Programiz C Online Compiler

Programiz PRO >

main.c

Share Run

```c
#include <stdio.h>
#include <stdbool.h>
#define N 9
void printGrid(int grid[N][N]) {
    for (int row = 0; row < N; row++) {
        for (int col = 0; col < N; col++) {
            printf("%2d", grid[row][col]);
        }
        printf("\n");
    }
}
bool isSafe(int grid[N][N], int row, int col, int num) {
    for (int x = 0; x < N; x++)
        if (grid[row][x] == num)
            return false;
    for (int x = 0; x < N; x++)
        if (grid[x][col] == num)
            return false;
    int startRow = row - row % 3, startCol = col - col % 3;
    for (int i = 0; i < 3; i++)
```

Output

Clear

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


=== Code Execution Successful ===
```

Programiz C Online Compiler

Programiz PRO ›

main.c

Share Run

```
    for (int i = 0; i < 3; i++)
        for (int j = 0; j < 3; j++)
            if (grid[i + startRow][j + startCol] == num)
                return false;

    return true;
}
bool solveSudoku(int grid[N][N]) {
    int row = -1, col = -1;
    bool emptyCell = false;
    for (int i = 0; i < N && !emptyCell; i++) {
        for (int j = 0; j < N && !emptyCell; j++) {
            if (grid[i][j] == 0) {
                row = i;
                col = j;
                emptyCell = true;
            }
        }
    }
    if (!emptyCell)
```

Output

Clear

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


=== Code Execution Successful ===
```

Programiz C Online Compiler

Programiz PRO >

main.c

Share Run

```
39      if (!emptyCell)
40          return true;
41      for (int num = 1; num <= 9; num++) {
42          if (isSafe(grid, row, col, num)) {
43              grid[row][col] = num;
44
45              if (solveSudoku(grid))
46                  return true;
47              grid[row][col] = 0;
48          }
49      }
50      return false;
51  }
52  int main() {
53      int grid[N][N] = {
54          {5, 3, 0, 0, 7, 0, 0, 0, 0},
55          {6, 0, 0, 1, 9, 5, 0, 0, 0},
56          {0, 9, 8, 0, 0, 0, 0, 6, 0},
57          {8, 0, 0, 0, 6, 0, 0, 0, 3},
```

Output

Clear

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


=== Code Execution Successful ===
```

Programiz C Online Compiler

Programiz PRO >

main.c

Share

Run

Output

Clear

```c
int main() {
    int grid[N][N] = {
        {5, 3, 0, 0, 7, 0, 0, 0, 0},
        {6, 0, 0, 1, 9, 5, 0, 0, 0},
        {0, 9, 8, 0, 0, 0, 0, 6, 0},
        {8, 0, 0, 0, 6, 0, 0, 0, 3},
        {4, 0, 0, 8, 0, 3, 0, 0, 1},
        {7, 0, 0, 0, 2, 0, 0, 0, 6},
        {0, 6, 0, 0, 0, 0, 2, 8, 0},
        {0, 0, 0, 4, 1, 9, 0, 0, 5},
        {0, 0, 0, 0, 8, 0, 0, 7, 9}
    };

    if (solveSudoku(grid))
        printGrid(grid);
    else
        printf("No solution exists\n");

    return 0;
}
```

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


=== Code Execution Successful ===
```