**Using misplaced tiles for 8 puzzle game Input –**

from queue import PriorityQueue

class PuzzleState: def \_\_init\_\_(self, tiles, empty\_tile\_index, moves=0):

self.tiles = tiles

self.empty\_tile\_index = empty\_tile\_index self.moves = moves

def is\_goal(self):

return self.tiles == GOAL\_STATE

def get\_possible\_moves(self): index = self.empty\_tile\_index possible\_moves = [] directions = [(-1, 0), (1, 0), (0, -1), (0, 1)] for direction in directions:

new\_index = index + direction[0] \* 3 + direction[1] if 0 <= new\_index < 9:

if direction[0] == -1 and index % 3 != 0: continue # Up if direction[0] == 1 and index % 3 != 2: continue # Down if direction[1] == -1 and index < 6: continue # Left if direction[1] == 1 and index > 2: continue # Right possible\_moves.append(new\_index) return possible\_moves

def generate\_new\_state(self, new\_empty\_index):

new\_tiles = list(self.tiles)

new\_tiles[self.empty\_tile\_index], new\_tiles[new\_empty\_index] = new\_tiles[new\_empty\_index], new\_tiles[self.empty\_tile\_index]

return PuzzleState(tuple(new\_tiles), new\_empty\_index, self.moves + 1)

def heuristic(self):

return sum(1 for i, tile in enumerate(self.tiles) if tile != 0 and tile != GOAL\_STATE[i])

def \_\_lt\_\_(self, other):

return False # This prevents direct comparison; we'll use tuples instead

def a\_star(initial\_state):

open\_set = PriorityQueue() open\_set.put((0, initial\_state)) closed\_set = set()

while not open\_set.empty():

current\_cost, current\_state = open\_set.get()

if current\_state.is\_goal():

return current\_state.moves

closed\_set.add(current\_state.tiles)

for new\_empty\_index in current\_state.get\_possible\_moves():

new\_state = current\_state.generate\_new\_state(new\_empty\_index)

if new\_state.tiles in closed\_set:

continue

cost = new\_state.moves + new\_state.heuristic() open\_set.put((cost, new\_state))

return -1 # If no solution is found

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

# User input for initial state

initial\_tiles = input("Enter the initial state (9 numbers separated by spaces, use 0 for the empty tile): ") initial\_tiles = tuple(map(int, initial\_tiles.split()))

# User input for goal state

goal\_tiles = input("Enter the goal state (9 numbers separated by spaces, use 0 for the empty tile):

")

global GOAL\_STATE

GOAL\_STATE = tuple(map(int, goal\_tiles.split()))

empty\_tile\_index = initial\_tiles.index(0) initial\_state = PuzzleState(initial\_tiles, empty\_tile\_index)

result = a\_star(initial\_state) print(f"Minimum moves to solve the puzzle: {result}")

**output-**

