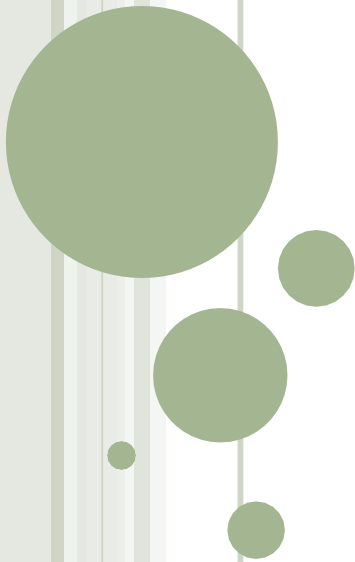


SUDOKU SOLVER

**Find the solution of your Sudoku
puzzle in a second's time!!**



INTRODUCTION

- Sudoku is one of the most popular and excellent brain puzzle games. The goal of Sudoku is to fill a 9×9 grid with numbers contain all of the digits between 1 and 9. Sudoku is a clear, easy and comes with good UI. It is the most beautiful, learnable, and user-friendly puzzle game. Play Sudoku daily, you will soon start to see improvements in your overall brain power



WHAT DOES MY PROJECT DO?

- A Sudoku solver solves any kind of a Sudoku given in the input, so you don't have to.
- The logic used in my program is designed such that it arranges all the numbers in a sequential manner according to the original result of the given Sudoku, in no time.



TECHNOLOGY USED-

- C(concept:- backtracking using recursion)
- IDE online gdb



CODE:

```
1  #include <stdio.h>
2  #include <stdlib.h>
3
4
5  #define N 9
6
7
8  void print(int arr[N][N])
9  {
10     for (int i = 0; i < N; i++)
11     {
12         for (int j = 0; j < N; j++)
13             printf("%d ", arr[i][j]);
14         printf("\n");
15     }
16 }
17
18
19 int isSafe(int grid[N][N], int row,
20           int col, int num)
21 {
22
23
24     for (int x = 0; x <= 8; x++)
25         if (grid[row][x] == num)
26             return 0;
27
```



```

28
29     for (int x = 0; x <= 8; x++)
30     |     if (grid[x][col] == num)
31     |         return 0;
32
33
34     int startRow = row - row % 3,
35     |     |     startCol = col - col % 3;
36
37     for (int i = 0; i < 3; i++)
38     |     for (int j = 0; j < 3; j++)
39     |     |     if (grid[i + startRow][j +
40     |     |     |     startCol] == num)
41     |     |     |     return 0;
42
43     return 1;
44 }
45
46
47 int solveSudoku(int grid[N][N], int row, int col)
48 {
49
50
51     if (row == N - 1 && col == N)
52     |     return 1;
53
54

```



```
55     if (col == N)
56     {
57         row++;
58         col = 0;
59     }
60
61
62     if (grid[row][col] > 0)
63         return solveSudoku(grid, row, col + 1);
64
65     for (int num = 1; num <= N; num++)
66     {
67
68
69         if (isSafe(grid, row, col, num)==1)
70         {
71
72             grid[row][col] = num;
73
74
75             if (solveSudoku(grid, row, col + 1)==1)
76                 return 1;
77         }
78
79         grid[row][col] = 0;
80     }
81     return 0;
82
```



```

main.c
107 // assumption with
108 // diff num value
109 grid[row][col] = 0;
110 }
111 return 0;
112 }
113
114 int main()
115 {
116 // 0 means unassigned cells
117 int grid[N][N] = { { 3, 0, 6, 5, 0, 8, 4, 0, 0 },
118 { 5, 2, 0, 0, 0, 0, 0, 0, 0 },
119 { 0, 8, 7, 0, 0, 0, 0, 3, 1 },
120 { 0, 0, 3, 0, 1, 0, 0, 8, 0 },
121 { 9, 0, 0, 8, 6, 3, 0, 0, 5 },
122 { 0, 5, 0, 0, 9, 0, 6, 0, 0 },
123 { 1, 3, 0, 0, 0, 0, 2, 5, 0 },
124 { 0, 0, 0, 0, 0, 0, 0, 7, 4 },
125 { 0, 0, 5, 2, 0, 6, 3, 0, 0 } };
126
127 if (solveSudoku(grid, 0, 0)==1)
128     print(grid);
129 else
130     printf("No solution exists");
131
132     return 0;
133
134 }
135

```

```

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

```

input



THANKYOU!

