PAREENITA A.SHIRSATH B.E.A.I.&.D.S. ROLL.NO : 57

BDA EXPERIMENT NO : 10

```
from collections import defaultdict

# Each page points to a list of outgoing links
graph = {
    'A': ['B', 'C'],
    'B': ['C', 'B'],
    'C': ['A', 'C'],
    'D': ['C']
}
# Initialize PageRank values
num_pages = len(graph)
page_rank = {page: 1.0 / num_pages for page in graph}

# Damping factor
d = 0.85
num_iterations = 10

# Map function
def mapper(page, rank, links):
    if len(links) == 0:
        return [(page, 0)]
    contribution = rank / len(links)
    return [(linked_page, contribution) for linked_page in links]

# Reduce function
def reducer(mapped_values):
    new_ranks = defaultdict(float)
    for page, contribution in mapped_values:
        new_ranks[page] += contribution
    return new_ranks

# PageRank Iteration
for iteration in range(num_iterations):
    mapped = []
    # Map step
    for page, rank in page_rank.items():
        mapped.extend(mapper(page, rank, graph[page]))

    # Reduce step
    new_ranks = reducer(mapped)

    # Apply damping factor
    for page in graph:
        new_ranks[page] = (1 - d)/num_pages + d * new_ranks.get(page, 0.0)

    page_rank = new_ranks

# Print final PageRank values
print("Final PageRank after {} iterations:".format(num_iterations))
for page, rank in sorted(page_rank.items()):
    print(f"{page}: {rank:.4f}")
```

```
Final PageRank after 10 iterations:
A: 0.2408
B: 0.2432
C: 0.4784
D: 0.0375
```