SE302 - Software Testing and Maintenance

Homework 01

Homework Assignment

Task 1 Description - Equivalence Class Testing for Shoe Store System

You are given a part of a software system designed for entering shoe items and their available sizes in a shoe store. The system reads a text file where each line contains:

- **Item name:** Alphabetic characters only, length between 2 and 15 characters.
- **Sizes:** Whole numbers in the range of 26 to 55.
- **Format:** `itemName,size1,size2,...`
- Sizes must be listed in ascending order.
- Maximum of five sizes per item.
- Commas are used as separators.
- Blank spaces should be ignored during input processing.

Your Tasks

1. Identify all equivalence classes for this system. Include both valid and invalid classes.
2. Design a set of test cases that covers all equivalence classes. For each test case, specify:
   - Input data (as a single line in the required format).
   - Expected output (VALID or INVALID).
   - Which equivalence classes are covered.
3. Ensure that your test set achieves full coverage of all identified equivalence classes.

Example Format

#	Test Input	Expected Output	Covered Classes
1	reebok,40,43	VALID	1, 4, 7, 9, 10, 13, 16, 18, 20
2	nike,41,42,43,44,45,46	INVALID	17

Task 2 Description - User Story, Test Scenarios, and Test Cases

You are part of a software testing team responsible for verifying the functionality of an online library system. This system allows registered users to search for books, borrow available ones, and return them later. Your task is to analyze the borrowing feature and design suitable test documentation.

1. User Story

As a ...,
I want ...,
So that ...

2. Acceptance Criteria

1. Users must be logged in to borrow a book.
2. Each user can borrow a maximum of five (5) books at once.
3. A book can be borrowed only if it is marked as "Available".
4. When borrowed, the book's status changes to "Borrowed".
5. The system should prevent a user from borrowing the same book twice.
6. Upon successful borrowing, the system must display a confirmation message:

   ```
   "Book '<Title>' successfully borrowed."
   ```

7. If borrowing is not possible, the system should display an appropriate error message.

3. Student Tasks

Each student must:

a. Write the user story in their own words (you may extend it to include returning books or searching functionality).

b. Derive a list of test scenarios based on the acceptance criteria.

c. Design detailed test cases for each scenario.

d. Prepare a short test execution report showing which tests would pass or fail.

4. Example Test Scenarios

Scenario ID	Scenario Description
TS1	Borrowing a book when the user is logged in and the book is available.
TS2	Attempt to borrow a book when the user is not logged in.

5. Example Test Cases

Test Case ID	Scenario	Preconditions / Test Data	Test Steps	Expected Result
TC1.1	TS1	User logged in; book “The Hobbit” is available.	1. Log in as a user. 2. Search for “The Hobbit”. 3. Click “Borrow”.	Book status changes to “Borrowed”. Message: “Book ’The Hobbit’ successfully borrowed.”
TC2.1	TS2	User not logged in; book “1984” is available.	1. Without logging in, search “1984”. 2. Click “Borrow”.	Error message: “You must be logged in to borrow books.”

6. Optional Extensions

For additional credit, students may:

- Add a “Return Book” feature with corresponding user stories and tests.
- Design test cases for the “Search and Filter” functionality.
- Apply boundary value analysis for the five-book borrowing limit.

7. Deliverables

Each submission should include:

i. User Story and Acceptance Criteria.

ii. List of Test Scenarios.

iii. Test Case Table (with IDs, inputs, steps, and expected outputs).

iv. Test Execution Summary.

Task 3 Description - Testing and Debugging Statistical Utility Functions

You are provided with a Python module `stats_utils.py` that implements several statistical functions:

- `mean(numbers)` – calculates the average of a list.
- `median(numbers)` – calculates the median.
- `mode(numbers)` – finds the most frequent value.
- `range_list(numbers)` – calculates the range (difference between max and min).
- `remove_outliers(numbers, threshold=2)` – removes outliers based on a threshold.

Your Objectives

1. **Write Unit Tests:**
   - Create a test suite using `unittest` (or `pytest`) to verify the correctness of each function.
   - Include normal cases, edge cases, and error cases:
     - Empty lists
     - Lists with one element
     - Lists with even and odd lengths
     - Lists with multiple modes
     - Lists with extreme values (for outlier removal)
2. **Run Tests and Record Outputs:**
   - Execute your tests and capture the results.
   - Note which tests fail and why.
3. **Analyze and Report Bugs:**
   - For each failing test, explain:
     - What the expected output was.
     - What the actual output was.
     - Why the function produced incorrect results.
   - Identify the bug in the code and suggest a fix.

Hints

- Use methods like `assertEqual()` for comparing expected and actual results.
- Consider edge cases such as:
  - `mean([])` should return `0`.
  - `median([1, 2, 3, 4])` should return `2.5`.
  - `mode([1, 1, 2, 3, 3])` should handle multiple modes correctly.
  - `range_list([10, 2, 7, 5])` should return `8` (max - min).
  - `remove_outliers([1, 2, 3, 100])` should remove `100`.

Deliverables

i. A Python file with your unit tests.

ii. A document/report that includes:

- Test cases and their outputs.
- A list of bugs found in the original code.
- Suggested corrections for each bug.

Submission Format: PDF or DOCX document titled: `SE302_Homework01_<StudentID>.pdf` and Python file named: `test_stats_utils.py`

Important Notes: Ensure your answers are detailed and can be easily understood by each stakeholder. **Plagiarism** will not be tolerated. Your assignment should be original work! **Good luck with your assignment!**