Ankit Kesar (1BM18CS150)

AI LAB - Program -(1)

```
def index(mylist, v):
    for i, v in enumerate(mylist):
        if v in x:
            return (i, x.index(v))

def manhattan(temp):
    sum = 0
    for i in range(3):
        for j in range(3):
            if temp[i][j] != 0:
                b = index(temp, temp[i][j])
                c = index(goal, temp[i][j])
                sum += abs(b[0] - c[0])
                sum += (abs(b[1] - c[1]))
    return sum

def possible_moves(temp, visited):
    possible_mvs = []
    b = index(temp, 0)
    direction = []
    if b[0] < 2:
        direction.append("d")
    if b[0] > 0:
        direction.append("u")
```

```
        if b[1] < 2:
            direction.append('r')
        if b[1] > 0:
            direction.append('l')
        for i in direction:
            move = gen(temp, i, b)
            if move not in visited:
                possible_mvs.append(move)
    return possible_mvs.

def solve(visited, limit, src)
    if src == goal:
        print("Required moves to reach the goal state" + str(limit))
        return True
    if limit > 3
        return False
    min = math.inf
    visited.append(src)
    possible_action = possible_moves(src, visited)
    new_moves = []
    for action in possible_action:
        man_dist = manhattan(action)
        if action not in visited and man_dist < min:
            min = min_dist
            new_move = action
```

```
print ("move : " limit+1)
print_matrix (new_move)

if solve (visited, limit+1, new_move):
        return True
else:
    return False.
```