AI ASSISTED CODING

LAB EXAM-2

NAME:B PRANEETH

H NO:2403A51246

BATCH:11

Subgroup:N

N.1-{S18N1]Compute directory sizes from listing

CONTEXT:

Folder size accounting in real estate listings platform.

Your Task:

Aggregate child sizes to parent paths.

Data & Edge Cases:

Listing '/a 10', '/a/b 5', '/a/c 7'.

AI Assistance Expectation:

Prefix-walk strategy + tests.

Constraints & Notes:

Include parents once.

Sample Input

['/a 10', '/a/b 5', '/a/c 7']

Sample Output

{'/a':22,'/a/b':5,'/a/c':7}

Acceptance Criteria: Parents include children

Code:

```python
def aggregate_folder_sizes(listings):
    """
    Aggregates folder sizes so that each parent includes the sizes of all its children.
    Args:
        listings (list of str): Each entry is of the form '/path size'.
    Returns:
        dict: Mapping from path to aggregated size.
    """
    from collections import defaultdict
    sizes = defaultdict(int)
    for entry in listings:
        path, size = entry.rsplit(' ', 1)
        size = int(size)
        # Walk up the path, adding size to each parent
        parts = path.strip('/').split('/')
        for i in range(1, len(parts)+1):
            parent = '/' + '/'.join(parts[:i])
            sizes[parent] += size
    return dict(sizes)


def test_aggregate_folder_sizes():
    # Sample input
    listings = ['/a 10', '/a/b 5', '/a/c 7']
    expected = {'/a': 22, '/a/b': 5, '/a/c': 7}
    result = aggregate_folder_sizes(listings)
    assert result == expected, f"Expected {expected}, got {result}"

    # Edge: single root
    listings = ['/root 100']
    expected = {'/root': 100}
    assert aggregate_folder_sizes(listings) == expected

    # Edge: nested
    listings = ['/x 1', '/x/y 2', '/x/y/z 3']
    expected = {'/x': 6, '/x/y': 5, '/x/y/z': 3}
    assert aggregate_folder_sizes(listings) == expected

    # Edge: multiple roots
    listings = ['/a 1', '/b 2', '/a/b 3']
    expected = {'/a': 4, '/a/b': 3, '/b': 2}
    assert aggregate_folder_sizes(listings) == expected

    print("All tests passed.")
```

Output:

```
All tests passed.
PS C:\Users\rushi\OneDrive\Desktop\msd> & C:/Users/rushi/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/Users/rushi
/OneDrive/Desktop/msd/folder_size_aggregate.py
All tests passed.
PS C:\Users\rushi\OneDrive\Desktop\msd> & C:/Users/rushi/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/Users/rushi
/OneDrive/Desktop/msd/folder_size_aggregate.py
```

N.2-{S18N12}COMPUTE SIMPLE SENTIMENT SCORE

Context:

Lexicon-based sentiment in real estate listings platform.

Your Task:

Score with good=+1, great=+2, bad=-1.

Data & Edge Cases:

Tokenize on spaces; strip punctuation.

AI Assistance Expectation:

AI for tests and simple tokenizer.

Constraints & Notes:

Return integer sum.

Sample Input

good product with bad packaging but great value

Sample Output

2

Acceptance Criteria: Correct sum

Code:

```python
def sentiment_score(text):
    import string
    lex = {'good':1, 'great':2, 'bad':-1}
    return sum(lex.get(w.strip(string.punctuation).lower(),0) for w in text.split(



def test_sentiment_score():
    assert sentiment_score('good product with bad packaging but great value') == 2
    assert sentiment_score('good! bad, great.') == 2
    assert sentiment_score('Good GREAT Bad') == 2
    assert sentiment_score('neutral listing only') == 0
    assert sentiment_score('good good bad great great bad') == 4
    print("All sentiment tests passed.")

if __name__ == "__main__":
    test_sentiment_score()
```

Output:

```
PS C:\Users\rushi\OneDrive\Desktop\msd> & C:/Users/rushi/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/Users/rushi
/OneDrive/Desktop/msd/sentiment_score.py
All sentiment tests passed.
PS C:\Users\rushi\OneDrive\Desktop\msd> & C:/Users/rushi/AppData/Local/Microsoft/WindowsApps/python3.11.exe c:/Users/rushi
/OneDrive/Desktop/msd/sentiment_score.py
```