In [*]:

```
import random

# Constants
GRID_SIZE = 3
POPULATION_SIZE = 100
INITIAL_CONFIGURATION = [[2, 8, 3], [1, 6, 4], [7, None, 5]]
TARGET_CONFIGURATION = [[1, 2, 3], [8, None, 4], [7, 6, 5]]

def generate_initial_configuration():
    # Shuffle the target configuration to create a solvable initial configuration
    initial = sum(INITIAL_CONFIGURATION, [])
    random.shuffle(initial)
    return [initial[i:i+GRID_SIZE] for i in range(0, GRID_SIZE*GRID_SIZE, GRID_SI

# Function to check if the current configuration is the target configuration
def is_target_configuration(grid):
    return grid == TARGET_CONFIGURATION

# Function to display the grid
def display_grid(grid):
    for row in grid:
        for tile in row:
            if tile is None:
                print('  ', end=' ')
            else:
                print(str(tile).rjust(2), end=' ')
        print()

# Function to get player's move
def get_move():
    move = input('Enter move (up/down/left/right): ')
    while move not in ['up', 'down', 'left', 'right']:
        move = input('Invalid move! Enter move (up/down/left/right): ')
    return move

# Function to update grid based on player's move
def update_grid(grid, move):
    empty_row, empty_col = get_empty_position(grid)
    if move == 'up':
        if empty_row == GRID_SIZE - 1:
            return False
        grid[empty_row][empty_col], grid[empty_row + 1][empty_col] = grid[empty_r
    elif move == 'down':
        if empty_row == 0:
            return False
        grid[empty_row][empty_col], grid[empty_row - 1][empty_col] = grid[empty_r
    elif move == 'left':
        if empty_col == GRID_SIZE - 1:
            return False
        grid[empty_row][empty_col], grid[empty_row][empty_col + 1] = grid[empty_r
    elif move == 'right':
        if empty_col == 0:
            return False
        grid[empty_row][empty_col], grid[empty_row][empty_col - 1] = grid[empty_r
    return True

# Function to get position of empty tile
def get_empty_position(grid):
    for row in range(GRID_SIZE):
```

```
        for col in range(GRID_SIZE):
            if grid[row][col] is None:
                return row, col

# Main loop
def play_game():
    # Generate initial configuration
    grid = generate_initial_configuration()

    # Game loop
    while not is_target_configuration(grid):
        # Display the current grid
        print('Current configuration:')
        display_grid(grid)

        # Get player's move
        move = get_move()

        # Update grid based on player's move
        if not update_grid(grid, move):
            print('Invalid move!')
            continue


play_game()
```

```
Current configuration:
 3  4  2
 6     5
 7  8  1
Enter move (up/down/left/right): up
Current configuration:
 3  4  2
 6  8  5
 7     1

Enter move (up/down/left/right):
```

In [ ]: