Project to solve suduko (“C” programming Language):

#include <stdio.h>

#include <stdlib.h>

#define N 9

void print(int arr[N][N])

{

for (int i = 0; i < N; i++)

{

for (int j = 0; j < N; j++)

printf("%d ",arr[i][j]);

printf("\n");

}

}

// Checks whether it will be legal

// to assign num to the

// given row, col

int isSafe(int grid[N][N], int row, int col, int num)

{

// Check if we find the same num in the similar row , we return 0

for (int x = 0; x <= 8; x++)

if (grid[row][x] == num)

return 0;

// Check if we find the same num in the

// similar column , we return 0

for (int x = 0; x <= 8; x++)

if (grid[x][col] == num)

return 0;

// Check if we find the same num in the particular 3\*3 matrix, we return 0

int startRow = row - row % 3;

int startCol = col - col % 3;

for (int i = 0; i < 3; i++)

for (int j = 0; j < 3; j++)

if (grid[i + startRow][j +

startCol] == num)

return 0;

return 1;

}

/\* Takes a partially filled-in grid and attempts

to assign values to all unassigned locations in

such a way to meet the requirements for

Sudoku solution (non-duplication across rows,

columns, and boxes) \*/

int solveSuduko(int grid[N][N], int row, int col)

{

// Check if we have reached the 8th row and 9th column (0 indexed matrix) , we are

// returning true to avoid further backtracking

if (row == N - 1 && col == N)

return 1;

// Check if column value becomes 9 ,we move to next row and

// column start from 0

if (col == N)

{

row++;

col = 0;

}

// Check if the current position

// of the grid already contains

// value >0, we iterate for next column

if (grid[row][col] > 0)

return solveSuduko(grid, row, col + 1);

for (int num = 1; num <= N; num++)

{

// Check if it is safe to place the num (1-9) in the

// given row ,col ->we move to next column

if (isSafe(grid, row, col, num)==1)

{

/\* assigning the num in the

current (row,col)

position of the grid

and assuming our assined num

in the position

is correct \*/

grid[row][col] = num;

// Checking for next possibility with next

// column

if (solveSuduko(grid, row, col + 1)==1)

return 1;

}

// Removing the assigned num ,

// since our assumption

// was wrong , and we go for next

// assumption with

// diff num value

grid[row][col] = 0;

}

return 0;

}

int main()

{

int grid[N][N] = { { 0, 0, 0, 2, 6, 0, 7, 0, 1 },

{ 6, 8, 0, 0, 7, 0, 0, 9, 0 },

{ 1, 9, 0, 0, 0, 4, 5, 0, 0 },

{ 8, 2, 0, 1, 0, 0, 0, 4, 0 },

{ 0, 0, 4, 6, 0, 2, 9, 0, 0 },

{ 0, 5, 0, 0, 0, 3, 0, 2, 8 },

{ 0, 0, 9, 3, 0, 0, 0, 7, 4 },

{ 0, 4, 0, 0, 5, 0, 0, 3, 6 },

{ 7, 0, 3, 0, 1, 8, 0, 0, 0 } };

if (solveSuduko(grid, 0, 0)==1)

print(grid);

else

printf("No solution exists");

return 0;

}