Project Title: *Survey Feedback Analyzer*

Problem Statement:

In any data-driven domain, especially in Data Science, analyzing textual feedback is a critical step for extracting insights, improving services, and understanding user sentiment. This project aims to help you apply core Python programming concepts to build a **text-based feedback analysis tool**. You will work with survey feedback entries stored in a dictionary of lists, perform basic data cleaning, extract meaningful insights, and apply logic using loops, conditionals, string operations, and user-defined functions.

Project Steps and Objectives:

📥 Step 1: Preloaded Feedbacks (Given Data) *(1 mark)*

Start your program with the following dictionary of lists, preloaded with 10 feedbacks:

```
feedback_data = {
    'S_No': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10],
    'Name': ['Ravi', 'Meera', 'Sam', 'Anu', 'Raj', 'Divya', 'Arjun', 'Kiran', 'Leela', 'Nisha'],
    'Feedback': [
        '  Very GOOD Service!!!',
        'poor support,   not happy   ',
        'GREAT experience! will come again.',
        'okay   okay...',
        '  not   BAD',
        'Excellent care, excellent staff!',
        'good food and good ambience!',
        'Poor response and poor handling of issue',
        'Satisfied. But could be better.',
        'Good support... quick service.'
    ],
    'Rating': [5, 2, 5, 3, 2, 5, 4, 1, 3, 4]
}
```

✍️ Step 2: Add More Feedbacks *(4 marks)*

- Ask the user to enter how many more feedbacks they want to add.
- For each feedback, collect the following inputs from the user:
    - **Name**
    - **Written Feedback (text)**
    - **Rating (1–5)**

- Automatically increment `S_No` starting from 11 onward.
- Append all new data into the `feedback_data` dictionary.

🧹 Step 3: Text Cleaning *(4 marks)*

Clean all feedback entries as follows:

- Remove punctuation (`.`, `,`, `!`, `?`)
- Replace multiple spaces with a **single space**
- Remove leading and trailing spaces
- Convert all text to **lowercase**

> 💡 *Tip: Use `.replace()`, `.split()` and `' '.join()` creatively.*

🔁 Step 4: Word Count Insights *(4 marks)*

Create a function `count_word_in_feedbacks(word)` that:

- Takes a word as input.
- Returns how many feedbacks **contain that word** (case-insensitive match).

Use this function to print:

- Number of feedbacks containing `"good"`
- Number of feedbacks containing `"poor"`
- Number of feedbacks containing `"excellent"`

📊 Step 5: Final Summary & Insights *(12 marks)*

- Display the final cleaned `feedback_data` (dictionary of lists).
- Print the **average rating** from all feedbacks.
- Find and display the feedback with the **longest comment** (in terms of word count).
- Print the **list of unique words** used across all feedbacks (avoid duplicates).

> 🧩 *(Optional)*: Sort feedbacks by **rating (highest to lowest)** using `zip()` and `sorted()`. Display the sorted entries.
