Week 7
Lec 1 —

What is Recursion? — Technique to solve a question.

Bookish → when a func^n calls itself

Desi

Bigger Problem soln → depends on ↓ Choti & same type ki problem pr

Problem statement

↓

Relation / formula

Bigger Problem → Choti Problem

(same type

Ex 2) **Fibonacci series**

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

Ex 3) Bigger Problem → 2 to power of n → 2^n

$$2^n = \underbrace{2 \times 2 \times 2 \times \ldots \ldots 2}_{2 \rightarrow n \text{ times}}$$

$2 \times 2 = 2^2$

$2^2 \times 2^3 = 2^5$

$2^5 \times 2^{17} = 2^{22}$

$2^n \rightarrow f(n)$

$$2^n = 2 \times 2^{n-1}$$

Bigger Problem — Choti Problem

$$f(n) = 2 \times f(n-1)$$

Ex 4) Problem statement → i/p – n, o/p – n!

$$n! = n \times (n-1)!$$

Bigger Problem — Choti Problem

$$f(n) = n \times f((n-1)!)$$

Ex5) Counting Reverse

Bigger Problem → f(n) → reverse counting from n to 1

f(n) → print(n) + f(n-1)

B.P — c.P

↓

f(5) → print(5), then f(4)

5 + f(4)

4 + f(3)

3 + f(2)

2 + f(1)

1 + f(0)

---

1. Code for factorial :-

```
int factorial (int n){
    int ans = n * factorial (n-1);
    return ans;
}
```

will not stop!

Recursive Code:-

- Base Case
- Recursive relation / call [Formula]
- Processing — (optional)

Code:-

```
int factorial (int n){
    //base case
    if (n==1)
        return 1;

    int ans = n * factorial(n-1);
                      choti problem

    return ans;
}
```

Counting Code →

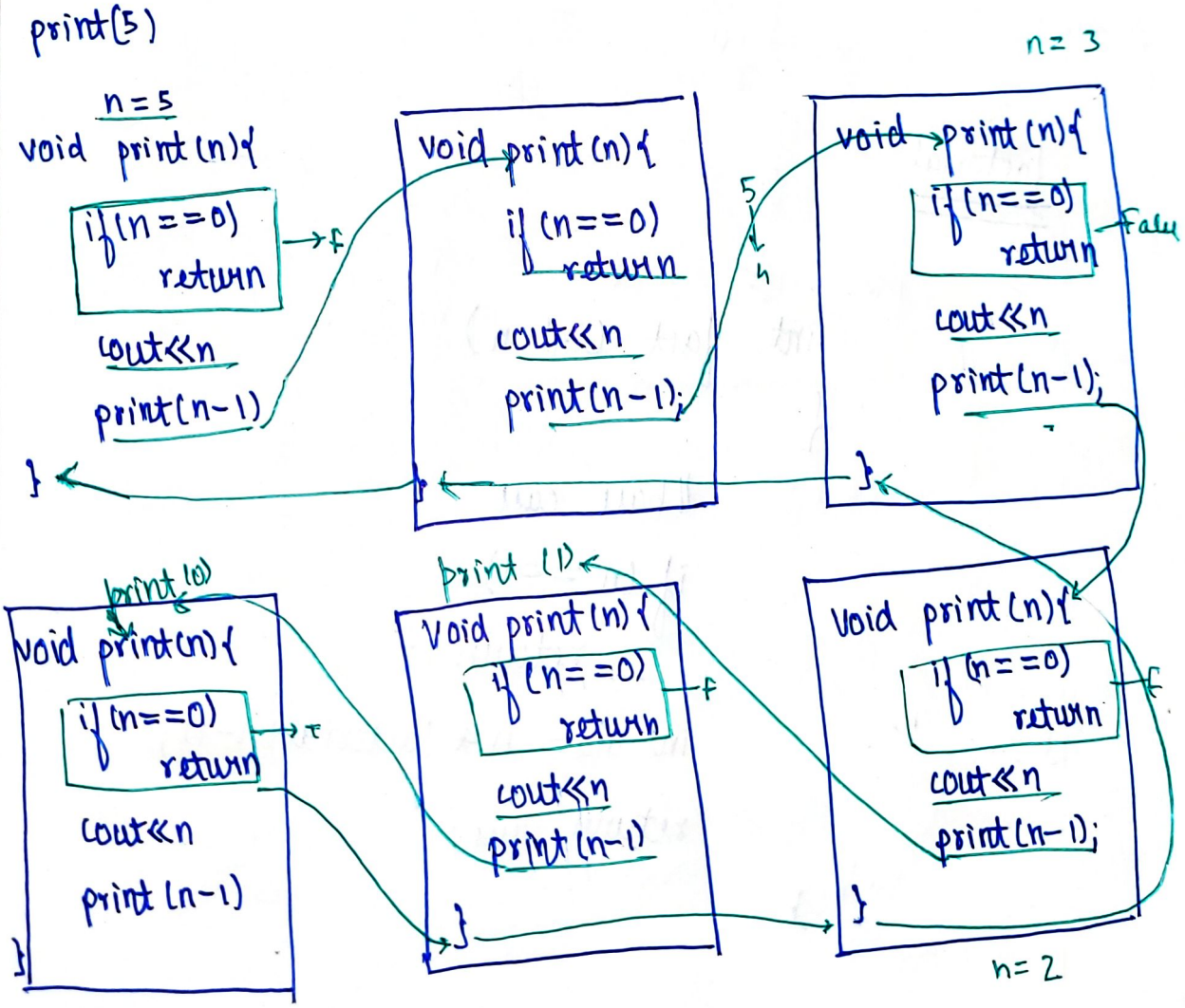
```
void printCounting (int n){
    // base case
    if (n==0){
        return;
    }
    // processing
    cout << n << endl;

    //recursive rel^n
    printCounting(n-1);
}
```

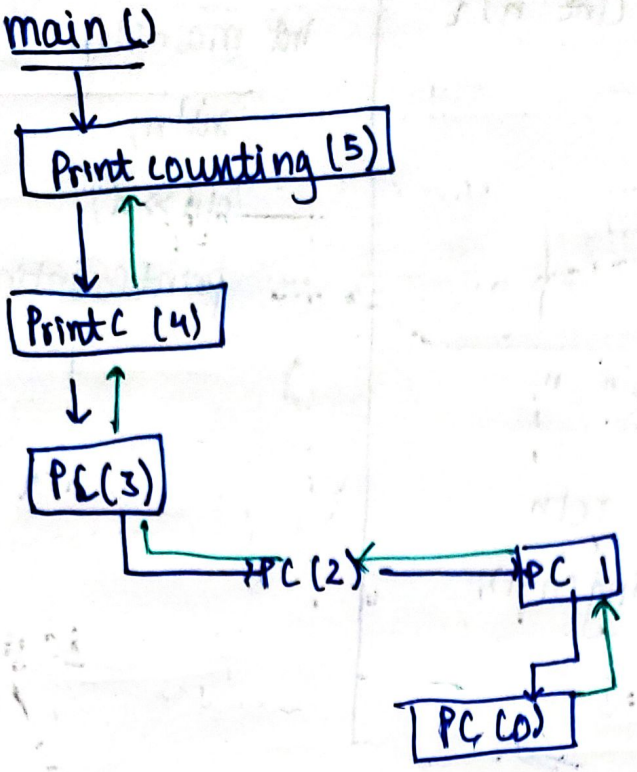
```
int main(){
    int n;
    cin>>n;
    printCounting(n);
}
```

```
void printCounting(int n){
    //base case
    if(n==0)
        return;

    //processing
    cout<<n<<" ";

    //recursive reln
    printCounting(n-1);
}
```

```
int main(){
    int n;
    cin>>n;
    printCounting(n);
}
```

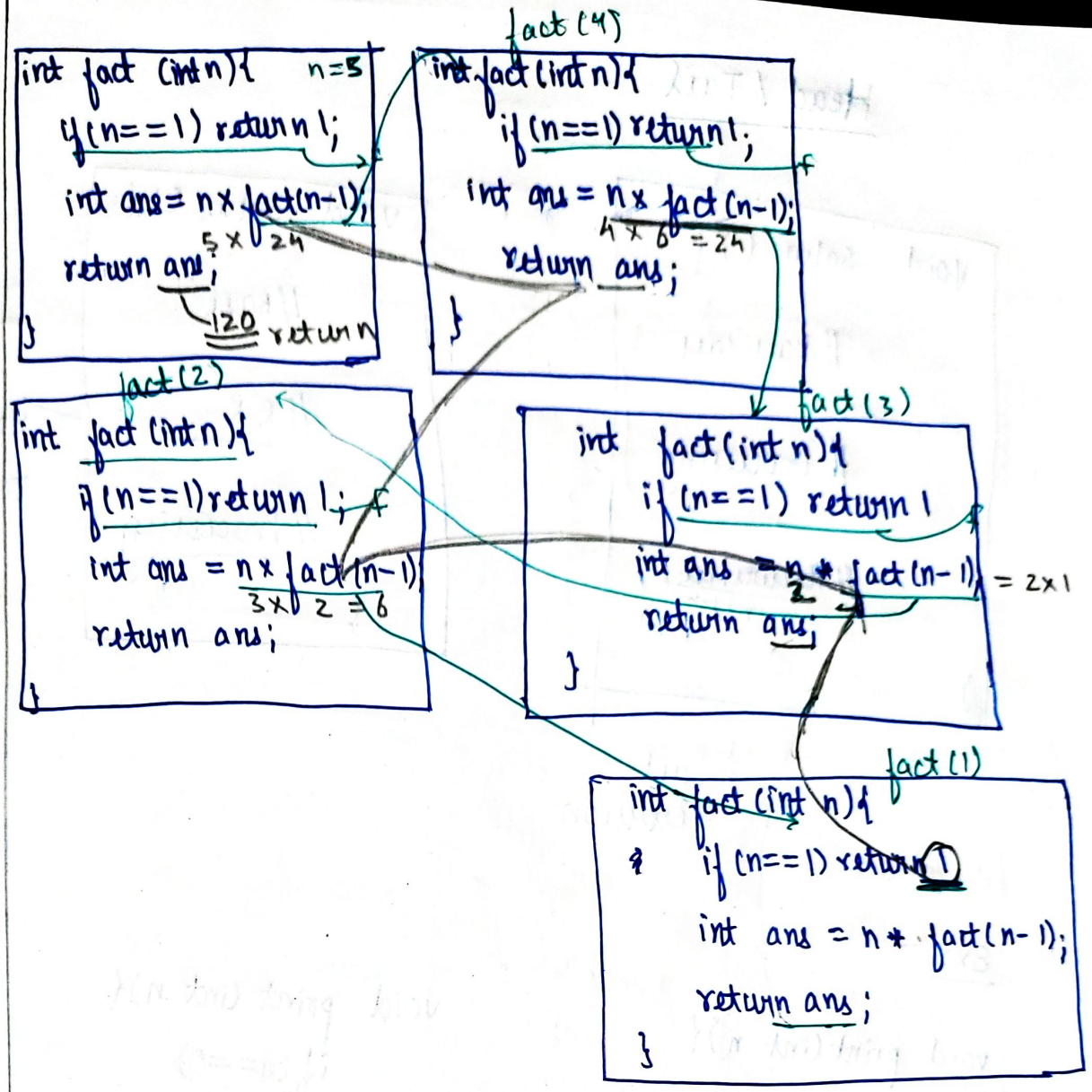
print(5)

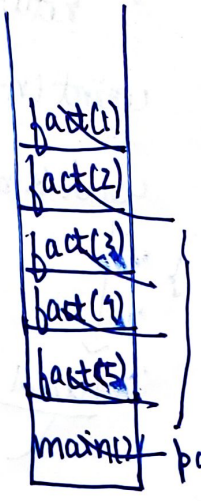




factorial:

```
int fact (int n)
{
    // base case
    if (n == 1)
        return 1;
    int ans = n * factorial(n-1);
    return ans;
}
```


int fact (int n){ n=5
if(n==1) return 1;
int ans = n * fact(n-1);
5 x 24
return ans;
120 return
}
fact(4)
int fact(int n){
if(n==1) return 1;
int ans = n * fact(n-1);
4 x 6 = 24
return ans;
}
fact(2)
int fact (int n){
if(n==1)return 1;
int ans = n * fact(n-1)
3x 2 = 6
return ans;
}
fact(3)
int fact(int n){
if(n==1) return 1
int ans = n * fact(n-1) = 2x1
2
return ans;
}
fact(1)
int fact (int n){
if(n==1) return 1
int ans = n * fact(n-1);
return ans;
}


fact(1)
fact(2)
fact(3)
fact(4)
fact(5)
main()
pop

Call stack

Head / Tail

```
Void solve (){
    // Base case
    // Processing
    // Recursive call
}
```

Tail recursion

```
void solve (){
    // Base
    // RR
    // Processing
}
```

Head recursion

Ex:

```
void print (int n){
    if (n==0)
        return;
    cout << n;
    print (n-1);
}
```

Tail → 5, 4, 3, 2, 1

```
void print (int n){
    if (n==0)
        return
    print (n-1);
    cout << n;
}
```

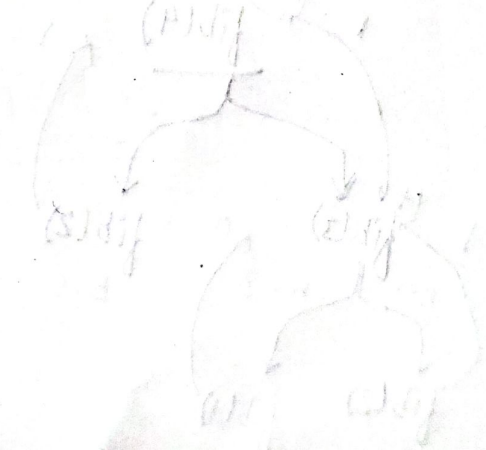
Head

1, 2, 3, 4, 5

Fibonacci series: -

```
int fib(int n){
    // base case
    if(n==1){
        //first term
        return 0;
    }
    if(n==2){
        //second term
        return 1;
    }
    //RR → f(n)= f(n-1) + f(n-2)
    int ans = fib(n-1) + fib(n-2);

    return ans;
}
```

```
int main(){
    cin >> n;
```



For n=5

```
int fib(n){
if(n==1) return 0;
if(n==2) return 1;
return fib(n-1) + fib(n-2);
}
```

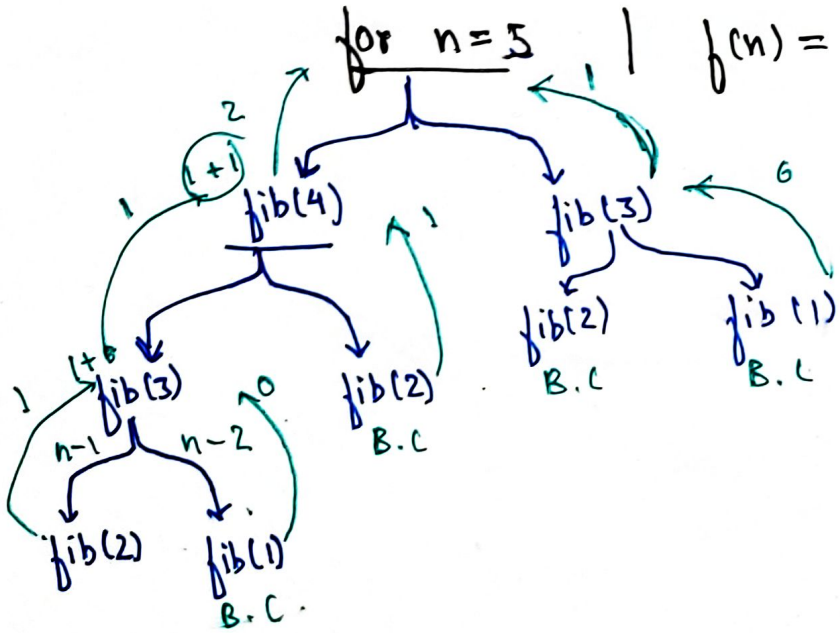
```
int fib(n){
if(n==1) return 0;
if(n==2) return 1;
return fib(n-1) + fib(n-2);
}
```

```
int fib(n){
if(n==1) return 0;
if(n==2) return 1;
return fib(n-1) + fib(n-2);
}
```

```
int fib(n);
{
if(n==1) return 0;
if(n==2) return 1;
return fib(n-1) + fib(n-2);
}
```

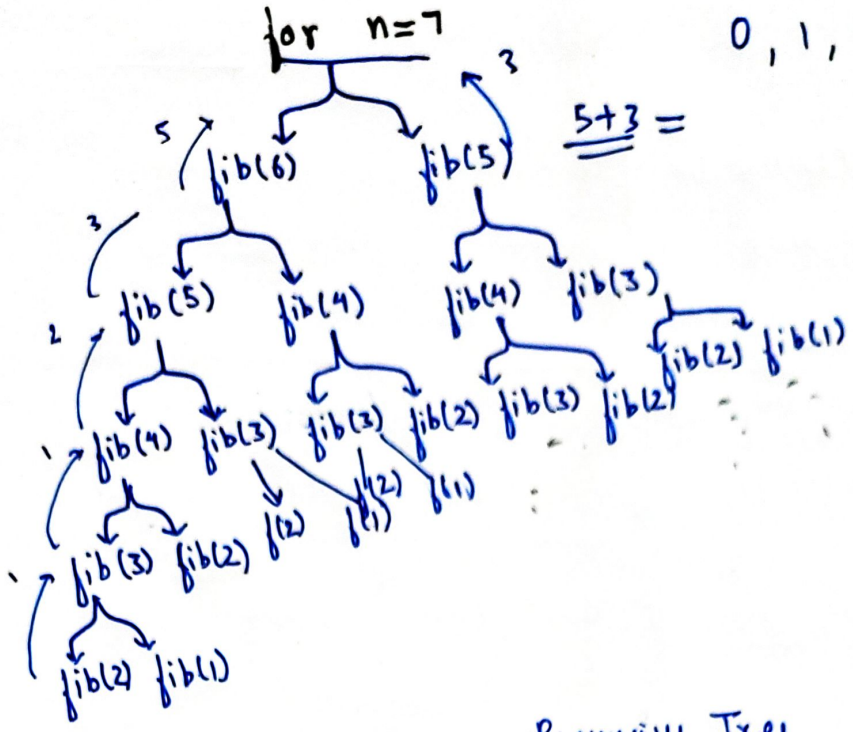
```
int fib(n){
if n==1 return 0
if n==2 return 1;
return fib(n-1)+
fibn-2);
```

```
int fib(n);
{
if(n==1) return 0;
if(n==2) return 1;
return fib(n-1)+fib(n-2);
}
```



fib(0) = 0

fib(2) = 1



0, 1, 1, 2, 3, 5, 8, 13 . . . .

5+3 =

Recursive Tree

Ratna

Magical line: Ek case solve kardo baaki recursion sambhal lega!

H.W → Todays class ques. 4-5 baar alg alg test case pe try

- Climb stair (Leetcode) ✓
- Array mei max^m no. , min^m no. — mat kro ✓
- Print all digits using recursion
- Array, key ka found key ka index