```
#define LENGTH                          (9)
#define SUB_BOX_LENGTH                  (LENGTH/3)
#define TOTAL_NUM                       (9)
#define S2I(x)                          ((x)-'0')

bool checkBox(char** board, int boardSize, int* boardColSize, int row, int column)
{
    int row_start;
    int column_start;
    int row_end;
    int column_end;
    bool map[TOTAL_NUM] = {false};


    row_start = ( row / SUB_BOX_LENGTH ) * SUB_BOX_LENGTH;
    column_start = ( column / SUB_BOX_LENGTH ) * SUB_BOX_LENGTH;

    row_end = row_start + SUB_BOX_LENGTH;
    column_end = column_start + SUB_BOX_LENGTH;

    for(;row_start<row_end;row_start++)
    {
        for(;column_start<column_end;column_start++)
        {
            if(map[S2I(board[row_start][column_start]) - 1] == true)
            {
                return false;
            }else
            {
                map[S2I(board[row_start][column_start]) - 1] = true;
            }

        }
    }

    return true;
}

bool checkRowOrColumn(char** board, int boardSize, int* boardColSize, int row_column, bool use_row)
{
    int column;
    int row;
    int* index;
    bool map[TOTAL_NUM] = {false};

    if(use_row == true)
    {
        index = &column;
        row = row_column;
    }else
    {
        index = &row;
        column = row_column;
    }

    *index = 0;

    while(*index < TOTAL_NUM)
    {
        if(board[row][column]  == '.' || map[S2I(board[row][column]) - 1] == true)
        {
            return false;
        }else
        {
            map[S2I(board[row][column]) - 1] = true;
        }

        (*index)++;
    }

    return true;
}

bool _solveSudoku(char** board, int boardSize, int* boardColSize, bool (*row_map)[TOTAL_NUM], bool (*column_map)[TOTAL_NUM], bool (*box_map)[SUB_BOX_LENGTH][TOTAL_NUM], int row, int c
    int row_index;
    int column_index;
    int next_row_index;
    int next_column_index;
    int num_index;
    int tmp_index;

    if(row == TOTAL_NUM)
    {
        return true;
    }

    row_index = row;
    column_index = column;


    if(board[row_index][column_index] == '.')
    {
        for(num_index = 0; num_index < TOTAL_NUM;num_index++)
        {
            if( row_map[row_index][num_index] == false &&
                column_map[column_index][num_index] == false &&
                box_map[row_index/SUB_BOX_LENGTH][column_index/SUB_BOX_LENGTH][num_index] == false
              )
            {
                row_map[row_index][num_index] = true;
                column_map[column_index][num_index] = true;
                box_map[row_index/SUB_BOX_LENGTH][column_index/SUB_BOX_LENGTH][num_index] = true;

                board[row_index][column_index] = num_index + '0' + 1;
                //printf("board[%d][%d]:%c\n", row_index, column_index, board[row_index][column_index]);
                do
                {
                    if(row == (TOTAL_NUM-1))
                    {
                        if(checkRowOrColumn(board, boardSize, boardColSize, column_index, false) == false)
                        {
                            break;
                        }
                    }

                    if(column == (TOTAL_NUM-1))
                    {
                        if(checkRowOrColumn(board, boardSize, boardColSize, row_index, true) == false)
                        {
                            break;
                        }
                    }

                    if( (row%SUB_BOX_LENGTH) == (SUB_BOX_LENGTH-1) && (column%SUB_BOX_LENGTH) == (SUB_BOX_LENGTH-1) )
                    {
                        if(checkBox(board, boardSize, boardColSize, row_index, column_index) == false)
                        {
                            break;
                        }
                    }

                    if(_solveSudoku(board, boardSize, boardColSize, row_map, column_map, box_map, (column+1)%TOTAL_NUM == 0 ? row+1:row, (column+1)%TOTAL_NUM))
                    {
                        return true;
                    }
                }while(0);

                row_map[row_index][num_index] = false;
                column_map[column_index][num_index] = false;
                box_map[row_index/SUB_BOX_LENGTH][column_index/SUB_BOX_LENGTH][num_index] = false;

            }

        }

        board[row_index][column_index] = '.';

    }else
    {
        if(_solveSudoku(board, boardSize, boardColSize, row_map, column_map, box_map, (column+1)%TOTAL_NUM == 0 ? row+1:row, (column+1)%TOTAL_NUM))
        {
            return true;
        }
    }

        //}
```

```
        //column_index = 0;
    //}

    return false;

}

void solveSudoku(char** board, int boardSize, int* boardColSize){
    bool row_map[LENGTH][TOTAL_NUM] = {false};
    bool column_map[LENGTH][TOTAL_NUM] = {false};
    bool box_map[SUB_BOX_LENGTH][SUB_BOX_LENGTH][TOTAL_NUM] = {false};
    int row_index;
    int column_index;
    char tmp;

    for(row_index = 0 ; row_index < boardSize; row_index++)
    {
        for(column_index = 0 ; column_index < *boardColSize; column_index++)
        {
            tmp = board[row_index][column_index];

            if(tmp != '.')
            {
                row_map[row_index][S2I(tmp)-1] = true;
                column_map[column_index][S2I(tmp)-1] = true;
                box_map[row_index/SUB_BOX_LENGTH][column_index/SUB_BOX_LENGTH][S2I(tmp)-1] = true;
            }

        }
    }

    _solveSudoku(board, boardSize, boardColSize, row_map, column_map, box_map, 0, 0);
}
```