```
3+4
```

```
 7
```

```
sudoku = [[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],[8,0,0,0,6,0,4,2,3],
```

```
def get_block_num(sudoku,pos):
    row_jump = pos[0]//3
    row_rem = pos[0]%3
    col_jump = pos[1]//3
    col_rem = pos[1]%3
    return (row_jump-1)*3 + (3)*int(row_rem!=0) + (col_jump) + int(col_rem!=0)
```

```
def get_pos_inside_block(sudoku,pos):
    num = get_block_num(sudoku,pos)
    if num%3==0:
        if pos[0]%3 ==0:
            return pos[1]
        else:
            (pos[0] - (pos[0]//3)*3 - 1)*3 + pos[1] - 6
    else:
        if pos[0]%3 ==0:
            return 6 + pos[1] - (num%3 - 1)*3
        else:
            return (pos[0] - (pos[0]//3)*3 - 1)*3 + pos[1] - (num%3 - 1)*3
```

```
def get_block(sudoku,x):
    list1=[]
    for i in range(1,10):
        for j in range(1,10):
            if (get_block_num(sudoku,(i,j))==x):
                 list1.append(sudoku[i-1][j-1])
    return list1
```

```
get_block(sudoku,2)
```

```
 [0, 7, 0, 1, 9, 5, 0, 0, 0]
```

```
def get_row(sudoku,x):
    return sudoku[x-1]
```

```
def get_col(sudoku,x):
    col = []
    for i in range(1,10):
        col.append(sudoku[i-1][x-1])
    return col
```

```
def find_first_unassigned_position(sudoku):
```

```
    for i in range(1,10):
        if (0 in sudoku[i-1]):
            return (i,sudoku[i-1].index(0)+1)
    return (-1,-1)


find_first_unassigned_position(sudoku)
```

```
     (1, 3)
```

```
def valid_list(list):
    for i in list:
        if (list.count(i)>1)and(i!=0):
            return False
    return True


def valid_sudoku(sudoku):
    count=0
    for i in range(1,10):
        if valid_list(get_row(sudoku,i))==True:
            count+=1
        if valid_list(get_block(sudoku,i))==True:
            count+=1
        if valid_list(get_col(sudoku,i))==True:
            count+=1
    if count==27:
        return True
    else:
        return False



valid_sudoku(sudoku)
```

```
     True
```

```
def get_candidates(sudoku,pos):
    list_cand=[]
    row = get_row(sudoku,pos[0])
    col = get_col(sudoku,pos[1])
    block = get_block(sudoku,get_block_num(sudoku,pos))
    for i in range(1,10):
        if (i not in row)and(i not in col)and(i not in block):
            list_cand.append(i)
    return list_cand


get_candidates(sudoku,(1,3))
```

```
     [1, 2, 4]
```

```
def make_move(sudoku,pos,num):
    sudoku[pos[0]-1][pos[1]-1] = num
```

```
    return sudoku


def undo_move(sudoku,pos):
    sudoku[pos[0]-1][pos[1]-1] = 0
    return sudoku


def sudoku_solver(sudoku):
    if find_first_unassigned_position(sudoku)==(-1,-1):
            return (True,sudoku)
    else:
        pos=find_first_unassigned_position(sudoku)
        can= get_candidates(sudoku, pos)
        for i in can:
            sudoku=make_move(sudoku, pos, i)
            if sudoku_solver(sudoku)[0] is True:
                return (True,sudoku)
            else:
                sudoku=undo_move(sudoku,pos)
                continue
    return (False, sudoku)


sudoku #before solving

     [[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, 4, 2, 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]]


sudoku_solver(sudoku)

     (True,
      [[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]])
```