1. Reverse a ArrayList

Problem: Given an ArrayList

a. Reverse it using a stack.

- **Explanation:** Push all nodes onto a stack, then pop them one by one to rebuild the linked list in reverse order.

b. Without using stack

---

2. Remove Duplicates from an ArrayList while Maintaining Order

ArrayList Given:

```
[1, 2, 2, 8, 9, 4, 8, 4, 1, 5, 5, 7]
```

Expected Result:

```
[1, 2, 8, 9, 4, 5, 7]
```

- **(a) Without using HashSet:**
  Traverse the list and add elements to a new list only if they are not already present.
- **(b) With using HashSet:**
  Use a `LinkedHashSet` (to preserve insertion order) to eliminate duplicates easily.

---

3. Check if a Word has Only Unique Characters

- **Problem:** Determine whether all characters in a given word are unique.

Examples:

- Word: `"Code"` → **Result: true**
- Word: `"CodeQuotient"` → **Result: false**
- **Explanation:** Use either a boolean array/HashSet to track seen characters. If a character repeats, return false.

---

4. Create a Frequency Map of Characters in a Word

- **Problem:** Given a word, create a frequency map (i.e., count how many times each character occurs).

Example 1:
Word: `"success"`
Result:

```
s : 3
u : 1
c : 2
e : 1
```

Example 2:
Word: `"CodeQuotient"`
Result:

```
C : 1
o : 2
d : 1
e : 2
Q : 1
u : 1
t : 2
i : 1
n : 1
```

- **Explanation:** Traverse each character of the word and store its count in a frequency map (using a `HashMap<Character, Integer>`).

---

5. Check if Two Strings are Anagrams

- **Problem:** Given two strings, check if one is an anagram of the other (i.e., both strings contain the same characters with the same frequency, but possibly in different order).

Example 1:
Input: `"listen"`, `"silent"`
Result: **true**

Example 2:
Input: `"triangle"`, `"integral"`
Result: **true**

Example 3:
Input: `"hello"`, `"world"`
Result: **false**

- **Explanation:** Use a frequency map (or sorting) to compare character counts of both strings.

---

6. Find the First Non-Repeating Character in a String

- **Problem:** Given a string, find the first character that does not repeat.

Example 1:
Input: `"swiss"`
Result: `"w"`

Example 2:
Input: `"programming"`
Result: `"p"`

Example 3:
Input: `"aabbcc"`
Result: `"No non-repeating character found"`

- **Explanation:** Use a frequency map to count occurrences, then scan the string again to find the first character with frequency 1.