Matrix:

Given a 9*9 Matrix.Print "Valid", if every 3*3 sub-matrix of it comprises all digits from 1 to 9.

Input Format:

First Line contains the 't', i.e; the no. of test cases. Next subsequent lines contain the nine by nine set.

Output Format:

Each line corresponding to the 9*9 set whether it is "VALID" or "INVALID".

Sample Input:

1

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

Sample Output:

VALID

Solution:

```
puzzle = [[] for j in range(10)]

def value_in_box(r, c, value):
    if r < 4:
        i = 1
    elif r < 7:
        i = 4
    else:
        i = 7
```

```
    if c < 4:
        j = 1
    elif c < 7:
        j = 4
    else:
        j = 7

    for a in range(i, i+3):
            for b in range(j, j+3):
                if a != r or b != c:
                    if (puzzle[a][b] == value):
                        return True
    return False

def is_safe(r, c, value):
    for i in range(1, 10):
        if i != c:
            if puzzle[r][i] == value:
                return False
    for i in range(1, 10):
        if i != r:
            if puzzle[i][c] == value:
                return False
    if (value_in_box(r, c, value)):
        return False

    return True

def sudoku_solver_util():
    for i in range(1, 10):
        for j in range(1, 10):
            if is_safe(i, j, puzzle[i][j]) == False:
                return False
    return True

def sudoku_solver():
    if (sudoku_solver_util()):
        print("VALID")
    else:
        print("INVALID")

def init():
    t = int(input())
```

```
    while (t):
        for i in range(1, 10):
            puzzle[i] = [int(x) for x in input().split()]
            puzzle[i].insert(0, 0)
        sudoku_solver()
        t -= 1
        for x in range(10):
            puzzle[x].clear()
init()
```