AI-ASSISTED CODING LAB TEST-2

NAME: BANDLA SAI NANDAN REDDY

HALL TICKET: 2403A510C5

BATCH : 05

SUBGROUP :C

C.1 — [S09C1] Debug de-duplication (case-insensitive)

Scenario (sports analytics):  
**Context:**  
Customer contact lists in the sports analytics CRM contain duplicates differing only by case (e.g., 'A@x.com' vs 'a@x.com').

**Your Task:**  
Write a function that returns the first occurrence of each email (case-insensitive) while preserving the original order.  
**Data & Edge Cases:**  
**Input:** list of emails. Normalize for comparison using lowercase; keep the original cased value for output.  
**AI Assistance Expectation:**  
Use AI to spot the bug (reinitializing `seen` in a loop) and propose a corrected, stable algorithm.  
**Constraints & Notes:**  
Include unit tests covering: ['A@x.com','a@x.com','B@y.com'] -> ['A@x.com','B@y.com']  
**Sample Input:**  
['A@x.com', 'a@x.com', 'B@y.com']  
**Sample Output:**['A@x.com', 'B@y.com'] Acceptance Criteria: Preserves first occurrence order; case-insensitive matching.

**PROMPT:**

Write a Python function that removes case-insensitive duplicate emails from a list while preserving the first occurrence and original casing. Make sure to avoid the bug of reinitializing the seen set inside the loop. Include unit tests with examples like ['A@x.com','a@x.com','B@y.com'] -> ['A@x.com','B@y.com'].

**CODE GENERATED WITH BUG:**

def deduplicate\_emails\_buggy(emails):

    result = []

    for email in emails:

        seen = set()

        normalized = email.lower()

        if normalized not in seen:

            seen.add(normalized)

            result.append(email)

    return result

emails = [

    "A@x.com", "a@x.com", "B@y.com", "b@Y.com",

    "C@z.com", "c@Z.com", "A@X.COM"

]

print("duplicates:”,deduplicate\_emails\_buggy(emails))

**OUTPUT (NOT CORRECT):**

****

**Observation :**

The buggy code keeps all emails because seen is reset in every loop, so duplicates are not removed.

Corrected code by AI:

def deduplicate\_emails(emails):

    seen = set()  # ✅ initialize once

    result = []

    for email in emails:

        normalized = email.lower()

        if normalized not in seen:

            seen.add(normalized)

            result.append(email)

    return result

# Test Data

emails = [

    "A@x.com", "a@x.com", "B@y.com", "b@Y.com",

    "C@z.com", "c@Z.com", "A@X.COM"

]

print("Corrected Output:", deduplicate\_emails(emails))

OUTPUT:



OBSERVATION:

The corrected code removes duplicates properly because seen is created once outside the loop and remembers already-seen emails.

**C.2 — [S09C2] TDD: slugify titles**  
Scenario (sports analytics):  
**Context:**  
Content titles in the sports analytics CMS must become SEO-friendly slugs for URLs.  
**Your Task:**  
Design tests first for slugify(text) then implement: lowercase, remove non-alnum except  
hyphen, spaces->hyphen, collapse multiple hyphens, trim hyphens.  
**Data & Edge Cases:**  
Test punctuation, multiple spaces, and boundary hyphens.  
**AI** **Assistance Expectation:**  
Use AI to generate parameterized tests (pytest) and then implement a regex-based slugify.  
**Constraints & Notes:**  
Return correct slugs for provided samples.  
**Sample Input:**  
['Hello World!', 'AI & You', 'Set9-C2']  
**Sample Output:**  
['hello-world', 'ai-you', 'set9-C2']  
Acceptance Criteria: All tests pass; edge cases covered.

**PROMPT:**

Write a Python function slugify(text) that converts titles into SEO-friendly slugs: lowercase, replace spaces with hyphens, remove non-alphanumeric characters except hyphen, collapse multiple hyphens, and trim leading/trailing hyphens. First, create parameterized pytest tests (including punctuation, multiple spaces, and boundary hyphens) and then implement slugify using regex. Example: ['Hello World!', 'AI & You', 'Set9-C2'] → ['hello-world', 'ai-you', 'set9-c2'].

**CODE GENERATED:**

import re

def slugify(text: str) -> str:

    """

    Convert text into an SEO-friendly slug:

    - Lowercase

    - Replace spaces with hyphens

    - Remove non-alphanumeric characters except hyphen

    - Collapse multiple hyphens

    - Trim leading/trailing hyphens

    """

    text = text.lower()

    text = re.sub(r'[^a-z0-9\s-]', '', text)   # remove special chars except space & hyphen

    text = re.sub(r'\s+', '-', text)           # spaces -> hyphen

    text = re.sub(r'-{2,}', '-', text)         # collapse multiple hyphens

    text = text.strip('-')                     # trim boundary hyphens

    return text

# --------- Tests (simple asserts) ---------

assert slugify("Hello World!") == "hello-world"

assert slugify("AI & You") == "ai-you"

assert slugify("Set9-C2") == "set9-c2"

assert slugify("Multiple    Spaces") == "multiple-spaces"

assert slugify("--Sports--") == "sports"

assert slugify("!!!Goals!!!") == "goals"

assert slugify("Mixed--Hyphen---Test") == "mixed-hyphen-test"

assert slugify("   Trim Spaces   ") == "trim-spaces"

assert slugify("") == ""

print("✅ All tests passed!")

# --------- Demo Run ---------

samples = ['Hello World!', 'AI & You', 'Set9-C2']

print("Demo Output:", [slugify(s) for s in samples])

OUTPUT:



OBSERVATION:

The slugify function successfully converts any title into a clean, SEO-friendly slug by making it lowercase, replacing spaces with hyphens, removing unwanted characters, collapsing extra hyphens, and trimming hyphens from the edges.