DOCUMENTATION

GROUP MEMBERS

NAME	ROLL NO
SHRUTI BELGALI	4
LAVINA GOLANI	16
GAYATRI NIPANKAR	39
VARSHA KUKREJA	29

SUDOKU SOLVER

For many who are new to Sudoku puzzles, finding a solution can be a complete mystery. On one hand, with so many numbers, Sudoku seems very mathematical. On the other hand, without the appropriate solution techniques, it can amount to a lot of guessing and checking. so here we designed a program to help you out.

What is Sudoku?

A Sudoku puzzle is defined as a logic-based, number-placement puzzle. The objective is to fill a 9×9 grid with digits in such a way that each column, each row, and each of the nine 3×3 grids that make up the larger 9×9 grid contains all of the digits from 1 to 9. Each Sudoku puzzle begins with some cells filled in. The player uses these seed numbers as a launching point toward finding the unique solution.

It is important to stress the fact that no number from 1 to 9 can be repeated in any row or column (although, the can be repeated along the diagonals).

How to Solve Sudoku Puzzles

There are two main techniques one can use to solve a Sudoku puzzle; Crosshatching and Pencilling In. These two techniques are simple, straightforward, reliable, and sufficient in solving most standard Sudoku puzzles. We will be majorly using the logic of pencilling in and bit of crosshatching logic.

IMPORTANT ELEMENTS OF THE PROGRAM AND THIER ROLE:

1. ***STRUCTURE MEMORY:*** A structure with tag name memory is declared globally. It has a data member which is an array of 9 integers. An array of 81 elements of datatype struct memory with variable m is declared globally. The prime aim is to associate an array with each of the 81 data elements of 9X9 Sudoku which at any time in the program will give the number of possibilities that may satisfy the empty box.

2. ***DISPLAY FUNCTION:*** The display function whenever called will display the Sudoku 9 X 9 grid.

3. ***CHECK FUNCTION****:"THIS FUNCTION IS THE BACKBONE OF THE ENTIRE PROGRAM".* The function is passed with values of row and column index of element whose solution is to be found. *It associates an array of 9 integers (1 to 9) with the index of the current grid element.*
    - *PENCLINING THE ROW: It deletes all those numbers from the array which are present in the same row as that of the element whose row no and column no is passed determining the possibilities of numbers for that unknown element* ***after pencilling the row*** *.If after pencilling the row, if size of array associated with the element reduces to 1 indicating there is only one possibility for that element to be filled then that element is filled and stored and the 9 X 9 GRID is updated and then the call returns back to main function and the check function is called for the next element whose value is known.*
    - *PENCLINING THE COLUMN: It deletes all those numbers from the array which are present in the same column as that of the element whose row no and column no is passed determining the possibilities of numbers for that unknown element* ***after pencilling the row& column both****. If after pencilling the row & column, if size of array associated with the element reduces to 1 indicating there is only one possibility for that element to be filled then that element is filled and stored and the 9 X 9 GRID is updated and then the call returns back to main function and the check function is called for the next element whose value is known.*
    - *PENCLINING THE GRID: It deletes all those numbers from the array which are present in the same 3 x 3 grid as that of the element whose row no and column no is passed determining the possibilities of numbers for that unknown element* ***after***

pencilling the row, column and grid *.If after pencilling the row , column and ,if size of array associated with the element reduces to 1 indicating there is only one possibility for that element to be filled then that element is filled and stored and the 9 X 9 GRID is updated and then the call returns back to main function and the check function is called for the next element whose value is known.*

- *After every element gets filled and causes the 9X9 grid to get updated the counter(keeps count of no of unknown elements in 9 X9 grid) decreases .Till the counter doesn't become zero ,the above 4 steps are traversed for each of the unknown element from the 81 elements of 9 X 9 grid.*

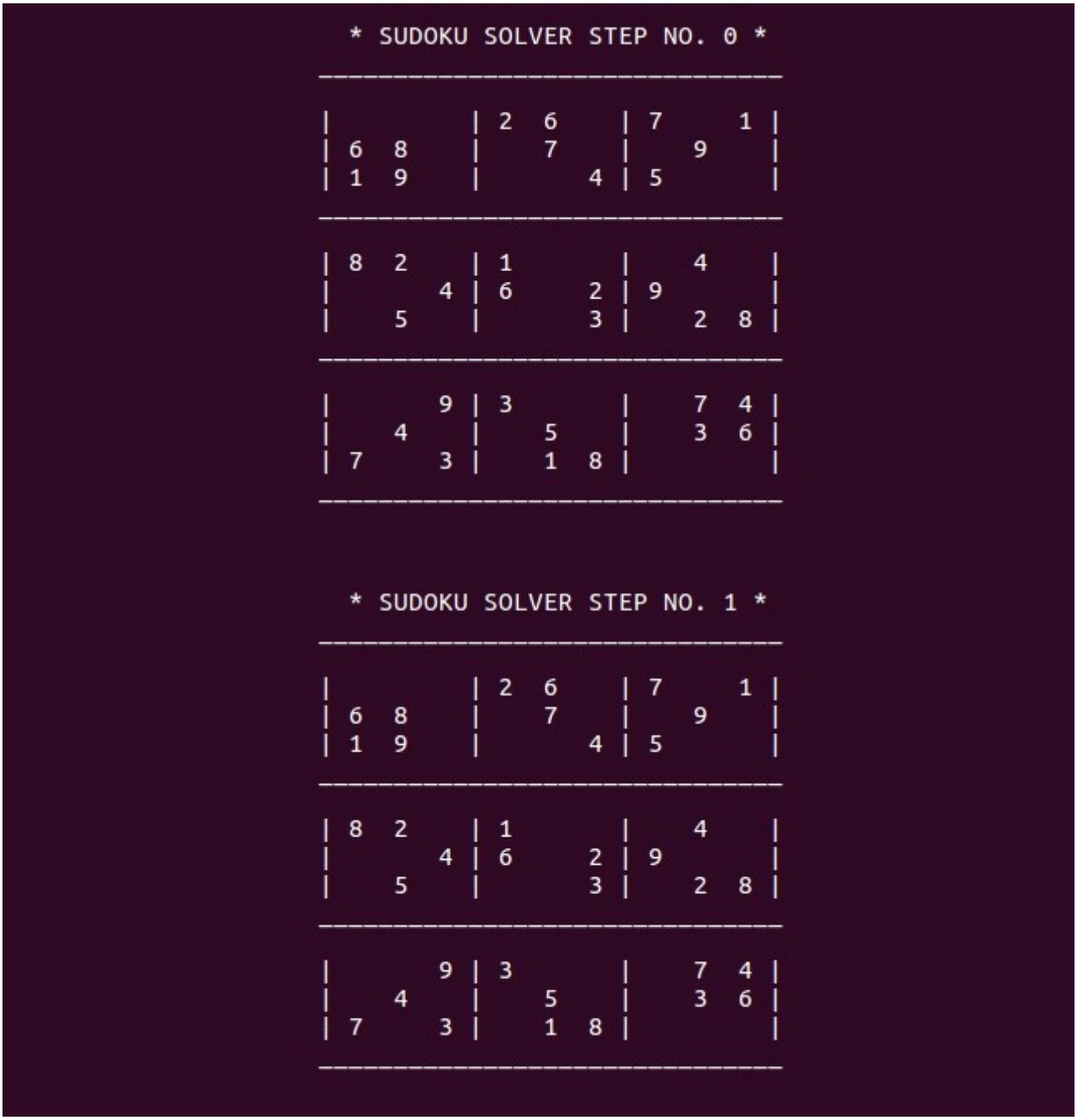
4. ***MAIN FUNCTION:*** It takes the input from the user and asks for at least 17 elements of Sudoku 9 X9 grid. It prompts the user with an error if less than 17 elements is given or a number greater than 9 or less than 0 is given as input by user. It takes the input in ***“ r c e”*** format i.e. ***“row no. column no. element”.*** It displays the origin 9X9 grid of Sudoku. With every input taken counter is decremented and user has to enter 100 when he/she finishes giving all the input to the program. TILL THE COUNTER DOESNT BECOME ZERO ,EACH

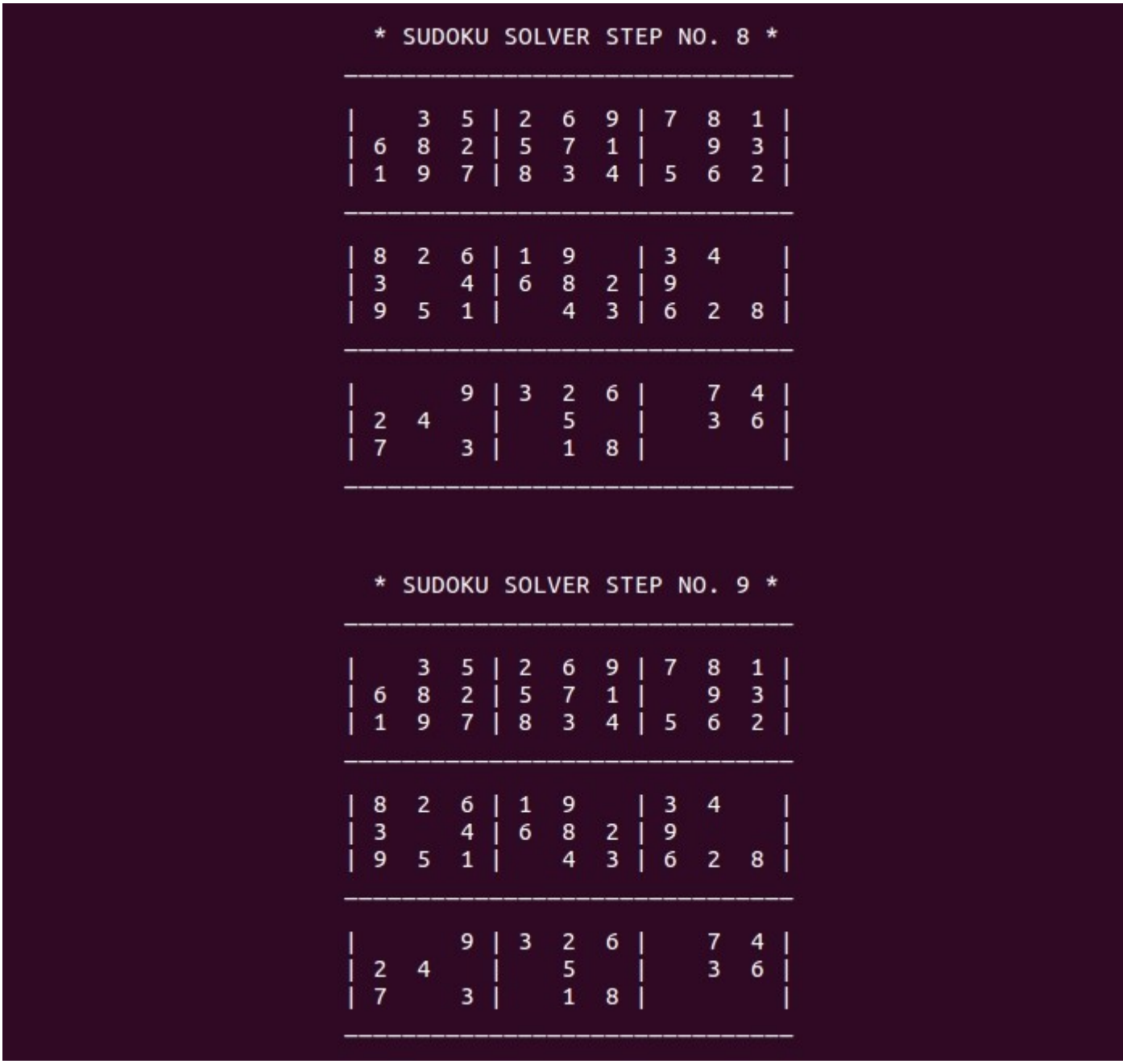
OF THE ELEMENT OF 9x9 GRID IS TRAVERSED ,AND FOR EACH UNKNOWN ELEMENT THE CHECK FUNCTION IS CALLED. When the counter becomes zero loop is terminated indicating ***Sudoku is solved.*** And for every 5 elements which get stored i.e. when grid is updated 5 times the display function is called to display the grid to give user an idea of how Sudoku was solved.

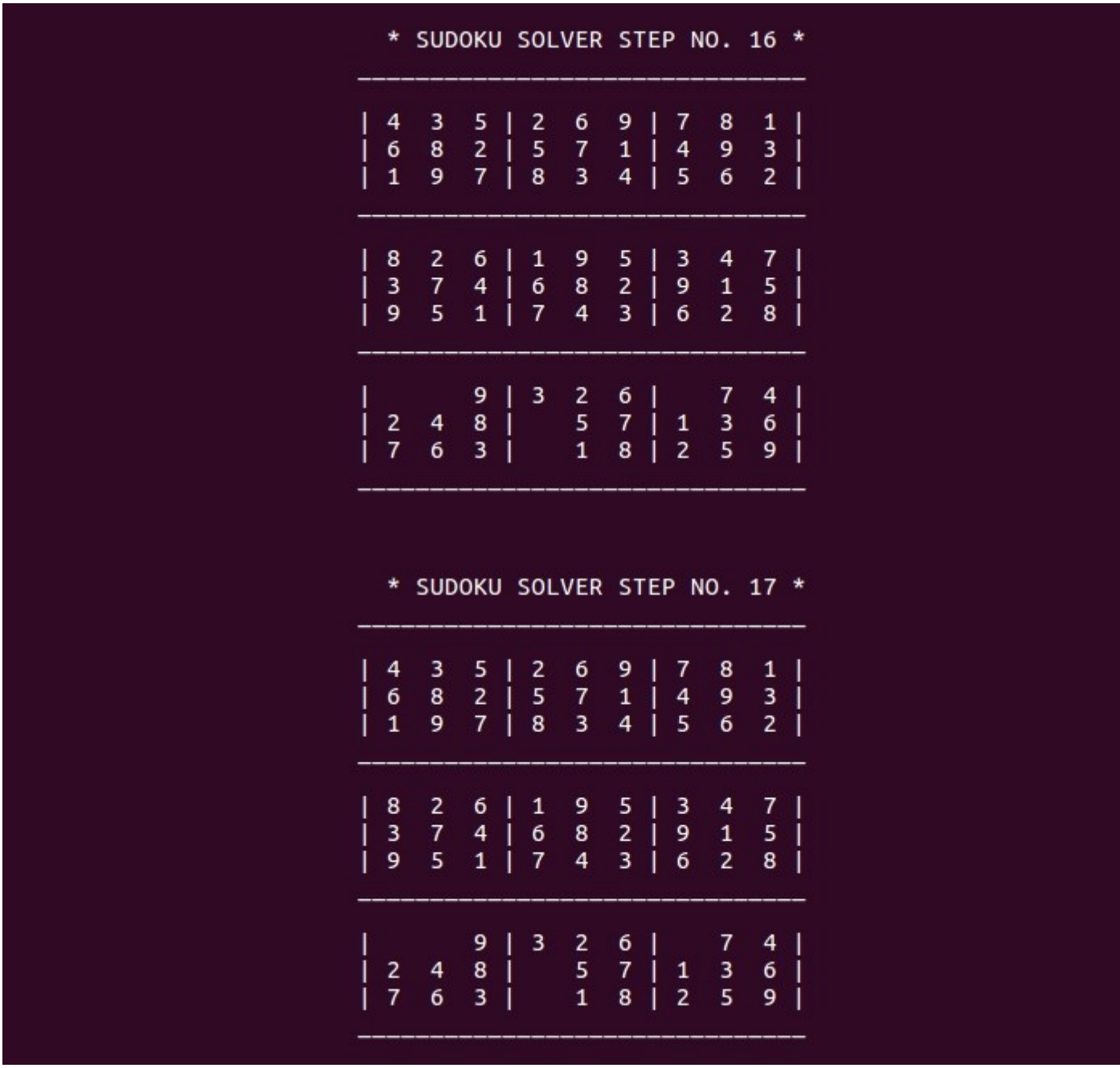
OUTPUT SCREENSHOTS

FOR VALID INPUT:the user enters 17 elements in the format as" row no. Col no. Element no"

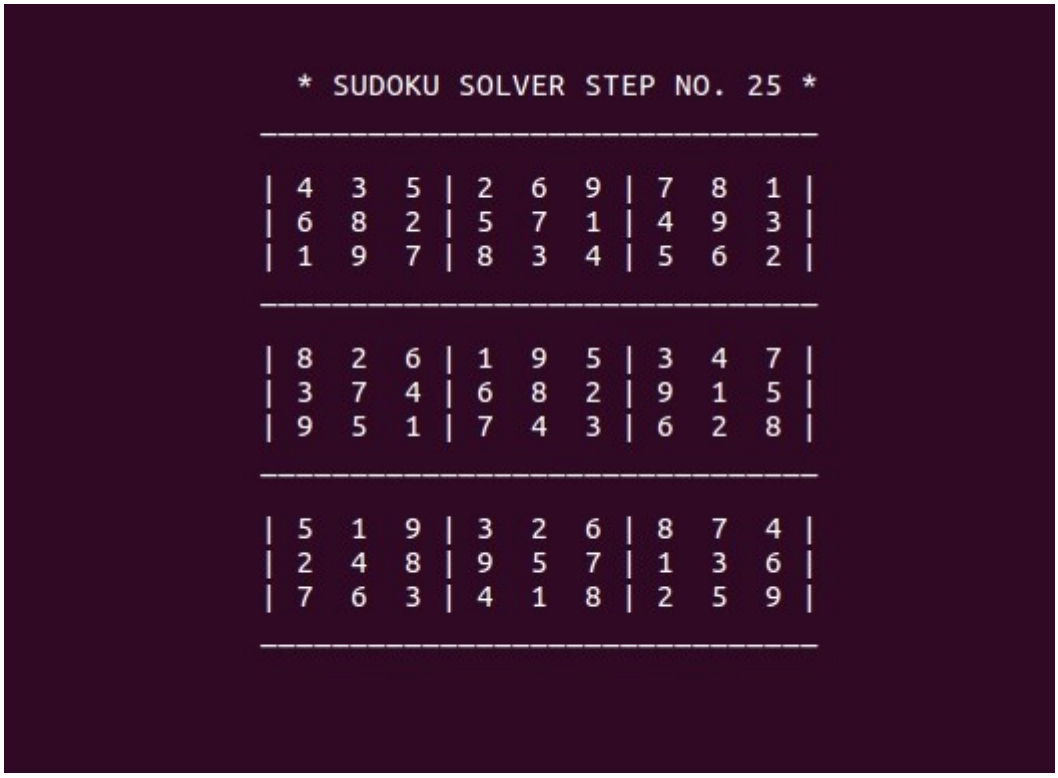
```
Enter the row, coloumn and element and 100 after you have completed entering the data:
Please ensure that you enter atleast 17 elements
1 4 2
1 5 6
1 7 7
1 9 1
2 1 6
2 2 8
2 5 7
2 8 9
3 1 1
3 2 9
3 6 4
3 7 5
4 1 8
4 2 2
4 4 1
4 8 4
5 3 4
5 4 6
5 6 2
5 7 9
6 2 5
6 6 3
6 8 2
6 9 8
7 3 9
7 4 3
7 8 7
7 9 4
8 2 4
8 5 5
8 8 3
8 9 6
9 1 7
9 3 3
9 5 1
9 6 8
100
```


* SUDOKU SOLVER STEP NO. 0 *
	2 6	7 1
6 8	7	9
1 9	4	5
8 2	1	4
4	6 2	9
5	3	2 8
9	3	7 4
4	5	3 6
7 3	1 8	
* SUDOKU SOLVER STEP NO. 1 *
	2 6	7 1
6 8	7	9
1 9	4	5
8 2	1	4
4	6 2	9
5	3	2 8
9	3	7 4
4	5	3 6
7 3	1 8	


* SUDOKU SOLVER STEP NO. 8 *
3 5	2 6 9	7 8 1
6 8 2	5 7 1	9 3
1 9 7	8 3 4	5 6 2
8 2 6	1 9	3 4
3 4	6 8 2	9
9 5 1	4 3	6 2 8
9	3 2 6	7 4
2 4	5	3 6
7 3	1 8
* SUDOKU SOLVER STEP NO. 9 *
3 5	2 6 9	7 8 1
6 8 2	5 7 1	9 3
1 9 7	8 3 4	5 6 2
8 2 6	1 9	3 4
3 4	6 8 2	9
9 5 1	4 3	6 2 8
9	3 2 6	7 4
2 4	5	3 6
7 3	1 8

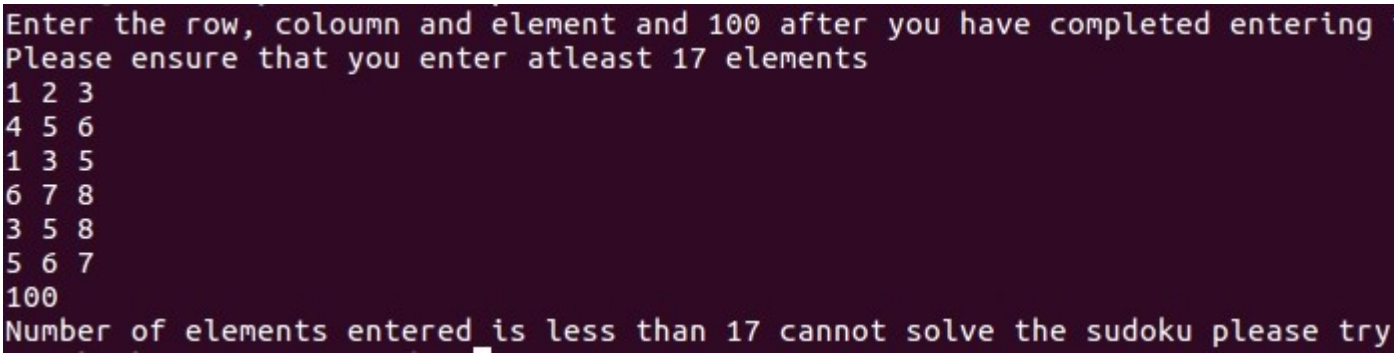


FINAL OUTPUT (SOLVED)

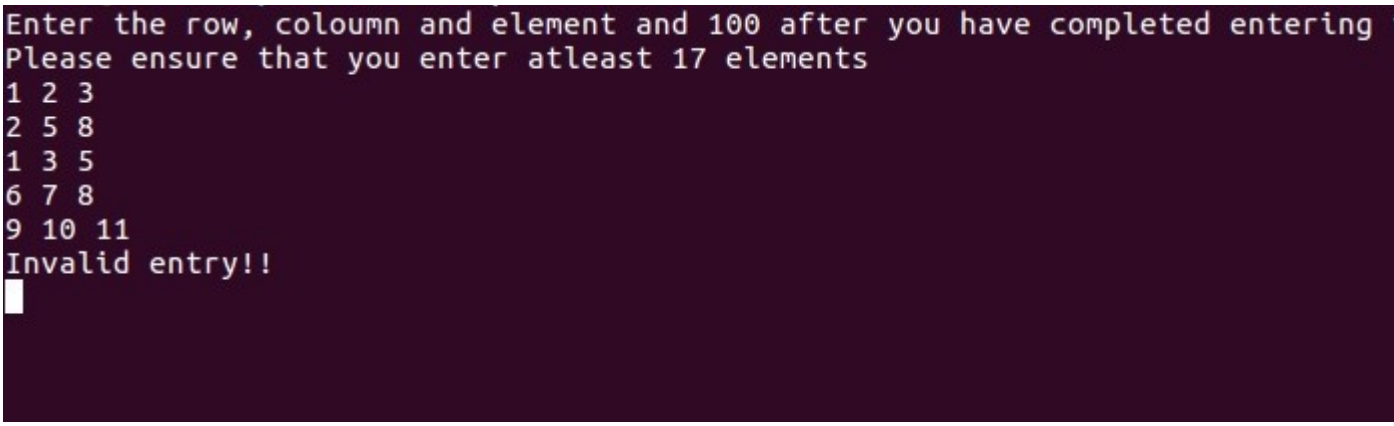


IF USER ENTERS ANY OF THESE INVALID INPUTS

1.when user enters less than 17 elements



2.when user enters either row no., col no. , or element more than 9



3.when user enters either row no., col no. , or element as a negative number

```
Enter the row, coloumn and element and 100 after you have completed entering
Please ensure that you enter atleast 17 elements
1 5 7
2 5 8
3 8 9
4 5 8
3 7 9
5 4 2
4 6 -2
Invalid entry!!
```