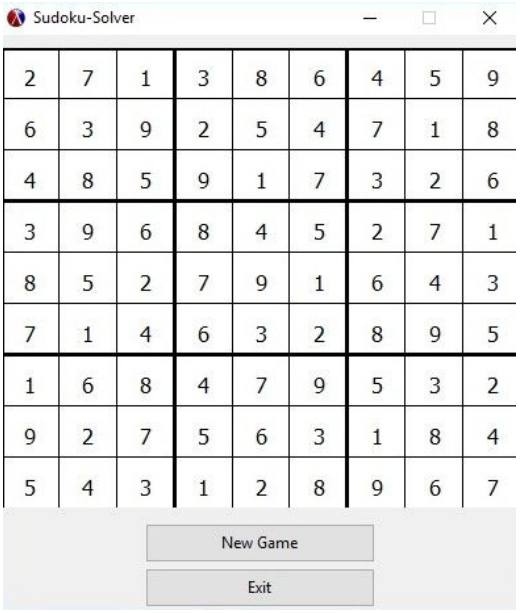
Coming to graphics, we have instantiated a frame to generate the interface for the user to play with the initial configuration, stored internally as a list of lists. The user can click a particular empty cell where he wants to put his choice. **Possible** choices for that cell until that point are shown in the form of a **radio box in a dialog box**. User selects one of them and continues playing. On pressing **Undo**, you get to erase your most recent choice. The code stores the **list of moves (i.e. the list of positions, internally in our list of lists, we put a 0)** that have been played by the user. You can undo as many

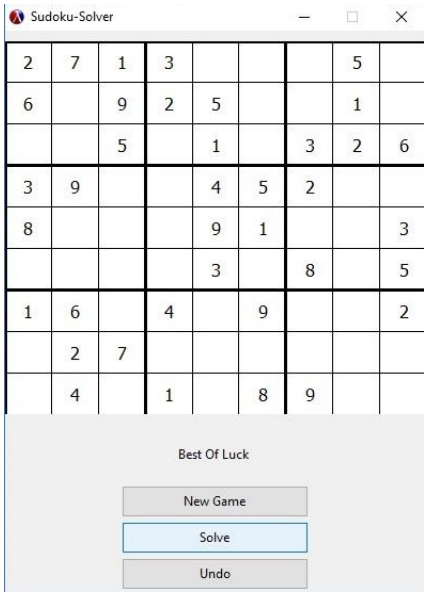
times as you want and return to your past set of choices. Once you reach a cell, where you can no longer place any number, you **do not get any choices** in the form of a radio dialog box. If you cleverly play along, and reach **successfully filling all the cells, then a "Congratulations"** message is displayed. The message is displayed above the three in-game buttons, and displays "Best of luck" on starting, "Undo" on pressing Undo.

Once you obtain the solved instance for the game, by pressing **Solve**, then you get the solution in a new frame, and can only move on to a new instance of the game, by pressing **New Game**.

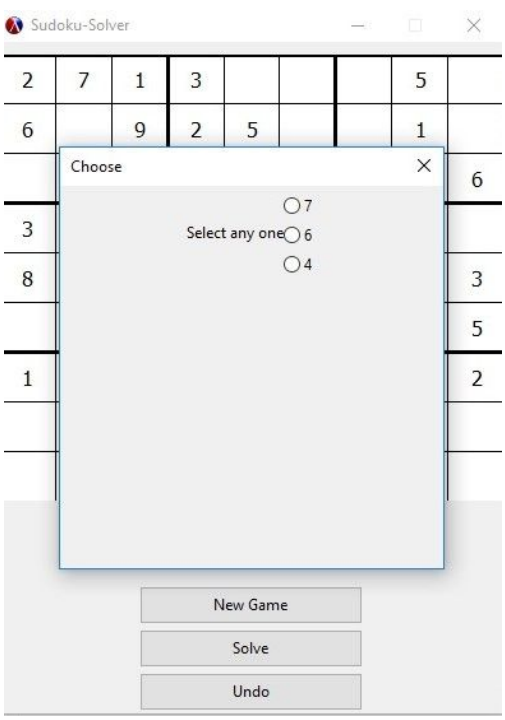
The *frame* as the parent has *panels* which serve as parents to the in-game *buttons*, namely, **New Game, Undo and Solve**. The interface is created using a **canvas**, which has a **drawing context**. We are handling all the *mouse-click events*, *button click events* with a suitable **callback** function. We are efficiently getting the *mouse-click coordinate*, and are using it to fill the cell as per our own ***devised grid and coordinate system***.

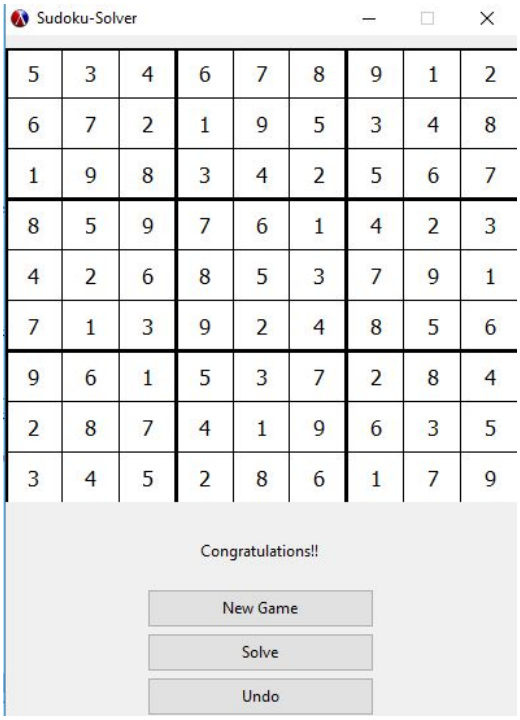
Input and Output Instances





Attached above are the screenshots for the solved frame and next is the frame that you get on compiling the code, the **initial instance, and below is the dialog box** with possible choices for that cell when you click upon a new cell**.**
The last frame is obtained when you are able to complete the entire game and the game wishes you "Congratulations".





Limitations and Bugs

We have tried our best to get rid of any bugs. Since, the coordinate system being relative, was being difficult to maintain with respect to getting mouse coordinates, the instantiated frame is of fixed size. It can't be resized to a desired size.
We have not classified the sudoku problems on the basis of their difficulties. It may be that consecutive initial randomly generated problems are of the same difficulty.

Points of interest and impact

We have made the game interface keeping in mind the widespread sudoku solvers where you can enjoy a game online. The Backtracking algorithm that is being used to widely solve computational constraints solving problems. Encoding the problem of sudoku in propositional logic and then feeding it to a solver is what excited us.

The GUI has been designed carefully, using an apt dialog box with choices dynamically in the form of a radio box and allowing easy undo until the very first choice. We also have coded for the solution to the problem in case the user is unable to solve for it and would like to learn his mistakes. A new game option has been provided so that the user can always start afresh.