**1. Interface-Based Approach**

This approach is based on the method signature and focuses on the types and constraints of input parameters.

**Characteristics**

* **List Size**: The number of elements in list1 and list2
  + Small (0–2 elements)
  + Medium (3–10 elements)
  + Large (more than 10 elements)
* **String Composition**: The structure of strings in the lists
  + Only lowercase letters
  + Only uppercase letters
  + Mixed case letters
  + Strings with non-alphabetic characters (e.g., numbers, symbols)
  + Empty strings
* **List Content**: The relationships between the elements
  + Both lists contain anagrams
  + No anagrams between lists
  + One list is empty
  + Both lists are empty
  + Lists contain duplicate elements

**2. Functionality-Based Approach**

This approach focuses on how the function behaves based on logical scenarios.

**Characteristics**

* **Anagram Presence**:
  + No anagrams present
  + Some anagrams present
  + All elements are anagrams
* **Case Sensitivity Handling**:
  + Anagrams differ only in case
  + Anagrams have mixed-case versions
* **Duplicates in Input Lists**:
  + No duplicates
  + Duplicates exist in one list
  + Duplicates exist across both lists

**(a) Are the partitions disjoint?**

No, some characteristics overlap. For example, "list contains anagrams" and "lists contain duplicates" are not mutually exclusive.

**Adjusted Partitioning:**

1. **List Size:** {empty, small (1-2 elements), medium (3-10 elements), large (>10 elements)}
2. **String Type:** {lowercase, uppercase, mixed case, special characters, empty strings}
3. **Anagram Groups:** {no anagrams, some anagrams, all elements are anagrams}
4. **Duplicates:** {no duplicates, duplicates in one list, duplicates in both lists}

Now, each partition is disjoint.

**(b) Does the partitioning cover the entire domain?**

Not fully, since we didn’t explicitly account for:

* Lists containing only one-word anagrams
* Lists that contain completely different lengths of words

**Fixed Partitioning:**

1. **List Size**: {empty, small, medium, large}
2. **String Composition**: {lowercase, uppercase, mixed case, special characters, empty}
3. **Anagram Presence**: {none, some, all}
4. **Duplicates**: {none, in one list, in both lists}
5. **Word Lengths**: {same length, different lengths}

We select a **base test** and generate test cases by varying one factor at a time.

**Base Test Choice**

* list1 = ["listen", "hello", "enlist"]
* list2 = ["silent", "world", "tinsel"]
* Expected Output: ["listen", "enlist", "silent", "tinsel"]

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Test ID | List Size | String Composition | Anagram Presence | Duplicates | Word Lengths | Expected Output |
| T1 (Base) | Medium | Lowercase | Some | None | Same | ["listen", "enlist", "silent", "tinsel"] |
| T2 | Empty | Lowercase | None | None | Same | [] |
| T3 | Small | Uppercase | Some | None | Same | ["LISTEN", "ENLIST", "SILENT", "TINSEL"] |
| T4 | Large | Mixed Case | Some | None | Same | Varies |
| T5 | Medium | Special Characters | None | None | Same | [] |
| T6 | Medium | Lowercase | All | None | Same | Entire list |
| T7 | Medium | Lowercase | Some | In one list | Same | Depends on duplicates |
| T8 | Medium | Lowercase | Some | In both lists | Same | Depends on duplicates |
| T9 | Medium | Lowercase | Some | None | Different | [] (if lengths differ) |

Total **9 tests**.