Analizziamo questo punto formalmente. L'espressione del for loop:

```
for(auto it = lst.begin(); it != lst.end(); ++it)
```

è semanticamente equivalente a:

```
{
    auto it = lst.begin();
    while(it != lst.end()) {
        // corpo del ciclo
        ++it;  // <-- questo avviene SEMPRE alla fine di ogni iterazione
    }
}
```

Questo significa che se nel corpo del ciclo fai:

```
for(auto it = lst.begin(); it != lst.end(); ++it) {
    if(condition) {
        it = lst.erase(it);  // erase() restituisce iteratore alla posizione
successiva
        // ... qui il ciclo fa comunque ++it alla fine!
    }
}
```

Dopo erase(), l'iteratore it punta già al prossimo elemento. Quando il ciclo for esegue il suo ++it di default, salti effettivamente un elemento!

Soluzioni corrette:

1. Usando il for senza incremento esplicito:

```
for(auto it = lst.begin(); it != lst.end(); ) {  // Nota: niente ++it qui
    if(condition) {
        it = lst.erase(it);  // erase() fa già l'incremento
    } else {
        ++it;  // incremento manuale solo se non c'è erase
    }
}
```

2. Usando il while (più chiaro semanticamente):

```
auto it = lst.begin();
while(it != lst.end()) {
```

```
    if(condition) {
        it = lst.erase(it);
    } else {
        ++it;
    }
}
```

Le soluzioni che hai visto probabilmente contengono un bug sottile: stanno saltando elementi dopo ogni erase(). Il problema può non manifestarsi in tutti i casi d'uso, ma è formalmente scorretto.