```
LAB 3
8 PUZZLE USING A* METHOD:

from queue import PriorityQueue

# Goal state
goal_state = ((1, 2, 3),
              (4, 5, 6),
              (7, 8, 0))

# Manhattan distance heuristic
def manhattan(state):
    distance = 0
    for i in range(3):
        for j in range(3):
            value = state[i][j]
            if value != 0:
                goal_x = (value - 1) // 3
                goal_y = (value - 1) % 3
                distance += abs(i - goal_x) + abs(j - goal_y)
    return distance

# Generate neighbors by sliding the blank space
def neighbors(state):
    result = []
    x, y = next((i, j) for i in range(3) for j in range(3) if state[i][j] ==
0)
    directions = [(-1,0),(1,0),(0,-1),(0,1)]  # Up, Down, Left, Right
    for dx, dy in directions:
        new_x, new_y = x + dx, y + dy
        if 0 <= new_x < 3 and 0 <= new_y < 3:
            new_state = [list(row) for row in state]
            new_state[x][y], new_state[new_x][new_y] =
new_state[new_x][new_y], new_state[x][y]
            result.append(tuple(tuple(row) for row in new_state))
    return result

# Reconstruct path
def reconstruct_path(parent, g_score):
    path = []
    current = list(parent.keys())[-1]  # goal node
    while current is not None:
        h = manhattan(current)
        g = g_score[current]
        f = g + h
        path.append((current, g, h, f))
        current = parent[current]
    return path[::-1]
```

```python
# A* algorithm with g, h, f calculation
def a_star_verbose(start_state):
    open_set = PriorityQueue()
    open_set.put((manhattan(start_state), start_state))
    parent = {start_state: None}
    g_score = {start_state: 0}

    while not open_set.empty():
        current = open_set.get()[1]

        if current == goal_state:
            return reconstruct_path(parent, g_score)

        for neighbor in neighbors(current):
            tentative_g = g_score[current] + 1
            if neighbor not in g_score or tentative_g < g_score[neighbor]:
                parent[neighbor] = current
                g_score[neighbor] = tentative_g
                f_score = tentative_g + manhattan(neighbor)
                open_set.put((f_score, neighbor))

    return None

# Example solvable start state (5 moves)
start_state = ((1, 2, 3),
               (0, 4, 6),
               (7, 5, 8))

# Run algorithm
solution = a_star_verbose(start_state)

# Print each step with g, h, f
if solution:
    for step, (state, g, h, f) in enumerate(solution):
        print(f"Step {step}: g={g}, h={h}, f={f}")
        for row in state:
            print(row)
        print()
else:
    print("No solution found.")
```

OUTPUT:

```
PS C:\Users\student> & C:/Users/Admin/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/student/Documents/AI LAB/ASTAR.py"
Step 0: g=0, h=3, f=3
(1, 2, 3)
(0, 4, 6)
(7, 5, 8)

Step 1: g=1, h=2, f=3
(1, 2, 3)
(1, 2, 3)
(1, 2, 3)
(1, 2, 3)
(1, 2, 3)
(1, 2, 3)
(4, 0, 6)
(7, 5, 8)

Step 2: g=2, h=1, f=3
(1, 2, 3)
(4, 5, 6)
(7, 0, 8)

Step 3: g=3, h=0, f=3
(1, 2, 3)
(4, 5, 6)
(7, 8, 0)

PS C:\Users\student>
```