def print\_b(src):

state = src.copy()

state[state.index(-1)] = ' '

print(

f"""

{state[0]} {state[1]} {state[2]}

{state[3]} {state[4]} {state[5]}

{state[6]} {state[7]} {state[8]}

"""

)

def h(state, target):

count=0

i=0

for j in state:

if state[i]!= target[i]:

count=count+1

return count

def astar(state,target):# Add inputs if more are required

states = [src]

g = 0

visited\_states =[]

while len(states):

print(f"Level: {g}")

moves = []

for state in states:

visited\_states.append(state)

print\_b(state)

if state == target:

print("Success")

return

moves += [move for move in possible\_moves(state, visited\_states) if move not in moves]

costs = [g + h(move, target) for move in moves]

states = [moves[i] for i in range(len(moves)) if costs[i] == min(costs)]

g += 1

print("Fail")

def possible\_moves(state, visited\_state): # Add inputs if more are required

# Find index of empty spot and assign it to b

b = state.index(-1);

# 'd' for down, 'u' for up, 'r' for right, 'l' for left - directions array

d = []

# Add all possible direction into directions array - Hint using if statements

if b - 3 in range(9):

d.append('u')

if b not in [0, 3, 6]:

d.append('l')

if b not in [2, 5, 8]:

d.append('r')

if b + 3 in range(9):

d.append('d')

# If direction is possible then add state to move

pos\_moves = []

# for all possible directions find the state if that move is played

### Jump to gen function to generate all possible moves in the given directions

for m in d:

pos\_moves.append(gen(state, m, b))

# return all possible moves only if the move not in visited\_states

return [move for move in pos\_moves if move not in visited\_state]

def gen(state, m, b):

temp = state.copy()

# if move is to slide empty spot to the left and so on

if m == 'u': temp[b - 3], temp[b] = temp[b], temp[b - 3]

if m == 'l': temp[b - 1], temp[b] = temp[b], temp[b - 1]

if m == 'r': temp[b + 1], temp[b] = temp[b], temp[b + 1]

if m == 'd': temp[b + 3], temp[b] = temp[b], temp[b + 3]

# return new state with tested move to later check if "src == target"

return temp

# Test 1

src = [1, 2, 3, -1, 4, 5, 6, 7, 8]

target = [1, 2, 3, 4, 5, -1, 6, 7, 8]

astar(src, target)

OUTPUT :

Level: 0

1 2 3

4 5

6 7 8

Level: 1

1 2 3

4 5

6 7 8

1 2 3

6 4 5

7 8

Level: 2

1 3

4 2 5

6 7 8

1 2 3

4 5

6 7 8

Success