☑ Level 60 — insert() & erase() (vector ka real pain area)

1 insert() — adding in the middle:

```
v.insert(position_iterator, value);
```

Example:

```
vector<int> v = {1, 2, 4};
v.insert(v.begin() + 2, 3);
Result: 1 2 3 4.
```

What ACTUALLY happens internally:

- Elements after index 2 are shifted right.
- Possible reallocation.
- Time complexity: O(n) ❌.

👉 This is NOT cheap. Remember that.

2 erase() — deleting elements: v.erase(position_iterator);

Example:

```
vector<int> v = {1, 2, 3, 4};
v.erase(v.begin() + 1);
Result: 1 3 4.
```

Internal reality: Elements after erased one are shifted left.

Time complexity: O(n) ❌.

3 Erase a RANGE:

```
vector<int> v = {1,2,3,4,5};
v.erase(v.begin() + 1, v.begin() + 4);
Removes: 2 3 4.
Remaining: 1 5.
```

👉 Second iterator is NOT included

4 Biggest beginner mistake (DON'T DO THIS):

❌ Wrong:
```
for(int i = 0; i < v.size(); i++) {
    if(v[i] % 2 == 0)
        v.erase(v.begin() + i);
    }
```

Why wrong? erase() shifts elements. Index gets messed up. Elements get skipped. This bug is VERY common.

5 Correct way (iterator-safe erase):

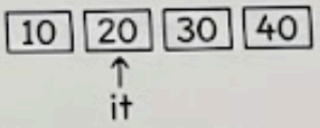
```
for(auto it = v.begin(); it != v.end(); ) {
    if(*it % 2 == 0) it = v.erase(it);
    else it++;
}
```

This is DSA-grade correct code.

Situation setup:

Index:	0	1	2	3
Value:	10	20	30	40

it points to 20.



Step 1: What does erase(it) do? v.erase(it);

✅ 1. Deletes element. Vector becomes: 10 30 40.

✅ 2. RETURNS iterator pointing to next valid element.

Step 2: After deletion: 10 30 40 ↑ 30 shifted left.

erase(it) → iterator pointing to 30.

Step 3: Why it = erase(it)? Old it invalid. Update it.

it = v.erase(it) means: Delete current, move it to next safe.

v1 = {1, 2, 3, 4, 5}
↑ it

① *it = 1 , it is address

1%2 == 0 ✗

it ++ it

② *it=2 v1 = {1, 2, 3, 4, 5}

2%2 == 0 ✓

erase (2)

v1 = {1, 3, 4, 5}

↑ it

it has same address

③ no, it ++

*it = 3 v1 = {1, 3, 4, 5}

↓ it

3%2==0 ✗

Else it ++

④ *it = 4, v1 = {1, 3, 4, 5}

↓ it

4%2==0 ✓

erase (4)

it has same address

v1 = {1, 3, 5}

↓ it

no, it ++

```
if (*it %2 == 0) {
    it = v.erase(it);
}
```

⑤ *it = 5, v1 = {1, 3, 5}

↓ it

5%2==0 ✗

Else it ++

now, it = v1.end()