SUDOKU SOLVER

Sudoku solver takes inputs as a 2-D array(usually 9x9) with 0 representing empty values and number from 1 to n(9) in some positions on the board. If the sudoku cannot be solved the program prints 'No solution' or it returns the solved sudoku board.

Functions

1. IsAvailable(int [][], int, int, int, int);

   The parameters are the sudoku board(2-D array), row number, column number and the number which is being tested as a possible value for the square. It checks the row,column and the 3x3 square to which the number belongs.

   Returns 1 if the number is a possible value for the square, 0 if not.

2. Sudoku_solve(int [][], int, int)

   The parameters of the function are the sudoku board, row and column index of the square which is to be filled. The function checks if the sqaure has a value(non-zero) and goes to the next column or the next of row if it has reached the last column. If the square is empty(zero) the fucntion calls IsAvailable() to check for the possible values of the square and then assigns the value to the square. This continues for each empty square. If it comes across a square where there is no possible value then the function backtracks to the last filled sqaure to check other permutations. The function returns 1 if the sudoku board is solved and 0 if not solved.

Sample input:

```
{3, 0, 6, 5, 0, 8, 4, 0, 0},
{5, 2, 0, 0, 0, 0, 0, 0, 0},
{0, 8, 7, 0, 0, 0, 0, 3, 1},
{0, 0, 3, 0, 1, 0, 0, 8, 0},
{9, 0, 0, 8, 6, 3, 0, 0, 5},
{0, 5, 0, 0, 9, 0, 6, 0, 0},
{1, 3, 0, 0, 0, 0, 2, 5, 0},
{0, 0, 0, 0, 0, 0, 0, 7, 4},
{0, 0, 5, 2, 0, 6, 3, 0, 0}
```

Sample output:

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