A* ALGORITHM USING MISPLACED TILES

CODE :

```
print(f"Santhosh N (1BM23CS302)")

from queue import PriorityQueue

def misplaced_tiles(state, goal):
    """Heuristic: Count how many tiles are misplaced compared to the goal."""
    return sum(1 for i in range(len(state)) if state[i] != goal[i] and state[i] != '_')

def get_neighbors(state):
    """Return a list of states reachable from the current state by sliding a tile."""
    neighbors = []
    idx = state.index('_')


    moves = []
    row, col = divmod(idx, 3)
    if row > 0: moves.append(idx - 3)
    if row < 2: moves.append(idx + 3)
    if col > 0: moves.append(idx - 1)
    if col < 2: moves.append(idx + 1)

    for move in moves:
        new_state = list(state)

        new_state[idx], new_state[move] = new_state[move], new_state[idx]
        neighbors.append(''.join(new_state))

    return neighbors

def reconstruct_path(came_from, current):
    """Reconstruct the path from start to goal."""
    path = [current]
    while current in came_from:
        current = came_from[current]
        path.append(current)
    path.reverse()
    return path

def a_star(start, goal):
    """A* algorithm to solve 8-puzzle using misplaced tiles heuristic."""
    open_set = PriorityQueue()
    open_set.put((misplaced_tiles(start, goal), 0, start))
```

```
    came_from = {}
    g_score = {start: 0}

    while not open_set.empty():
        f, g, current = open_set.get()

        if current == goal:
            return reconstruct_path(came_from, current)

        for neighbor in get_neighbors(current):
            tentative_g_score = g + 1

            if neighbor not in g_score or tentative_g_score < g_score[neighbor]:
                came_from[neighbor] = current
                g_score[neighbor] = tentative_g_score
                f_score = tentative_g_score + misplaced_tiles(neighbor, goal)
                open_set.put((f_score, tentative_g_score, neighbor))

    return None

def print_state(state):
    """Pretty print the 8-puzzle state."""
    for i in range(0, 9, 3):
        print(state[i:i+3].replace('_', ' '))
    print()

def valid_state(state):
    """Check if input state is valid (length 9 and contains 1-8 and _ exactly once each)."""
    if len(state) != 9:
        return False
    tiles = set(state)
    required_tiles = set('12345678_')
    if tiles != required_tiles:
        return False

    for ch in required_tiles:
        if state.count(ch) != 1:
            return False
    return True

if __name__ == "__main__":
```

A* ALGORITHM USING MISPLACED TILES

```
while True:
    start_state = input("Start state: ").strip()
    if valid_state(start_state):
        break
    print("Invalid state! Please enter exactly 9 characters with digits 1-8 and one '_'.")



while True:
    goal_state = input("Goal state: ").strip()
    if valid_state(goal_state):
        break
    print("Invalid state! Please enter exactly 9 characters with digits 1-8 and one '_'.")

print("\nSolving puzzle...\n")
solution = a_star(start_state, goal_state)

if solution:
    print(f"Solution found in {len(solution) - 1}th Depth\n")
    for step in solution:
        print_state(step)
else:
    print("No solution found.")

print(f"TOTAL COST IS {len(solution) - 1}\n")
```

A* ALGORITHM USING MISPLACED TILES

OUTPUT :

```
= RESTART: C:/Users/student/AppData/Local/Programs/Python/Python313/302/lab4_mis
.py
Santhosh N (1BM23CS302)
Start state: 2831647_5
Goal state: 1238_4765

Solving puzzle...

Solution found in 5th Depth

283
164
7 5

283
1 4
765

2 3
184
765

 23
184
765

123
 84
765

123
8 4
765

TOTAL COST IS 5
```