**Project Requirement and Specification**

**On**

**SUDOKU SOLVER**

**2 -Week Winter Programming Bootcamp**

**Submitted to: Submitted by:**

Ms. Meenakshi Mandola NANDINI GUPTA

UNIVERSITY ROLL.NO:-2014421

**Guided by:** SEMESTER:-III

Ms. Meenakshi Mandola Session: 2020-2021

**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**

**GRAPHIC ERA DEEMED TO BE UNVERSITY, DEHRADUN**

* **ABOUT PROJECT:**

The program statement is “ Input a partially filled 9\*9 2D array and if Sudoku if completely filled print the solved Sudoku and if not then print the message No solution exist ” in C++ .

 Sudoku is a puzzle where the objective is to fill a 9×9 square grid with digits numbered 1 to 9, so that each column, each row, and each of the nine 3×3 sub-grids contains all of the digits, but absolutely **NO** duplicates. The puzzle starts out partially filled in i.e some of the numbers are given to you as a starting point and other you have to fill.

* **TECHNOLOGIES USED IN THE PROJECT**:

To run the program in C++ you only need to download Codeblock from the internet.

* **Functions /Methods used in this project are:**

Approach:

1.find out a cell which is empty and keep a track of column number and row number.

2.Consider a number to be filled in the empty cell ,first check if it is safe ,if yes then assign the number to the empty cell otherwise try a different number. If a cell is discovered where none of the 9 digits is allowed then use backtracking method.

3.Backtracking: If we found a cell where no possible solution exists, then obviously we must have done something wrong with the previous cell. We should go back and change something in one of the previous cells and see if that works. Thus the algorithm will leave this unsolvable cell blank for now and move back i.e **backtrack**to the previous cell. The value in that cell is then incremented by one. This is repeated until the allowed value in the last (81st) cell is discovered. Backtracking is a depth-first search (in contrast to a breadth-first search), because it will completely explore one branch to a possible solution before moving to another branch.

4. Similarly try to fill other empty cell recursively.

5. If you are able to fill all the cell without any conflict then simply return Sudoku.

* **Headerfile-**

#include<iostream>

#include<cstring>

* **To solve this, we will follow these steps −**
* **Define a method called UsedInCol()**

This will take parameter as grid ,col and num , it will Returns whether any assigned entry in the specified column matches to the given number.

* **Define a method called UsedInRow(),**

This will take parameter grid ,row and numIt Returns whether any assigned entry in the specified row matches to the given number.

* **Define a method called UsedInBox()**

This will take parameter grid, boxStartRow, boxStartCol, num .It Returns whether any assigned entry within the specified 3x3 box matches the given number.

* **Define a method called FindUnassignedLocation**(),

This will take parameter grid, address of row and col.It Searches the grid to find an entry that is still unassigned.

* **Define a method called isSafe (),**

This will take parameter grid,row, col, num . It Returns whether it will be legal to assign num to the

* **Define a method called SolveSudoku(),**

This will take the parameter grid. It assign values to all unassigned locations for Sudoku solution.

* **Define a method called printgrid**()

,this will take the parameter grid and the solved Sudoku.