## Section 1: Basic Implementation (All Maps)

1. **Create a program to store and display student roll numbers with names using a `HashMap`.**
   - Add at least 5 entries.
   - Print all entries using entrySet.
2. **Use a `LinkedHashMap` to store product IDs and product names.**
   - Add items in random order and observe the output order.
   - Add duplicate keys and observe the result.
3. **Create a program to store login credentials (username and password) using `Hashtable`.**
   - Add null values and observe behavior.
   - Try with null key.

---

## Section 2: Functional & Practical Use Cases

5. **Count frequency of characters in a string using `HashMap`.**
   - Input: `"JavaCollections"`
   - Output: `{J=1, a=2, v=1, ...}`
6. **Maintain insertion order of users and their roles using `LinkedHashMap`.**
   - Add entries like: `("admin", "Alice"), ("editor", "Bob")...`
   - Print the values in insertion order.
7. **Use `Hashtable` to simulate a basic phonebook application (Name -> Phone number).**
   - Perform add, search, and delete operations.
   - Disallow null keys and values.
8. **Create a `TreeMap` that stores employee IDs and names.**
   - Print the map sorted by ID.
   - Retrieve the employee with the **lowest and highest** ID.

---

## Section 3: Comparison & Behavior

9. **Create and compare output of HashMap, LinkedHashMap, TreeMap, and Hashtable** using the same input:
   - Add: `{"C", "Cat"}, {"B", "Bat"}, {"A", "Ant"}`
   - Print all maps and observe ordering and behavior with null keys/values.
10. **Write a method to copy all entries from one map (say HashMap) to another map (say TreeMap).**

- Then sort the copied data using `TreeMap`.

---

## Section 4: Advanced & Logical Tasks

11. **Store employee names and their salaries in a `HashMap`.**

- Find the employee(s) with the highest salary.

12. **Using a TreeMap, create a dictionary that stores words and their meanings.**

- Display all words in alphabetical order.

13. **Group a list of strings by their first character using `HashMap<Character, List<String>>`.**

- Input: `["apple", "banana", "apricot", "blueberry"]`
- Output: `{a=[apple, apricot], b=[banana, blueberry]}`

14. **Create a voting system where candidate names are keys and vote count is the value.**

- Use `HashMap`.
- Input multiple votes and display the winner.