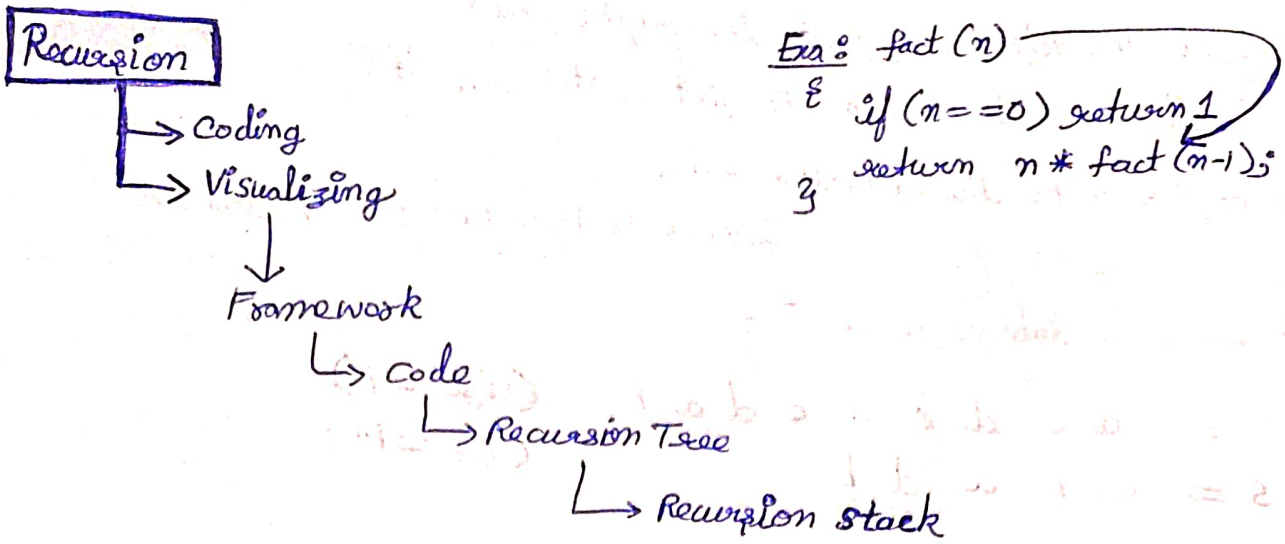
RECURSION FOUNDATIONS



$Fib(n) = Fib(n-1) + Fib(n-2)$

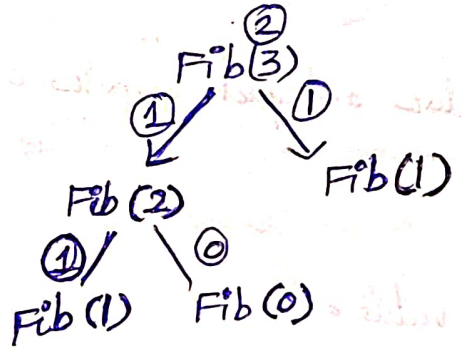
$n \leq 1$, $Fib(n) = n$

① Code

```
Fib(n)
[ if (n<=1) return n;
  return Fib(n-1) + Fib(n-2) ]
```

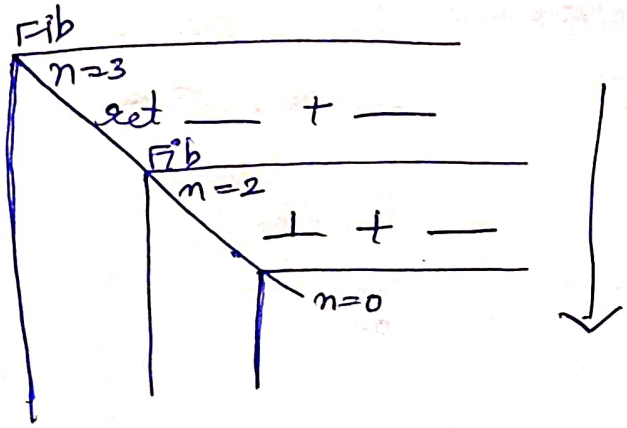
② Recursion Tree — [Functioncalls → Name (Parameter)
Return Value]

Eg:-



← Call graph

③ Recursion Stack



Exactly how functions calls are executed in C++.

Which is hard?

① Write new → Harder

② Understand Written → With practice can be learned

When you are dealing with recursion there are only certain types of code :

① Encode a logic
② Use recursion to [do, recurse, comeback] → 4 Forms of Recursion problem
③ K^th Finding
④ Fractals

① Encode logic

Q. Check if Palindrome using recursion ?

s = 'madam'

① Pruning → check for valid parameters value
② Base Case
③ Calculate

```
isPalindrome (first, last) {
    if (first ≥ last) return 1; // Base case
    //calculate
    if (s[first] != s[last]) return 0;
    return isPalindrome (first+1, last-1);
}
```

Q. You are given of n & r. Calculate (^nCr) recursively

Relation : $^nC_r = {}^{n-1}C_r + {}^{n-1}C_{r-1}$

Pseudo code :-

```
int nCr (int n, int r) {
    //Pruning
    if (r<0 || r>n) return 0;
    //Base Case
    if (n==0) return 1;
    //calculate
    return nCr (n-1, r-1) + nCr (n-1, r);
}
```

Q. Write the recursive code for general n ?

DO, Recurse, Comeback Pattern

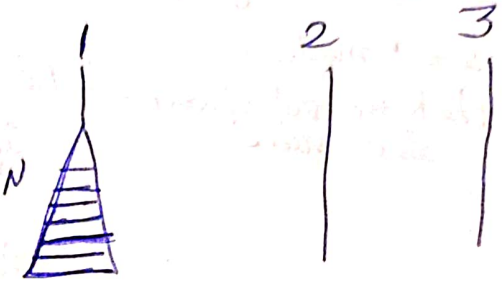
1. Do → Print the current no N.
2. ~~Recurse~~ Recurse → call the func^n for N-1
3. Comeback → After the recursive call returns print N again.

```
4   N
3   N-1
2   .
1   2
2   1
3   2
4   .
    .
    N-1
    N
```

Pseudocode :-

```
printer (int N) {
    if (n==1) cout<<1<<endl; //Base case
    return;
    cout<< N <<endl;
    printer (N-1);
    cout << N << endl;
}
```

Q. Tower of Hanoi



move (N, from, to, extra)
move (N-1, from, extra, to)
N disk → from ⟶ to
move (N-1, extra, to, from)

Eg: N=3

Code :-

```
void movedisk(int disks, int from, int to,
              int extra) {
    if (disks == 0) return; //Base case
    cout << "from" << from << "To" << to << endl;
    movedisk(disks-1, extra, to, from);
}
```

1 → 3
1 → 2 (Do, Recurse comeback in no order)
3 → 2
1 → 3
2 → 1
2 → 3
1 → 3

Recursion tree :-