```
import numpy as np


# 1.2 - Simple PageRank
def calculate_pagerank(M, num_iterations=20, tolerance=1e-6):
    n = M.shape[0]
    r = np.ones(n) # Initialize r0 with equal probabilities
    results = [r.copy()]

    for i in range(num_iterations):
        r_next = M @ r
        results.append(r_next.copy())
        if np.allclose(r, r_next, rtol=tolerance): # Check for convergence
            break
        r = r_next
    return results

M = np.array([
    [0, 1/2, 1],
    [1/2, 0, 0],
    [1/2, 1/2, 0]
])

iterations = calculate_pagerank(M)

print("\nSimplified PageRank Iterations:")
print("-" * 40)
for i, r in enumerate(iterations):
    print(f"Iteration {i}:")
    for j, value in enumerate(r):
        page = chr(65 + j)  # Convert 0,1,2 to A,B,C
        print(f"Page {page}: {value:.6f}")
    print()

print("Final converged values:")
print("-" * 40)
final_values = iterations[-1]
for i, value in enumerate(final_values):
    page = chr(65 + i)
    print(f"Page {page}: {value:.6f}")


# 2 - Modified PageRank
def calculate_pagerank_with_damping(links, damping=0.85, num_iterations=50,
tolerance=1e-6):
    n = len(links)
    # Initialize PageRank scores with 1/n
    scores = {page: 1.0 for page in links}
    iterations = [{page: score for page, score in scores.items()}]

    for iteration in range(num_iterations):
        new_scores = {}
        max_change = 0

        for page in links:
            # Calculate sum of contributions from incoming links
            incoming_pr = sum(scores[src_page] / len(links[src_page])
                            for src_page, targets in links.items()
                            if page in targets)
            # Apply damping factor formula: (1-d) + d * (sum of contributions)
            new_score = (1 - damping) + damping * incoming_pr
            max_change = max(max_change, abs(new_score - scores[page]))
            new_scores[page] = new_score

        # Store this iteration's results
        iterations.append({page: score for page, score in new_scores.items()})
        scores = new_scores

        # Check for convergence
        if max_change < tolerance:
            break

    return iterations

links = {
    'A': ['B', 'C'],  # Page A links to B and C
    'B': ['A'],       # Page B links to A
    'C': ['A']        # Page C links to A
}

# Calculate PageRank
iterations = calculate_pagerank_with_damping(links)

# Print results for each iteration
print("\nModified PageRank Iterations:")
print("-" * 50)
for i, scores in enumerate(iterations):
    print(f"\nIteration {i}:")
    for page, score in sorted(scores.items()):
        print(f"PR({page}) = {score:.6f}")

print("\nFinal converged values:")
print("-" * 50)
final_scores = iterations[-1]
for page, score in sorted(final_scores.items()):
    print(f"PR({page}) = {score:.6f}")

# Print the number of iterations needed for convergence
print(f"\nConverged after {len(iterations)-1} iterations")
```