① Introduction
- Print Increasing
- Print Decreasing
- Print Increasing Decreasing
- Power - linear
- Power - logarithmic
- Power - zig zag
- Tower of Hanoi

② Recursion and arrays
-

Principle of Mathematical Induction (Eg $\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$

① Trivial case ≡ Base case

$\sum_{i=0}^{0} i = 0$, $\sum_{i=1}^{1} i = 1$ ✓

② assume formula is valid for k ✱

$$\sum_{i=1}^{k} i = \frac{k \times (k+1)}{2}$$

③ Prove for k+1

$$\sum_{i=1}^{k+1} i == \frac{(k+1)[(k+1)+1]}{2} \quad ? \text{ Prove.}$$

If we can prove, then formula is correct.

Similarly, we expect from recursive function to solve upto k and we solve for K+1

① Print Increasing

a) Expectation

void printInc (int n) :→ 1, 2, 3, ... n

b) Keep faith on recursion for smaller values.

void printInc (n-1) :- 1, 2 .. n-1 ✓✓

Assume it is correct, do not question the little one.

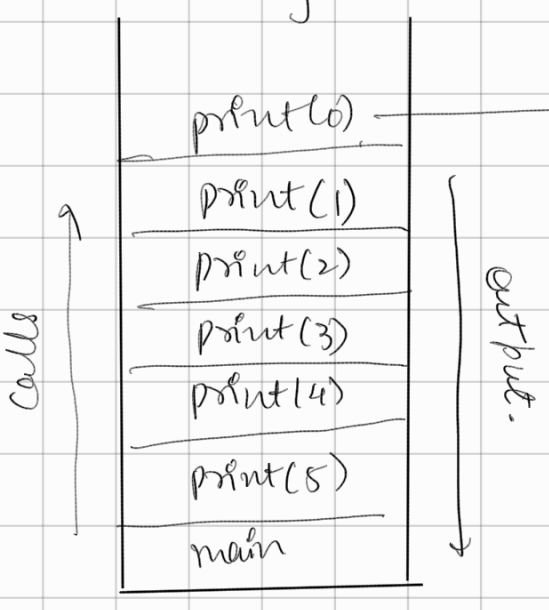
c) Mat expectations with faith

Print (n)

d) Base case. (Eg, if n=0, then don't print only return

CODE (POST ORDER)

```
void printInc (int n) {
    if (n==0) return;        // Base call
    printInc (n-1)           // Call for smaller problem
    Syso (n);                // Print current n
}
```

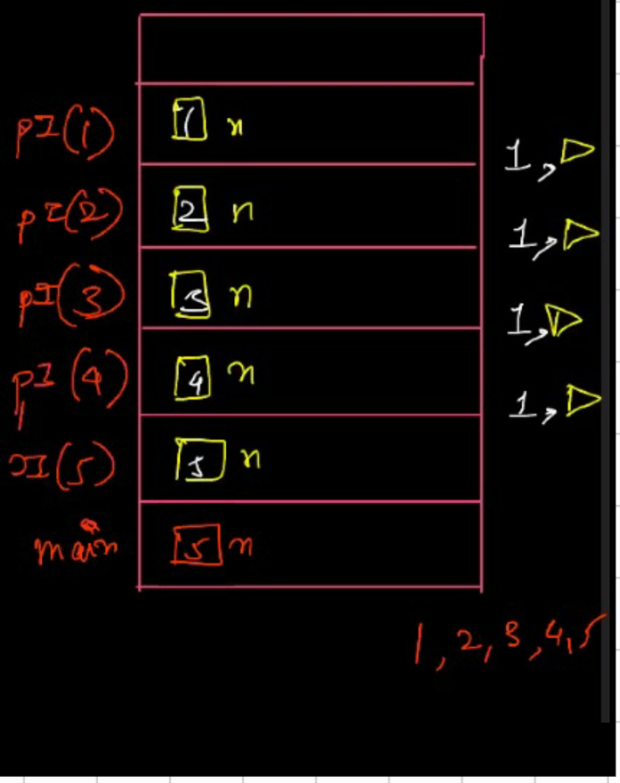


→ to stop here, we need a base case

base case: if (n==0)
return.

```
void printInc(int n) {
Base case → (a) if(n==0) return;
Faith → (1) printInc(n-1);
Meeting Expectation → (2) System.out.println(n);
}

main(){
    printInc(5);
}
```



(2) Print Decreasing

expectation

(a) printDecreasing(int n): n, n-1, ... 1

(b) printDec(n-1): n-2, n-2, n-3 ... 1 faith

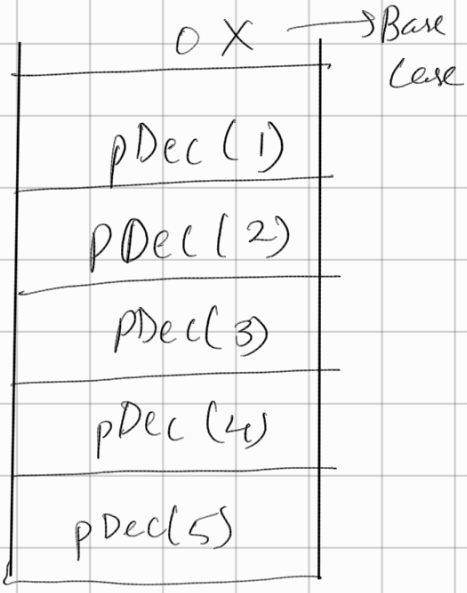
(c) mult expectation: print n

print first
then call n-1

Base case: if(n==0)
return

CODE (PRE ORDER)

```
if(n==0) return;
Syso(n);
printDec(n-1);
```



(3) Print Increasing Decreasing

(1) Expectation:

pdi (n) ⇒ n, n-1 ... 1 ... n-1, n.

(2) Faith:

pdi (n-1) ⇒ n-1 ... 1 ... n-1

(3) muting expectation

Syso (n) → preorder
Syso (n) → postorder

CODE:

```
if (n==0) return;
Syso(n);
pdi(n-1);
Syso(n);
}
```

(4) Factorial.

(a) Expectation: fac(n) → 1x 2x ... n
(b) faith : fac(n-1) → 1x 2x ... n-1
(c) work : return n x fac(n-1)

CODE:

```
int fact(int n){
    if (n<=1)
        return 1;
    return n*fact(n-1);
}
```

⑤ Power linear ($O(n)$)

ⓐ Expect : pow(x, n) ⟶ x^n

ⓑ faith : pow(x, n-1) ⇒ x^{n-1}

ⓒ work : x * pow(x, n-1);

CODE:

```
pow(x, n){
    if(n==0)
        return 1;
    return x * pow(x, n-1);
}
```

⑥ Power logarithmic ($O(\log_2 n)$)

```
pow(x, n){
    if (n==0) return 1;

    int ans = pow(x, n/2);
    ans *= ans;
    if(n%2==1)
        ans *= x;
    return ans;
}
```

① expect: pow(x, n) ⟶ x^n

② faith: pow(x, n/2) ⟶ $x^{n/2}$ Important

③ work: ans = pow(x, n/2); ans *= ans;
if (n%2==1) ans *= x;