Nidhish Lakhinana
1BM18CS059

```
~~def~~ def h( State, target)
    dist = 0
    for i in state :
        d1 , d2 = state.index(i), target.index(i)
        x1 , y1 = d1 % 3 , d1 // 3
        x2 , y2 = d2 % 3 , d2 // 3
        dist += abs (x1 - x2) + abs ( y1 - y2)
    return dist

def astar (src, target):
    states = [src]
    g = 0
    visited_states = set()
    while len(states):
        print (f"Level : {g}")
        moves = []
        for state in states :
            visited_states.add(tuple(state))
            print_grid(state)
            if ~~g>2~~ g < 2 :
                if state == target:
                    print ("success")
                    return
                moves += [move for move in possible_moves(state
                , visited_states) if move not in moves]
                else
                    print (" NOSOLUTION")
                    ~~return~~ ~~breat~~ break
        costs = [g + h(move, target) for move in moves] # f(n) = g(n)+h(n)
        states = [moves[i] for i in range(len(moves)) if costs[i]
    g += 1              == min(costs)]
    print ("NO SOLUTION")

def possible_moves(state, visited_states):
    b = state.index(-1)
    d = []
    if 9 > b - 3 >= 0:
        d += 'u'
    if 9 > b + 3 >= 0:
        d += 'd'
    if b not in [2,5,8]:
        d += 'r'
    if b not in [0,3,6]:
        d += 'l'
    pos_moves = []
    for move in d:
        pos_moves.~~etc~~append(gen(state, move, b))
    return [move for move in pos_moves if tuple(move) not in
            visited_states]
    ~~return~~
```