Ai assisted coding lab test 2

Name:N.Prudhvi

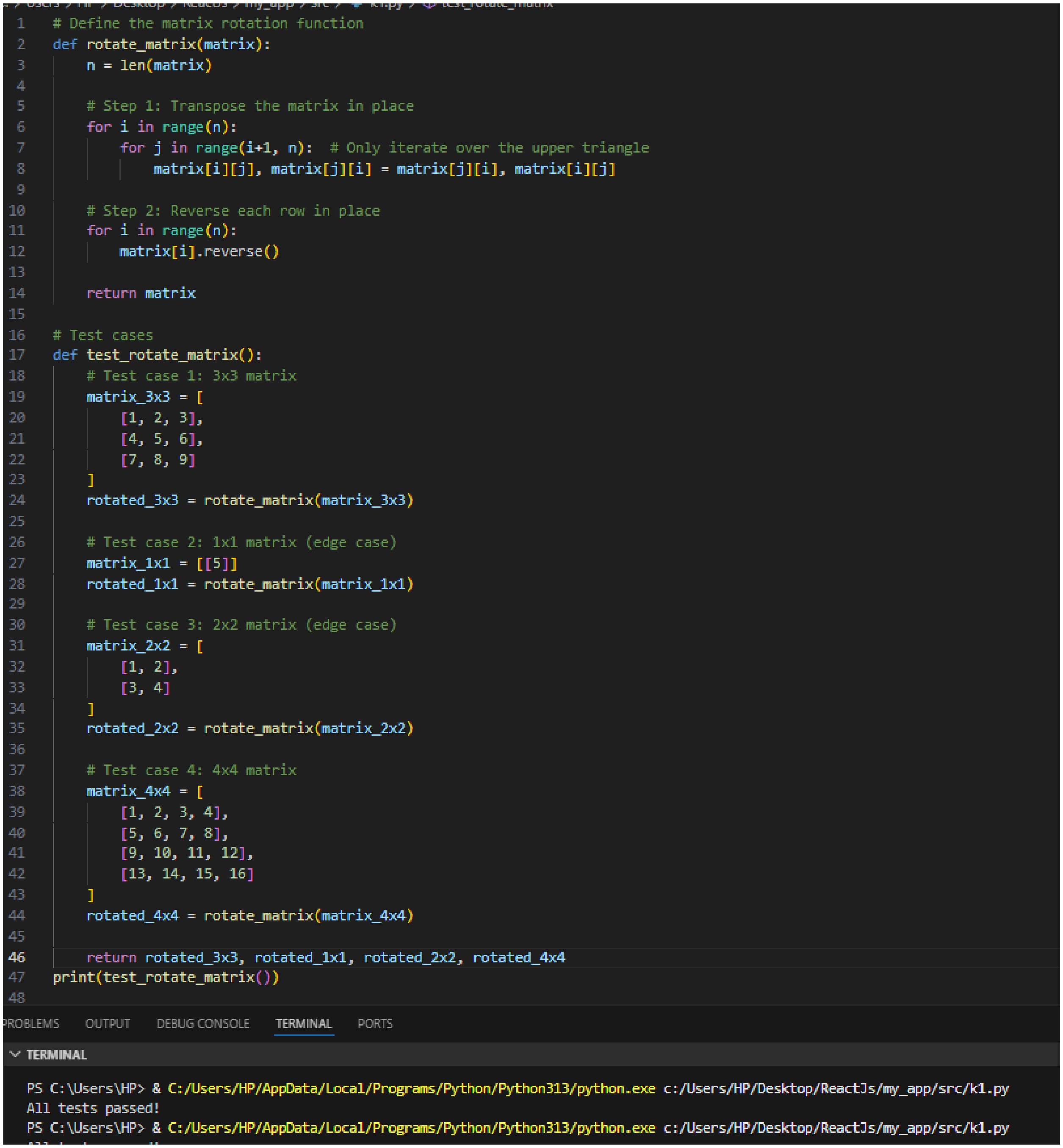
HT.NO:2403A510G7

Set k: K.1 — [S09K1] Rotate NxN matrix 90° clockwise

Prompt:

Design a function that takes a square matrix (NxN) and rotates it by 90 degrees in the clockwise direction, modifying the matrix in-place. Your solution should be able to handle cases where the matrix size is 1x1 or 2x2.

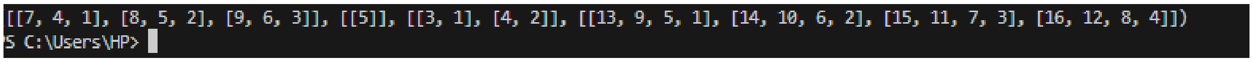
Code:



```python
# Define the matrix rotation function
def rotate_matrix(matrix):
    n = len(matrix)

    # Step 1: Transpose the matrix in place
    for i in range(n):
        for j in range(i+1, n):  # Only iterate over the upper triangle
            matrix[i][j], matrix[j][i] = matrix[j][i], matrix[i][j]

    # Step 2: Reverse each row in place
    for i in range(n):
        matrix[i].reverse()

    return matrix

# Test cases
def test_rotate_matrix():
    # Test case 1: 3x3 matrix
    matrix_3x3 = [
        [1, 2, 3],
        [4, 5, 6],
        [7, 8, 9]
    ]
    rotated_3x3 = rotate_matrix(matrix_3x3)

    # Test case 2: 1x1 matrix (edge case)
    matrix_1x1 = [[5]]
    rotated_1x1 = rotate_matrix(matrix_1x1)

    # Test case 3: 2x2 matrix (edge case)
    matrix_2x2 = [
        [1, 2],
        [3, 4]
    ]
    rotated_2x2 = rotate_matrix(matrix_2x2)

    # Test case 4: 4x4 matrix
    matrix_4x4 = [
        [1, 2, 3, 4],
        [5, 6, 7, 8],
        [9, 10, 11, 12],
        [13, 14, 15, 16]
    ]
    rotated_4x4 = rotate_matrix(matrix_4x4)

    return rotated_3x3, rotated_1x1, rotated_2x2, rotated_4x4
print(test_rotate_matrix())
```

```
PS C:\Users\HP> & C:/Users/HP/AppData/Local/Programs/Python/Python313/python.exe c:/Users/HP/Desktop/ReactJs/my_app/src/k1.py
All tests passed!
PS C:\Users\HP> & C:/Users/HP/AppData/Local/Programs/Python/Python313/python.exe c:/Users/HP/Desktop/ReactJs/my_app/src/k1.py
```

Output:



```
[[7, 4, 1], [8, 5, 2], [9, 6, 3]], [[5]], [[3, 1], [4, 2]], [[13, 9, 5, 1], [14, 10, 6, 2], [15, 11, 7, 3], [16, 12, 8, 4]])
S C:\Users\HP>
```

Observations:

- **In-place Transformation**: The solution rotates the matrix in-place without using additional space for another matrix.

- **Edge Case Handling**: The approach works efficiently even for small matrices, such as 1x1 and 2x2.

    - **1x1** matrix doesn't change after rotation, which the code handles naturally.
    - **2x2** matrix needs minimal swaps but follows the same logic as larger matrices.

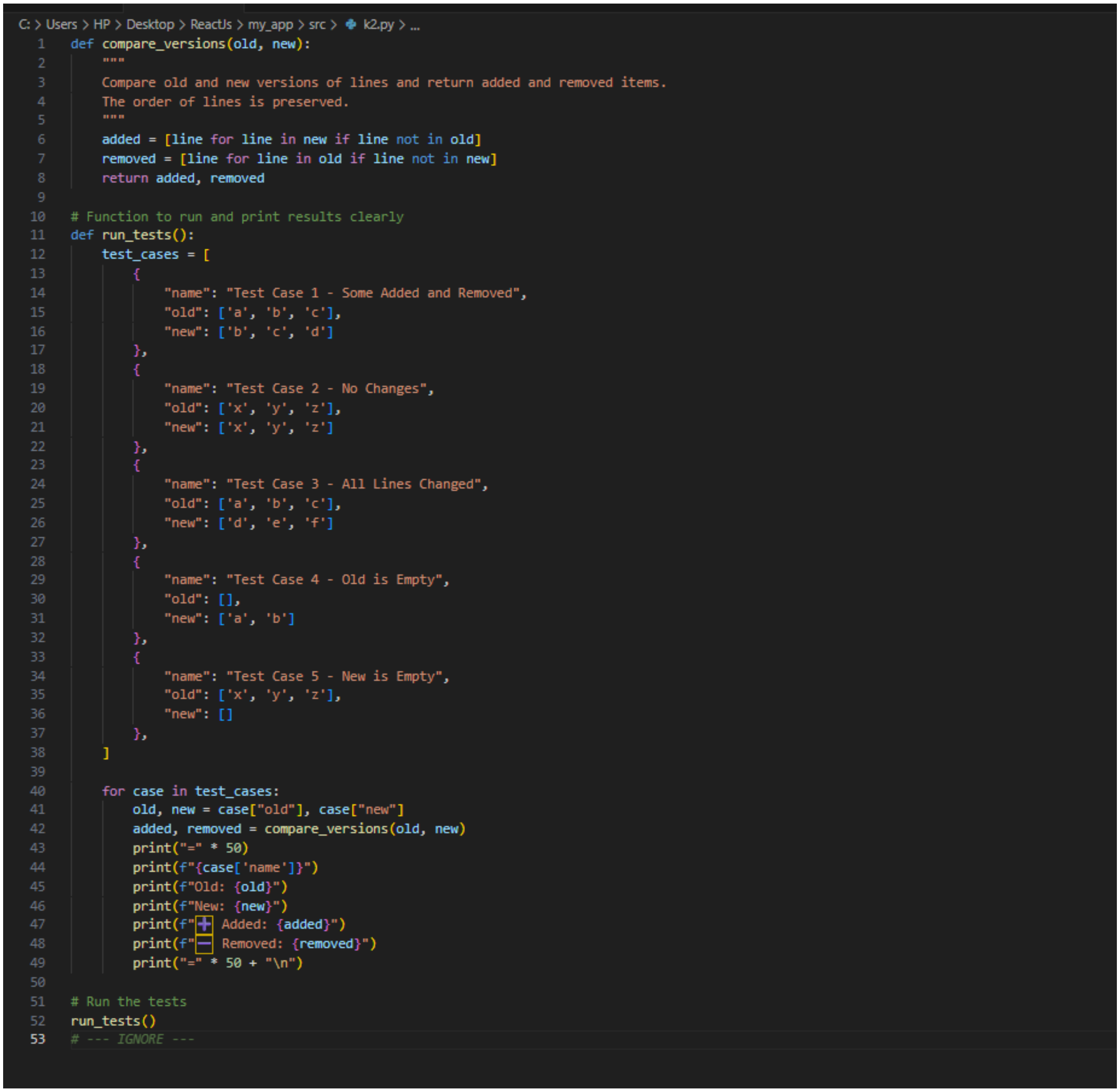
- **Time Complexity**: The time complexity is O(N2)O(N^2)O(N2) because we need to visit every element at least once during both the transpose and the row-reversal steps.

K.2 — [S09K2] Compute added/removed lines

Prompt:

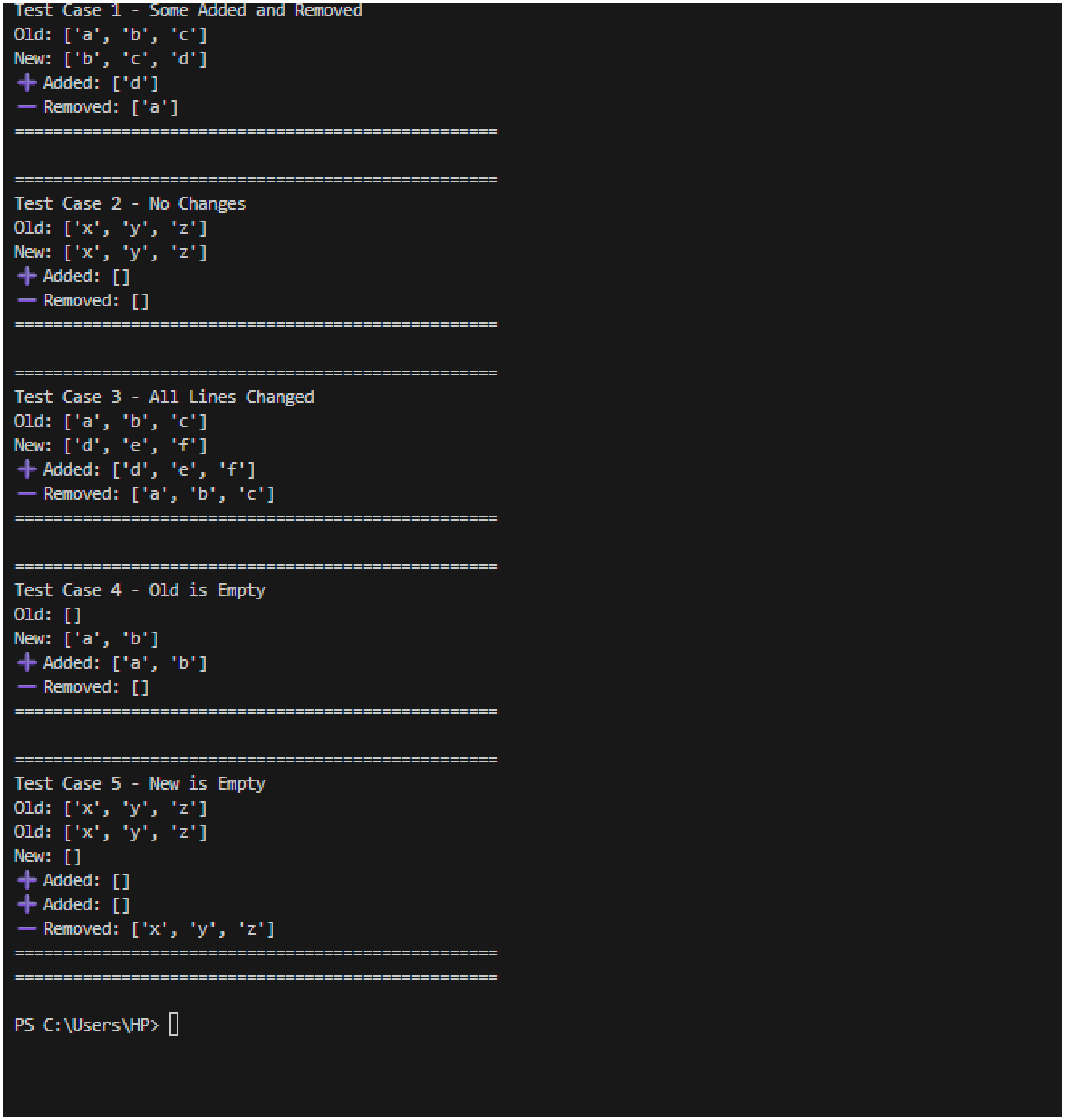
Create a function that takes two lists, `old` and `new`, representing lines from two versions of a document. The function should return two lists: one for lines that were added and one for lines that were removed, ensuring that the order of lines is preserved and no duplicates appear in the output.

CODE:


C: > Users > HP > Desktop > ReactJs > my_app > src > k2.py > ...

```python
def compare_versions(old, new):
    """
    Compare old and new versions of lines and return added and removed items.
    The order of lines is preserved.
    """
    added = [line for line in new if line not in old]
    removed = [line for line in old if line not in new]
    return added, removed

# Function to run and print results clearly
def run_tests():
    test_cases = [
        {
            "name": "Test Case 1 - Some Added and Removed",
            "old": ['a', 'b', 'c'],
            "new": ['b', 'c', 'd']
        },
        {
            "name": "Test Case 2 - No Changes",
            "old": ['x', 'y', 'z'],
            "new": ['x', 'y', 'z']
        },
        {
            "name": "Test Case 3 - All Lines Changed",
            "old": ['a', 'b', 'c'],
            "new": ['d', 'e', 'f']
        },
        {
            "name": "Test Case 4 - Old is Empty",
            "old": [],
            "new": ['a', 'b']
        },
        {
            "name": "Test Case 5 - New is Empty",
            "old": ['x', 'y', 'z'],
            "new": []
        },
    ]

    for case in test_cases:
        old, new = case["old"], case["new"]
        added, removed = compare_versions(old, new)
        print("=" * 50)
        print(f"{case['name']}")
        print(f"Old: {old}")
        print(f"New: {new}")
        print(f"➕ Added: {added}")
        print(f"➖ Removed: {removed}")
        print("=" * 50 + "\n")

# Run the tests
run_tests()
# --- IGNORE ---
```

Output:



```
Test Case 1 - Some Added and Removed
Old: ['a', 'b', 'c']
New: ['b', 'c', 'd']
➕ Added: ['d']
➖ Removed: ['a']
==================================================

==================================================
Test Case 2 - No Changes
Old: ['x', 'y', 'z']
New: ['x', 'y', 'z']
➕ Added: []
➖ Removed: []
==================================================

==================================================
Test Case 3 - All Lines Changed
Old: ['a', 'b', 'c']
New: ['d', 'e', 'f']
➕ Added: ['d', 'e', 'f']
➖ Removed: ['a', 'b', 'c']
==================================================

==================================================
Test Case 4 - Old is Empty
Old: []
New: ['a', 'b']
➕ Added: ['a', 'b']
➖ Removed: []
==================================================

==================================================
Test Case 5 - New is Empty
Old: ['x', 'y', 'z']
Old: ['x', 'y', 'z']
New: []
➕ Added: []
➕ Added: []
➖ Removed: ['x', 'y', 'z']
==================================================
==================================================

PS C:\Users\HP> 
```

Observations:

1)Order is preserved: The output maintains the original order of lines in both `old` and `new` lists, which is critical for readability in version diffs.

2)**Only differences shown**: Unchanged lines are excluded, focusing the output solely on what's added or removed—ideal for change reviews in sports analytics.

3) **No duplicates in output**: Since we're only listing lines that are strictly in one list and not the other, there's no risk of duplicated entries.

4)**Handles edge cases**:

- Works correctly when either list is empty.
- Correctly identifies full replacements (all lines changed).