COURSE PROJECT DOCUMENTATION

CS 101 PROJECTS 2015

SUDOKU QUEST

(INITIALLY THE PROJECT WAS ENROLLED AS SUDOKU AUTO-SOLVER)

TEAM ID-221

MITHUN OBULAMPALLI (140040107)

REVANTH RAJ MAMIDI (140040094)

B YASHWANTH KUMAR (140010044)

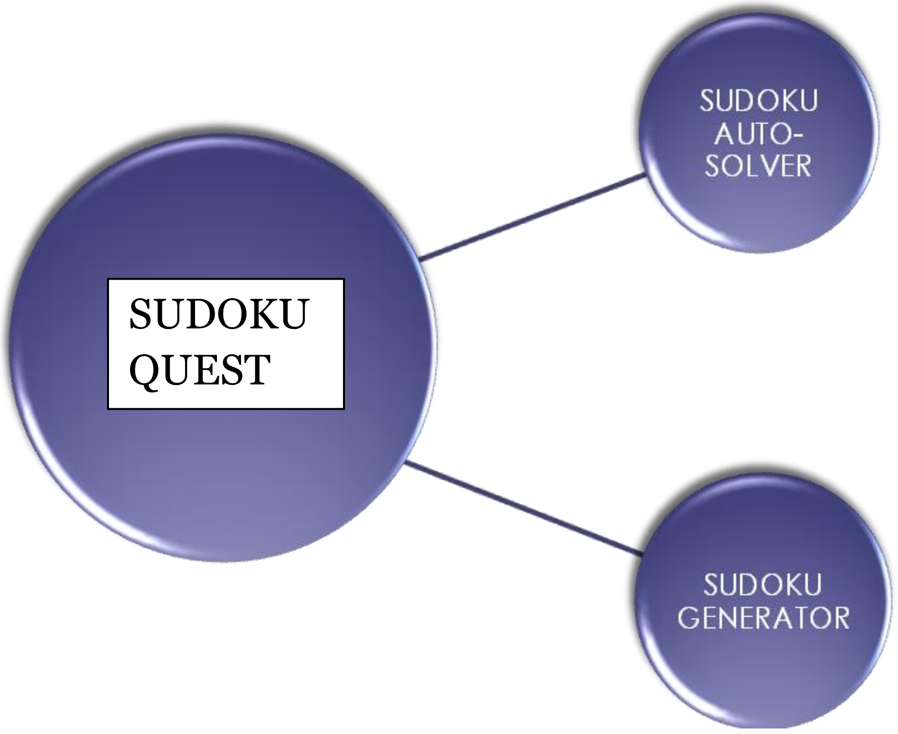
J HARVEER SINGH (140100057)

TABLE OF CONTENTS

INTRODUCTION

- Sudoku, originally called number placing game, is a logic based, combinatorial number placement puzzle.
- The objective is to fill a 9x9 grid with digits so that each row ,each column and each of the 3x3 sub-grids that composes the 9x9 grid contains all the digits from 1-9.
- The puzzle setter provides a partially completed grid, which for a well-posed puzzle has a unique solution.

Here, in this project, we came up with an app to generate random puzzles along with the intelligence to solve the puzzles on its own.[i]



PROBLEM STATEMENT

- The aim of the project is to make an application which can solve the user given Sudoku and also can generate a valid Sudoku.

- The difficulty level of the Sudoku can be selected by the user from the three modes namely "easy", "medium" or "hard". There is not much difference in the three modes except the number of previously defined numbers in the grid are reduced decreasing the chances of finding the position of a certain number making the puzzle challenging to solve.

- The application should be made as user-friendly as possible.

- It also has to give the solutions for the generated Sudoku puzzles.

REQUIREMENTS

A) HARDWARE REQUIREMENTS:

1. Keyboard

2. Mouse

B) SOFTWARE REQUIREMENTS:

Ubuntu or Windows operating system with simple code blocks installed with and also simple cpp library

IMPLEMENTATION

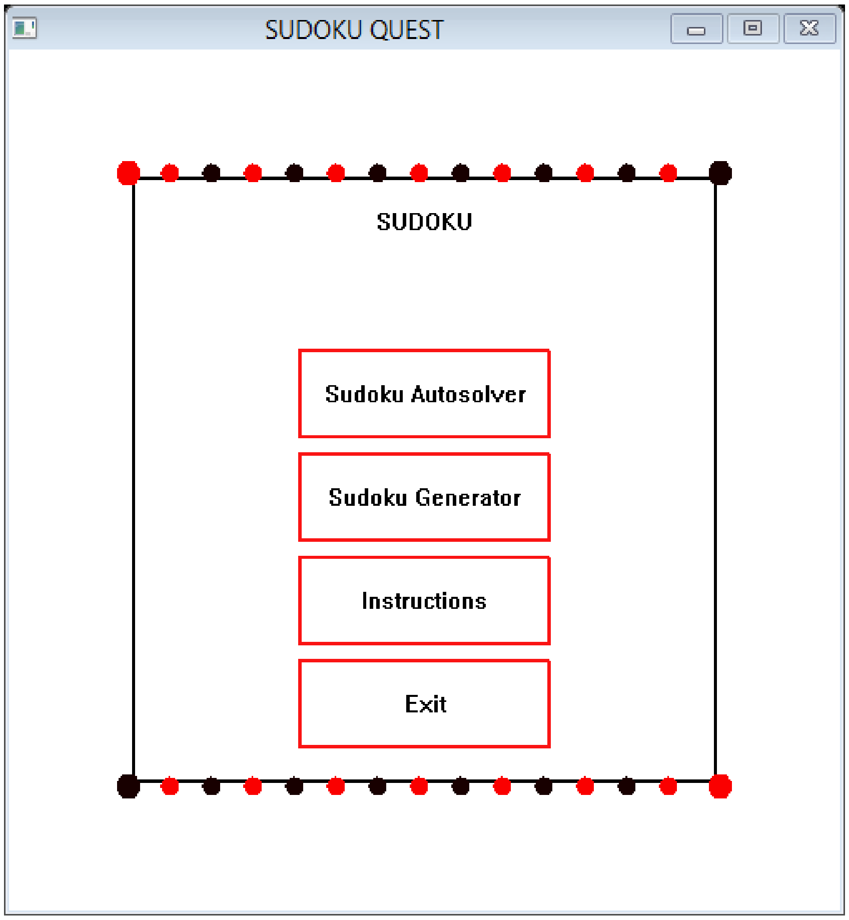
A) FUNCTIONALITY

1. **Sudoku Auto-solver**: The app takes in an input from the user, an incomplete puzzle and solves it using the back tracking concept. It first determines the positions of whatever numbers that can be directly found out from the given puzzle. Then it incorporates these new numbers into the puzzle in their appropriate positions. This is yet again an incomplete Sudoku but with more numbers than the previous one. Then it takes this new puzzle as input and solves it again, generating few more new numbers. It continues this process till the whole puzzle is filled and hence solves it.

2. **Sudoku Generator**: If the user wants to solve a puzzle then he has to give the difficulty which he wants to play .Then the app loads one of the preloaded Sudoku randomly. The user can then input the numbers and then submit it to cross check with the solution. The solution is not preloaded but it uses the auto-solver feature to get the solution.

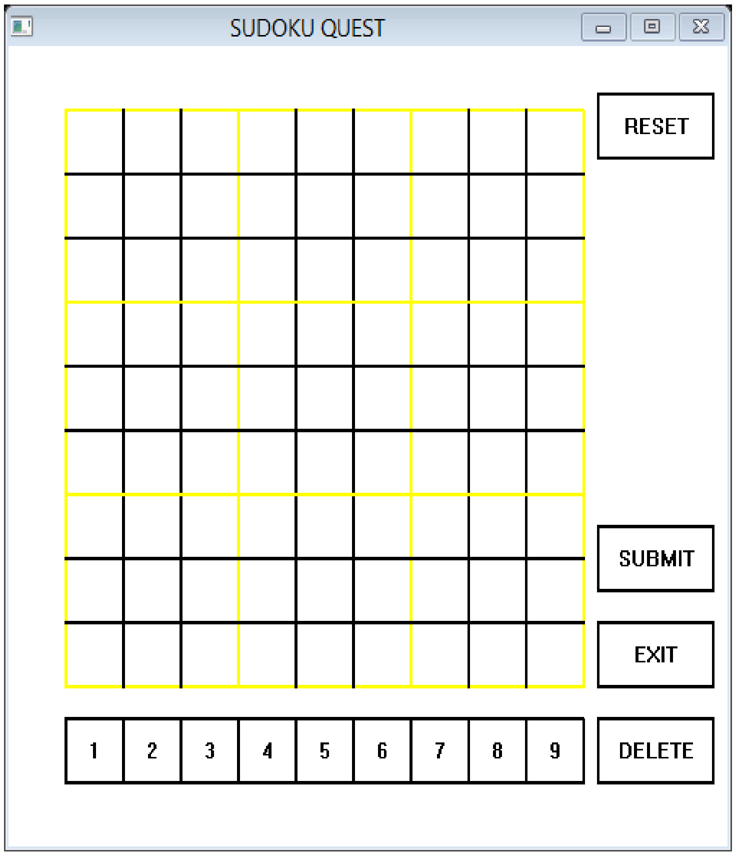
WORKING AND DATA

- Let's see how the app works now. We will see the working of all the functionalities as well.

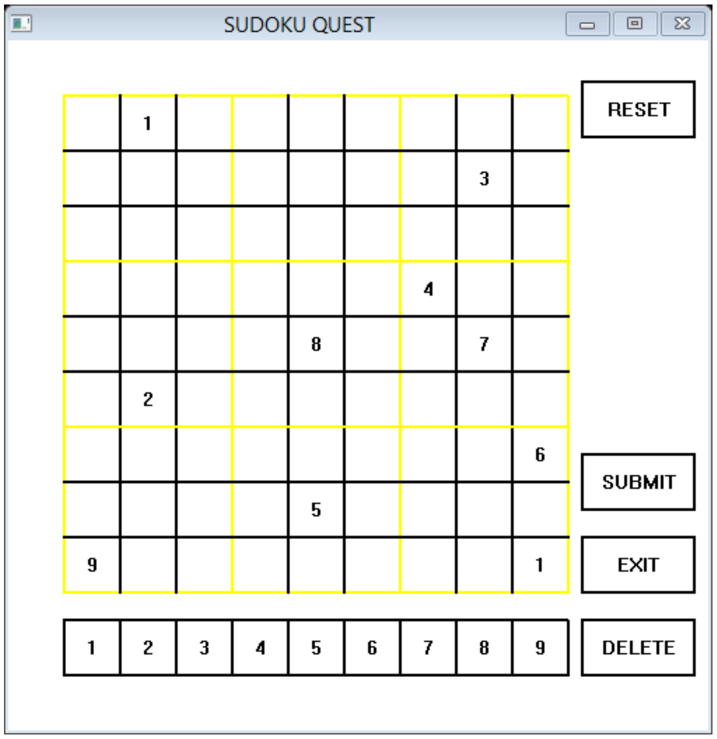


Home screen of the app 1

- When the auto-solver function is called this dialog appears.

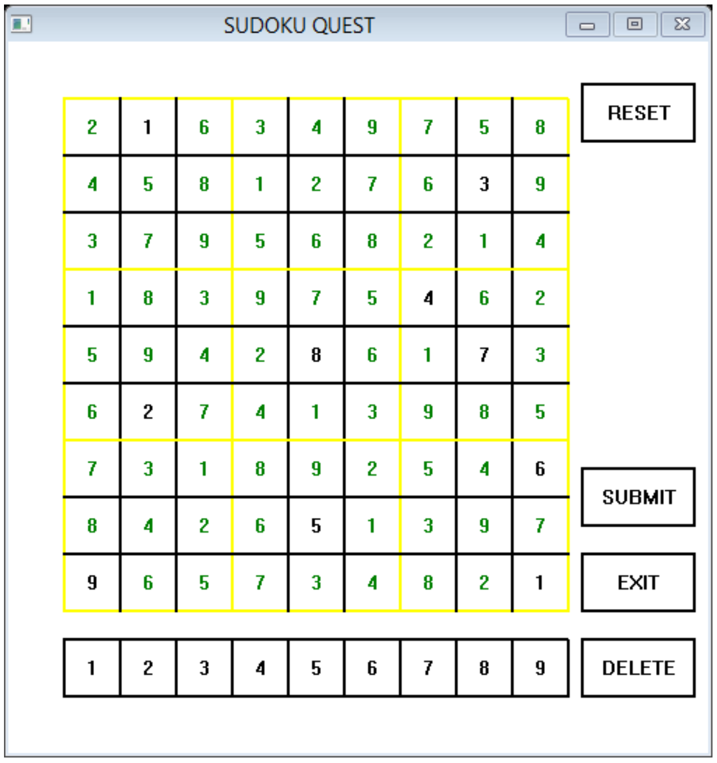


- We shall try giving it some input and make the app solve it



	1							
							3	
						4		
				8			7	
	2							
								6
				5				
9								1

1	2	3	4	5	6	7	8	9

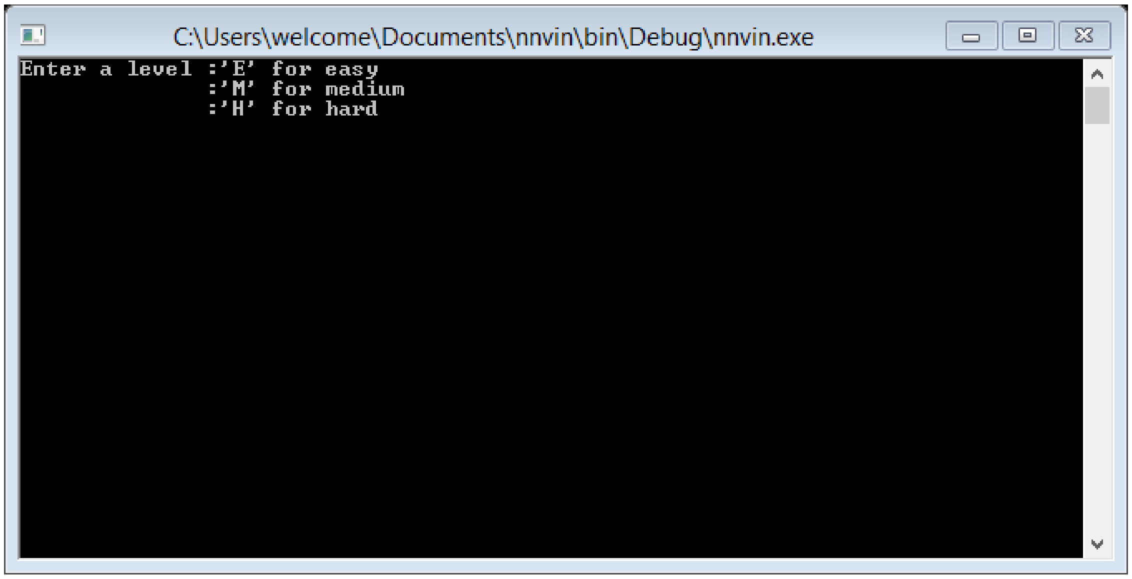


2	1	6	3	4	9	7	5	8
4	5	8	1	2	7	6	3	9
3	7	9	5	6	8	2	1	4
1	8	3	9	7	5	4	6	2
5	9	4	2	8	6	1	7	3
6	2	7	4	1	3	9	8	5
7	3	1	8	9	2	5	4	6
8	4	2	6	5	1	3	9	7
9	6	5	7	3	4	8	2	1

1	2	3	4	5	6	7	8	9

- We get the solution as shown above if we click submit button.

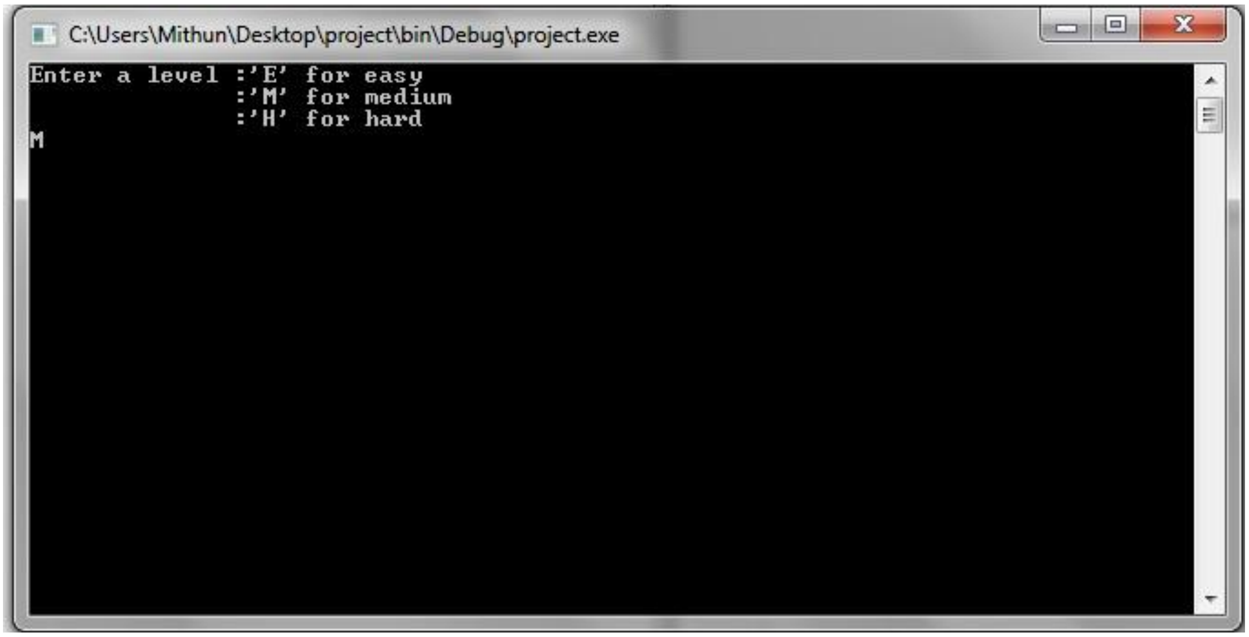
- If we call for Sudoku Generator function then we get this dialogue box.



C:\Users\welcome\Documents\nnvin\bin\Debug\nnvin.exe

```
Enter a level :'E' for easy
              :'M' for medium
              :'H' for hard
```

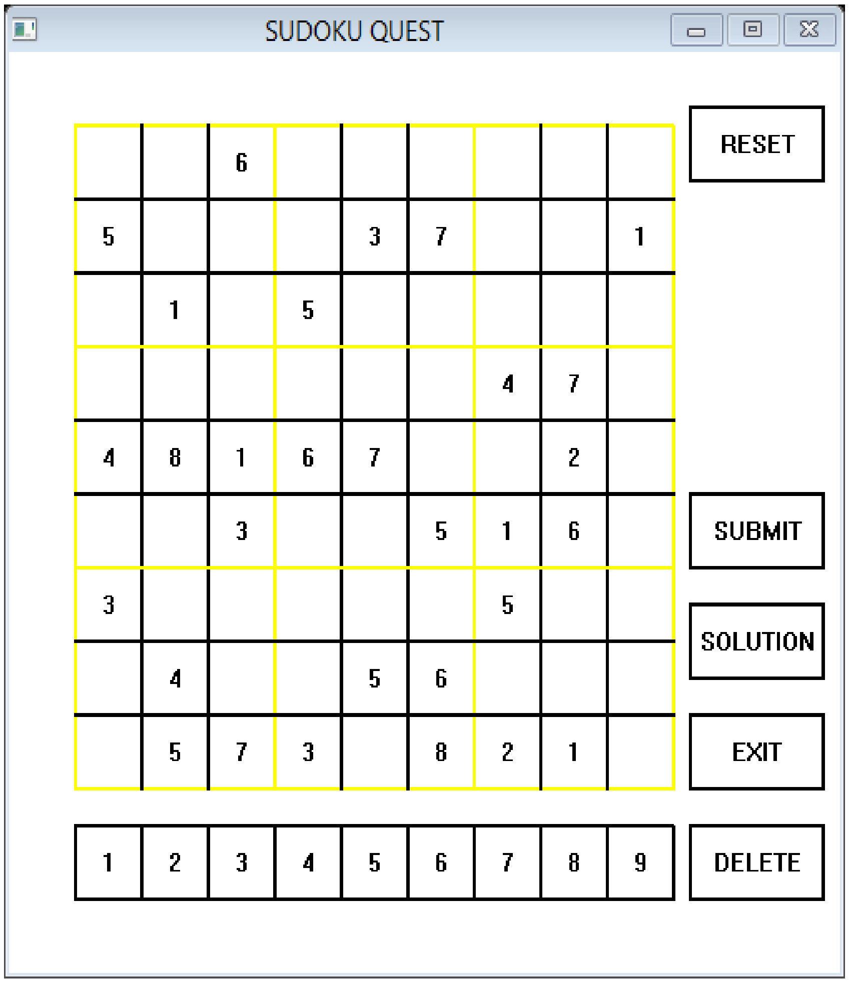
- Let's go for a medium level puzzle. Press "M" and this dialogue appears



C:\Users\Mithun\Desktop\project\bin\Debug\project.exe

```
Enter a level :'E' for easy
              :'M' for medium
              :'H' for hard
M
```

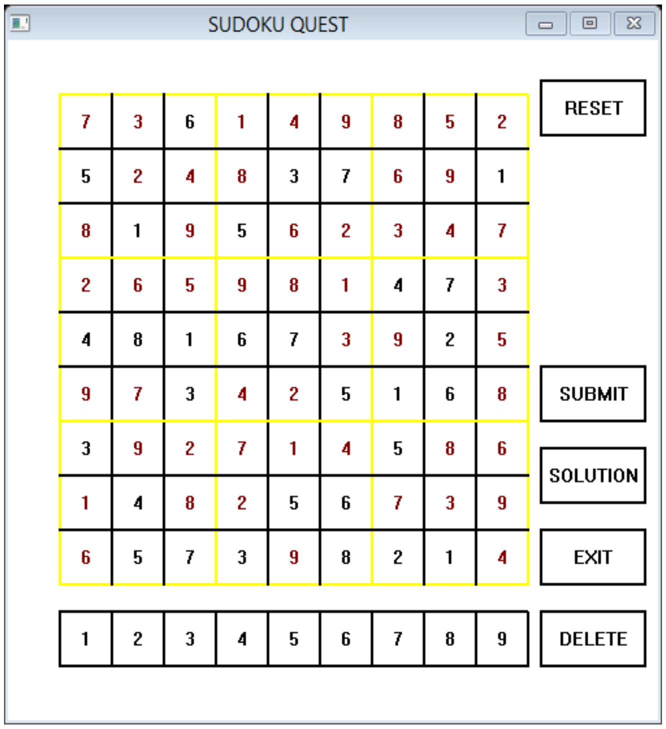
- It selects a random sudoku from the text files and generates a Sudoku as shown.



		6						
5				3	7			1
	1		5					
						4	7	
4	8	1	6	7			2	
		3			5	1	6	
3						5		
	4			5	6			
	5	7	3		8	2	1	

1	2	3	4	5	6	7	8	9

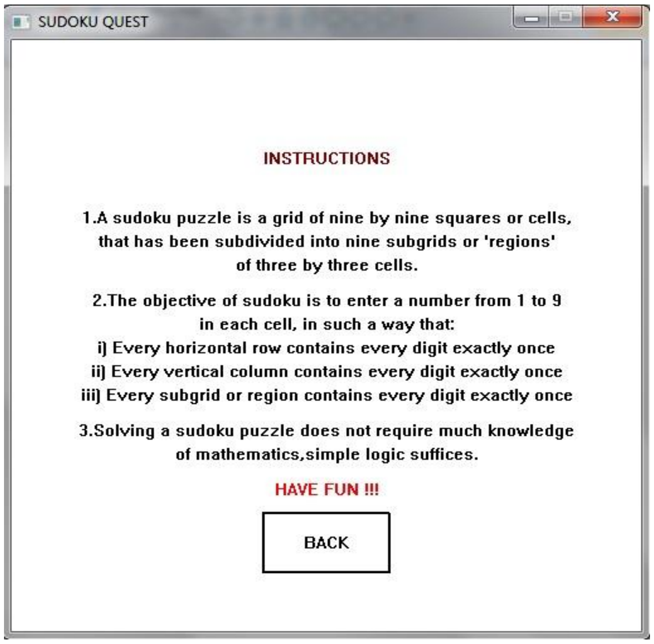
- We can solve it or else we can prompt for solution which it gives as shown.



7	3	6	1	4	9	8	5	2
5	2	4	8	3	7	6	9	1
8	1	9	5	6	2	3	4	7
2	6	5	9	8	1	4	7	3
4	8	1	6	7	3	9	2	5
9	7	3	4	2	5	1	6	8
3	9	2	7	1	4	5	8	6
1	4	8	2	5	6	7	3	9
6	5	7	3	9	8	2	1	4

1	2	3	4	5	6	7	8	9

- If the user presses the exit option then the app completely shuts down.
- There is a provision for the user to see the instructions also.



INSTRUCTIONS

1.A sudoku puzzle is a grid of nine by nine squares or cells,
that has been subdivided into nine subgrids or 'regions'
of three by three cells.

2.The objective of sudoku is to enter a number from 1 to 9
in each cell, in such a way that:
i) Every horizontal row contains every digit exactly once
ii) Every vertical column contains every digit exactly once
iii) Every subgrid or region contains every digit exactly once

3.Solving a sudoku puzzle does not require much knowledge
of mathematics,simple logic suffices.

HAVE FUN !!!

- Upon clicking back option it returns to the main interface.

DISCUSSION OF SYSTEMS

1. Things done as per plan:

- Sudoku Auto-solver
- Creating user-friendly graphical interface

2. What we added more than discussed in SRS:

- We felt that preloading the solutions also for the preloaded puzzles would make the application very crude.
- So, we made our auto-solver to then and there generate the solutions for the generated Sudoku.
- How much ever simple a game may ever be , its mandatory to give instructions to the user and hence we added instructions too.

3. Changes made in plan:

- At first the whole application ran in command prompt itself.
- But we thought it was too crude and hence we changed the interface and made the whole game to run through a canvas.
- The conversion and also the obtaining of click and matching the click co-ordinates to different functions was not very difficult but time consuming.

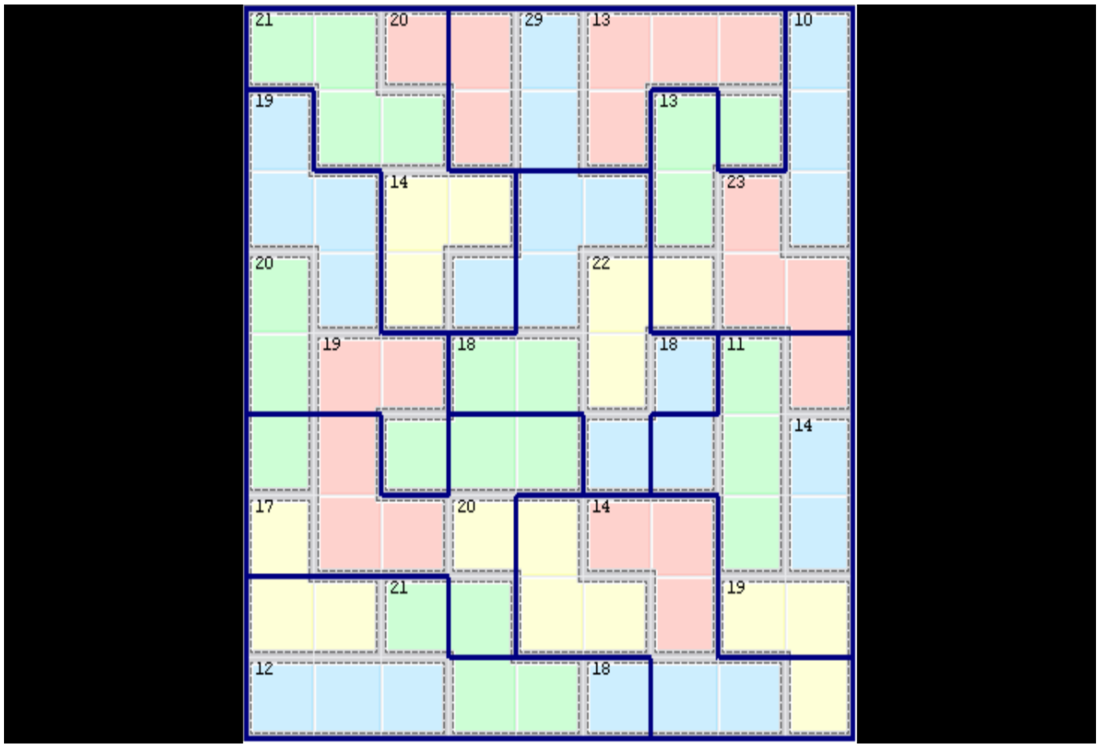
CHALLENGES

- Coming up with a backtracking algorithm for the auto solver function. At the beginning we wasted a lot of time but as we started defining functions then things started to fall into place.
- The other problem was displaying the preloaded Sudoku. At first we thought it wouldn't be much of a problem but later it proved to be a big problem.
- We first tried comma separated variables giving the x-coordinate, y-coordinate and the number to be displayed in each line. This method proved to consume lot of time and hence we looked up in previous projects and took them as a reference.

FUTURE WORK

A lot of improvements can be made to this document.

- Inputting more numbers of preloaded Sudoku puzzles in each level or even writing an algorithm for generating a random Sudoku puzzle.
- Changing the grid size and increasing the numbers.
- The shape of the 3x3 sub-grids can also be changed. They don't necessarily be square. For example



REFERENCES

- File operations : “An introduction to programming through C++” by Abhiram Ranade

- http://www.sanfoundry.com/cpp-program-solve-sudoku-problem -backtracking/

LINK TO OUR SPOKEN TUTORIAL:

https://www.youtube.com/watch?v=cm9vTAjV1oY