ROSENIVITHA J - 192424009

**EXP 1: Write a python program to solve 8-puzzle problem**

**AIM:**

To write a program to implement 8-puzzle problem using python

**PROGRAM:**

def print\_board(state):

for i in range(0, 9, 3):

row = state[i:i+3]

print(" ".join(str(x) if x != 0 else " " for x in row))

print()

def manhattan\_distance(state, goal):

distance = 0

for i in range(1, 9): # tiles 1–8

x1, y1 = divmod(state.index(i), 3)

x2, y2 = divmod(goal.index(i), 3)

distance += abs(x1 - x2) + abs(y1 - y2)

return distance

def get\_neighbors(state):

neighbors = []

zero\_index = state.index(0)

x, y = divmod(zero\_index, 3)

moves = {

'UP': (x - 1, y),

'DOWN': (x + 1, y),

'LEFT': (x, y - 1),

'RIGHT': (x, y + 1),

}

for move, (nx, ny) in moves.items():

if 0 <= nx < 3 and 0 <= ny < 3:

new\_index = nx \* 3 + ny

new\_state = list(state)

# Swap blank with target tile

new\_state[zero\_index], new\_state[new\_index] = new\_state[new\_index], new\_state[zero\_index]

neighbors.append((tuple(new\_state), move))

return neighbors

def a\_star(start, goal):

pq = []

heapq.heappush(pq, (manhattan\_distance(start, goal), 0, start, []))

visited = set()

while pq:

f, g, state, path = heapq.heappop(pq)

if state == goal:

return path, state

if state in visited:

continue

visited.add(state)

for neighbor, move in get\_neighbors(state):

if neighbor not in visited:

new\_path = path + [neighbor]

h = manhattan\_distance(neighbor, goal)

heapq.heappush(pq, (g + 1 + h, g + 1, neighbor, new\_path))

return None, None

if \_\_name\_\_ == "\_\_main\_\_":

start\_state = (1, 2, 3,

4, 0, 6,

7, 5, 8)

goal\_state = (1, 2, 3,

4, 5, 6,

7, 8, 0)

print("Initial State:")

print\_board(start\_state)

solution, final\_state = a\_star(start\_state, goal\_state)

if solution:

print("Solution found in", len(solution), "steps:\n")

current = start\_state

print\_board(current)

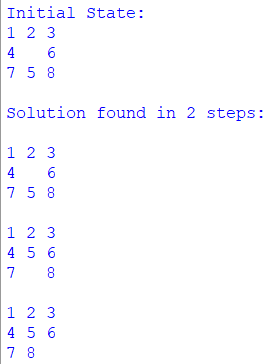
for step in solution:

print\_board(step)

else:

print("No solution exists.")

OUTPUT:



**RESULT:**

Thus, the output is verified for 8-puzzle problem.