(Q) Print name n times using recursion

let n = 3

```
void f(i, n) {

    if (i > n) return;      base    ✗

    ✓ cout << "Sejal";
    ✓ f(i+1, n);
}

int main() {
    int n;
    cin >> n;
    f(1, n);
}
```

TC : O(n)
SC : O(n)

```
f(2,3) {
    if (2 > 3) return;    ✗
    cout << "Sejal";  ✓
    f(i+1, n);   ✓
}
```

```
f(3,3) {
    if (3 > 3) return;  ✗
    cout << "Sejal";  ✓
    f(i+1, n);   ✓
}
```

```
f(4,3) {
    if (4 > 3) return;  ✓
    cout << "Sejal";  ✗
    f(i+1, n);   ✗
}
```

Sejal
sejal
sejal

```
        f(1,3)
      ↗
   f(2,3)
  ↙   ↗
f(3,3)
  ↙ ↗
f(4,3) —— base
```

| f(3,3) |
| f(2,3) | stack
| f(1,3) | space
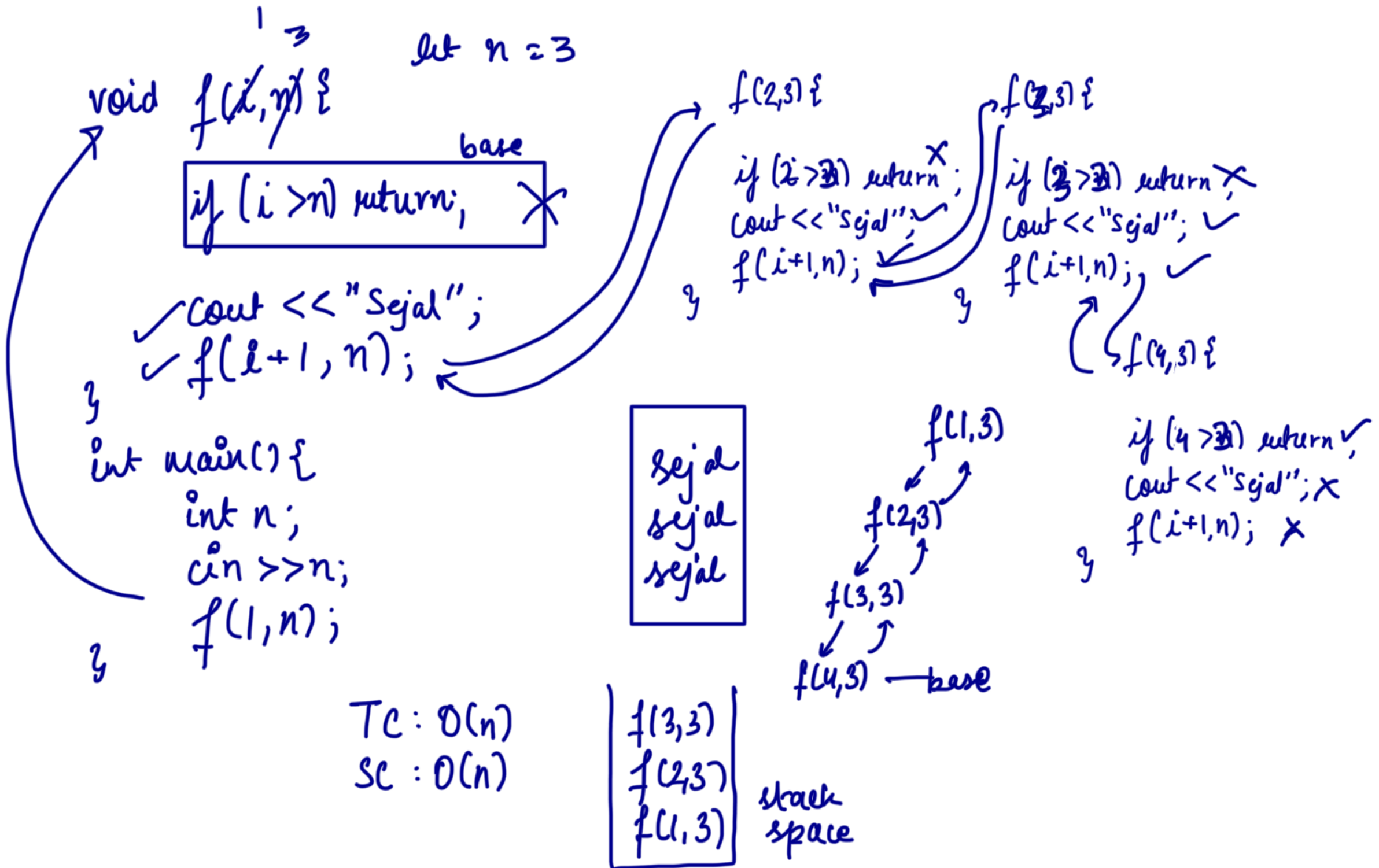
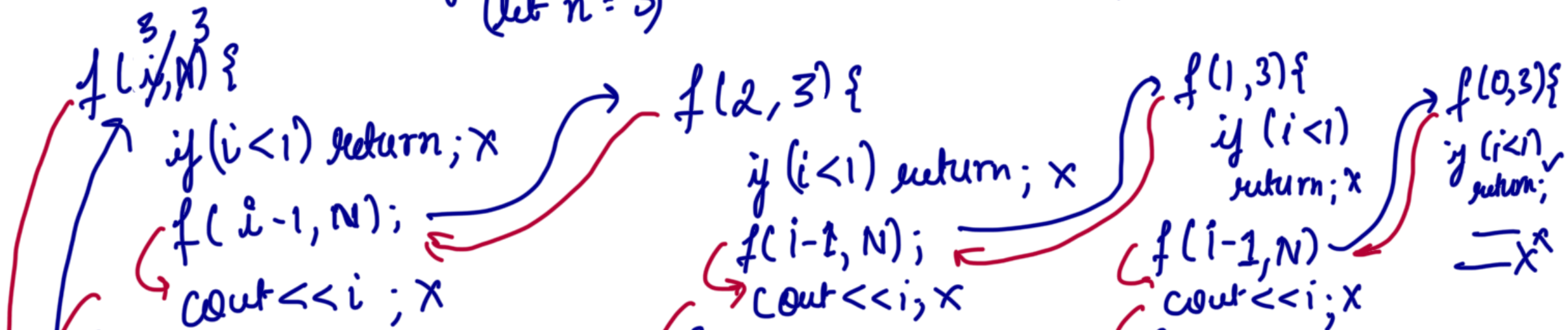**(Q2) print unary from (1→N)**

```
f(i, N) {
    if (i > N) return;
    cout << i;
    f(i+1, N);
}
int main() {
    int N;
    cin >> N;
    f(1, N);
}
```

(Q3) print unary from (N→1)

```
f(i, n) {
    if (i < 1) return;
    cout << i;
    f(i-1, n);
}
int main() {
    int n;
    cin >> n;
    f(n, n);
}
```

(Q4) 1 to N (by backtracking) → don't use $f(i+1, n)$
(let n = 3)

```
f(i, N) {    3,3
    if (i < 1) return; x
    f(i-1, N);
    cout << i; x
```

```
f(2, 3) {
    if (i < 1) return; x
    f(i-1, N);
    cout << i; x
```

```
f(1, 3) {
    if (i < 1)
        return; x
    f(i-1, N)
    cout << i; x
```

```
f(0, 3) {
    if (i < 1)
        return;
        x
```

```
int main() {
    cin >> n;
    f(n, n);
}
```

Output:

1
2
3

This happen when you write print after function call

(Q5) print N to 1 by back tracking   [f(i-1, N) x]
(let n = 3)

```
            1  2  3
f(i, n) {
    if (i > n) return; x
    f(i+1, n); ✓
    cout << i; x)
}

int main() {
    int n;
    cin >> n;
    f(1, n)
}
```

```
f(2, 3) {
    if (i > n) return; x
    f(i+1, n); ✓
    cout << i; x
}          3
           2
```

```
f(3, 3) {
    if (i > n) return; x
    f(i+1, n); ✓
    cout << i; x
}          3
```

```
f(4, 3) {
    if (i > n)
        return; ✓
    —x
    —x
}
```

Output	
	3
	2
	1