📘 NOTES on "Remove Digits" (DP + Recursion + Memoization)

✅ Problem Summary

Tumhe ek number **n** diya hota hai.

Har step me tum **n** ka koi ek non-zero digit (jaise 1,2,3,4,5...9) subtract kar sakte ho.

Goal:

👉 **Minimum steps** find karo to make the number **0**.

Example:

n = 27

Digits: 2,7

27 → 20 → 18 → 10 → 9 → 0 (5 steps)

🧩 Code Explanation

🔷 1. `getdigits(n)`

Ye function n ke andar se **non-zero digits** nikaal kar vector me daal deta hai.

Working:

- n % 10 se last digit milti hai.
- Agar digit 0 nahi ho → vector me push.
- Fir `n = n/10` karke next digit.

Example:

n = 4057

Digits → {7,5,4}

🔷 2. Memoization Array `memo`

`memo[n]` store karega:

👉 **minimum steps required to make n → 0**

Initial values → `-1` means not calculated yet.

🔷 3. `f(n)` — Main Recursion + DP

Base Case:

```
if(n == 0) return 0;
```

0 ko 0 banane ke zero operations chahiye.

Memoization Check:

```
if(memo[n] != -1) return memo[n];
```

Agar answer pehle se calculated hai → directly return.

Main Logic:

```
vector<int> dp = getdigits(n);
int result = INT_MAX;

for each digit d in dp:
    result = min(result, 1 + f(n - d))
```

Har digit subtract karke dekho

→ jo bhi minimum steps deta hai wahi answer.

Finally store:

```
return memo[n] = result;
```

🔷 4. `main()`

- n input lete hain.
- Memo (size n+1) ko -1 se initialize.
- f(n) call karke print.

🧠 Time Complexity

- Har number 1 se n tak sirf **once** calculate hota hai.
- Har step me approx 1–9 digits check.

👉 **O(n × digits)** ≈ O(9n) → **O(n)**

(kaafi fast hai)

📌 Final Notes

✔️ Problem digit DP + memoization ka classic example hai

✔️ Har step me number ka non-zero digit subtract hota hai

✔️ Memoization guarantee karta hai ki har state ek hi baar solve hogi

---