Documentation for Implement rand10() Using rand7()

Table of Contents

1. Problem Statement

Given an API rand7() that returns a uniform random integer from 1 to 7, implement a function rand10() that returns a uniform random integer from 1 to 10 using only rand7().

Constraints:

- Only the rand7() API can be used.
- You should not use any built-in random functions.
- The implementation must ensure uniform distribution for numbers 1 through 10.

2. Intuition

Since rand7() gives 7 outcomes uniformly, and 10 is not a multiple of 7, we must find a way to map the output of rand7() to 1–10 uniformly.
To do that, we can combine multiple calls to rand7() to generate a larger uniform space and selectively map a subset of it to 1–10.

3. Key Observations

- rand7() × rand7() can create 49 unique pairs (i.e., a uniform distribution over 1–49).
- The numbers 1 to 40 from this set can be evenly divided into 10 groups (i.e., 40 is divisible by 10).
- Discard the remaining 9 values (41–49) and retry to preserve uniformity.

4. Approach

- Call rand7() twice to simulate a uniform number from 1 to 49.
- If the result is ≤ 40, use (number - 1) % 10 + 1 to map it to 1–10.
- If the number is > 40, retry (loop) to ensure fairness.
- This method guarantees uniformity in the 1–10 range.

5. Edge Cases

- Infinite loop? No, since we retry only when the number is > 40, and the retry rate is low.
- Bias? No, numbers 1–40 are evenly mapped to 1–10.

6. Complexity Analysis

Time Complexity

- Average Case:

  Probability of valid number ≤ 40 = 40/49

  Expected number of tries = 49 / 40 ≈ 1.225

  Each try uses 2 calls → Expected rand7() calls: ~2.45`
- Worst Case: Infinite loop theoretically possible (very low probability), but practically rare.

Space Complexity

- O(1) — No extra space is used beyond constants.

7. Alternative Approaches

Approach	Pros	Cons
Generate 1–49 and map 1–40	Uniform, efficient	Retry needed for 41–49
Generate 1–63 using 3×rand7()	Slightly less retry rate	More calls to rand7() per try
Precomputed table (hardcoding)	Fast lookup	High space complexity, not scalable

8. Test Cases

Input (n calls)	Expected Output Example	Notes
n = 1	[5]	Any integer 1–10
n = 5	[1, 10, 3, 6, 8]	All numbers should be 1–10
n = 10000	Uniform distribution	Run frequency test if needed

9. Final Thoughts

- This method leverages probability and uniform mapping efficiently.
- With only ~2.45 calls to rand7() per rand10(), it balances simplicity and performance.
- In interviews, this shows a good grasp of probability, randomness, and optimization.