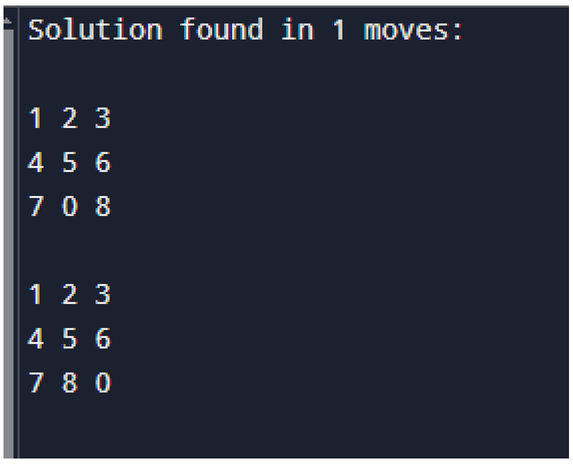
(1) 8 PUZZLE GAME USING BFS

```python
from collections import deque


class PuzzleState:
    def __init__(self, board, zero_position, moves=0, previous=None):
        self.board = board
        self.zero_position = zero_position
        self.moves = moves
        self.previous = previous

    def __str__(self):
        return '\n'.join([' '.join(row) for row in self.board])

    def is_goal(self, goal):
        return self.board == goal

    def get_possible_moves(self):
        row, col = self.zero_position
        possible_moves = []
        directions = [(-1, 0), (1, 0), (0, -1), (0, 1)]  # Up, Down, Left, Right

        for dr, dc in directions:
            new_row, new_col = row + dr, col + dc
            if 0 <= new_row < 3 and 0 <= new_col < 3:
                possible_moves.append((new_row, new_col))

        return possible_moves

    def move(self, new_zero_position):
        new_board = [row[:] for row in self.board]  # Create a copy of the board
        r1, c1 = self.zero_position
        r2, c2 = new_zero_position

        # Swap the zero with the target position
        new_board[r1][c1], new_board[r2][c2] = new_board[r2][c2], new_board[r1][c1]
        return PuzzleState(new_board, new_zero_position, self.moves + 1, self)

def bfs(start, goal):
    queue = deque([start])
    visited = set()
    visited.add(tuple(map(tuple, start.board)))

    while queue:
        current_state = queue.popleft()

        if current_state.is_goal(goal):
            return current_state

        for move in current_state.get_possible_moves():
            new_state = current_state.move(move)

            if tuple(map(tuple, new_state.board)) not in visited:
                visited.add(tuple(map(tuple, new_state.board)))
                queue.append(new_state)

    return None

def print_solution(solution):
    moves = []
    while solution:
        moves.append(solution)
        solution = solution.previous
    for state in reversed(moves):
        print(state)
        print()

if __name__ == "__main__":
    start_board = [
        ['1', '2', '3'],
        ['4', '5', '6'],
        ['7', '0', '8']
    ]

    goal_board = [
        ['1', '2', '3'],
        ['4', '5', '6'],
        ['7', '8', '0']
    ]

    zero_position = (2, 1)  # Initial position of the zero

    start_state = PuzzleState(start_board, zero_position)
    goal_state = goal_board

    solution = bfs(start_state, goal_state)

    if solution:
        print("Solution found in {} moves:\n".format(solution.moves))
        print_solution(solution)
    else:
        print("No solution found.")
```

OUTPUT:



```
Solution found in 1 moves:

1 2 3
4 5 6
7 0 8

1 2 3
4 5 6
7 8 0
```