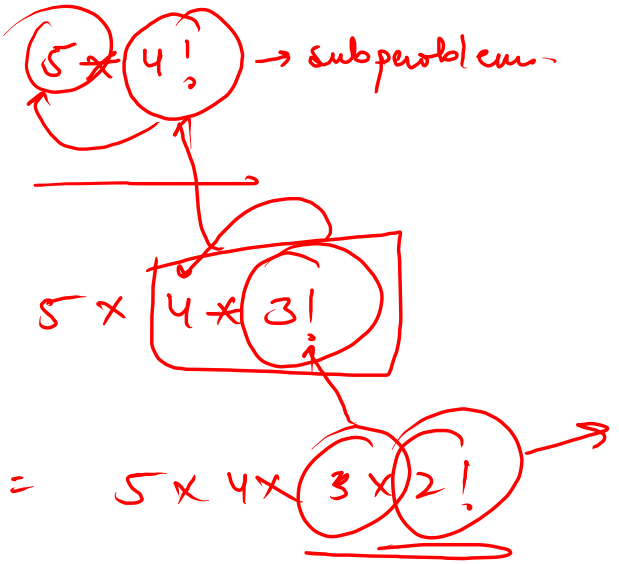
Recursion :-

↳ fn. calling itself ×

Solving problems using subproblems called recursion

factorial (5) = 5 * 4 x 3 x 2 x 1



PMI (Principle of Mathematical Induction)

$S_n = 1+2+3+4+ \quad ---- \quad +n = \frac{n(n+1)}{2}$

Step-01 $\Rightarrow n=1$

$\hookrightarrow S_1 = 1$

$\frac{1(1+1)}{2} = \frac{2}{2} = 1$

$n = k+1$

Assumption Step-02 Faith

$\Rightarrow S_k \Rightarrow$ Sum of first k natural nos. $= \frac{k(k+1)}{2}$ → 1

Step-03 $\Rightarrow S_{k+1} \Rightarrow$ Sum of first (k+1) natural nos.

$\Rightarrow 1+2+3+4+ \quad --- \quad k+(k+1) \Rightarrow \frac{n(n+1)}{2}$

$= \frac{k(k+1)}{2} + (k+1) \Rightarrow \frac{k(k+1)+2(k+1)}{2}$

$\Rightarrow \frac{(k+1)(k+2)}{2} \Rightarrow \frac{(k+1)((k+1)+1)}{2}$

k+1

Factorial (n) → n=5

5! → using recursion

Assumption + My work = Expectation

① Expectation :- Find factorial of n (n=5)

② Assumption / Faith :- → Find the factorial of (n-1) → 4! → subproblem

③ My work :- → 5! = 5 × 4! (4! — Assumption)

④ Base case :- if (n==0) return 1

```
ps int factorial (int n) {
    if (n==0)
        return 1;                    → base
    int subAns = factorial(n-1);     → 1
    int ans = subAns * n;            → 2
    return ans;
}
```

Do the dry run

n=5 → n=4 → n=3 → n=2 → n=1 → n=0

stopping cond^n → base case

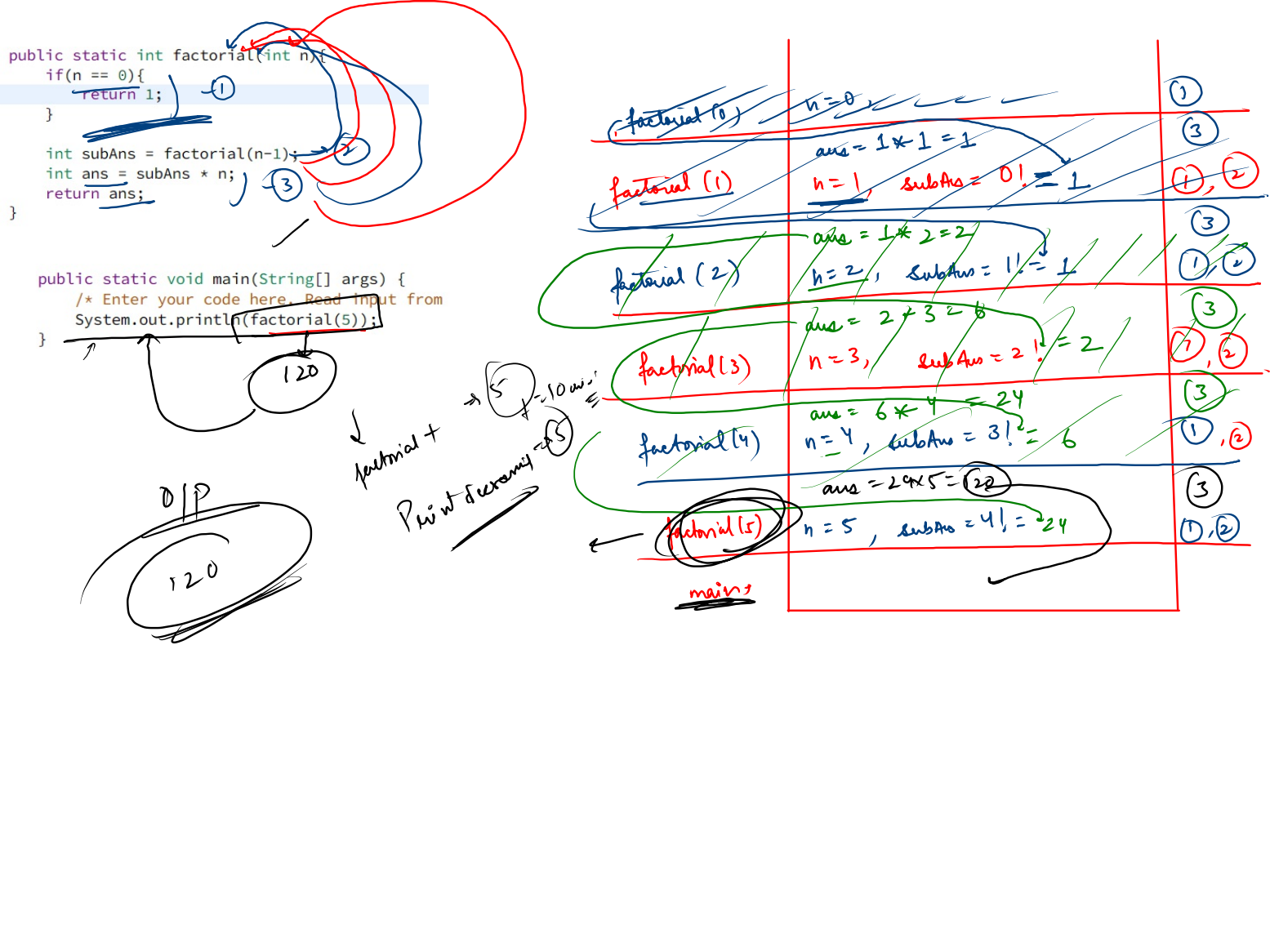
n=-1 → n=-2 → n=-3 → n=-4 → ... → n=-∞

5! = 10

n=7 → 0! = 1

n=100 → 0! = 1

5! → 4! → 3! → 2! → 1! → 0! → n==0 → return 1


public static int factorial(int n){
if(n == 0){
return 1;
}
int subAns = factorial(n-1);
int ans = subAns * n;
return ans;
}
public static void main(String[] args) {
/* Enter your code here. Read input from
System.out.println(factorial(5));
}
120
factorial +
Pending
O/P
120
factorial(0)
n=0
ans = 1*1 = 1
factorial(1)
n=1, subAns = 0! = 1
ans = 1*2 = 2
factorial(2)
n=2, subAns = 1! = 1
ans = 2*3 = 6
factorial(3)
n=3, subAns = 2! = 2
ans = 6*4 = 24
factorial(4)
n=4, subAns = 3! = 6
ans = 24*5 = 120
factorial(5)
n=5, subAns = 4! = 24
main

```java
import java.io.*;
import java.util.*;
import java.text.*;
import java.math.*;
import java.util.regex.*;

public class Solution {
    public static int factorial(int n){
        if(n == 0) return 1;

        return factorial(n-1)*n;
    }

    public static void main(String[] args) {
        /* Enter your code here. Read input from STDIN. Print output to STDOUT. Your class s
        Scanner scn = new Scanner(System.in);
        int n =scn.nextInt();
        System.out.println(factorial(n));
    }
}
```

Print Decreasing (Day 23)

Problem | Submissions | Leaderboard | Discussions

1. You are given a positive number n.
2. You are required to print the counting from n to 1.
3. You are allowed to not use any loops.

Note -> The online judge can't force you to write the function recursively but that is what the spirit of question is. Write recursive and not iterative logic. The purpose of the question is to aid learning recursion and not test you.

Sample Input 0

```
5
```

Sample Output 0

```
5
4
3
2
1
```

n = 5

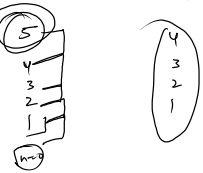
5 4 3 2 1

expectation :- Print decreasing order from n to 1 i.e. 5 to 1

Assumption / faith → Print the decreasing order from (n-1) to 1 i.e. 4 to 1

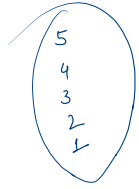
My work :- Print the value of 'n'

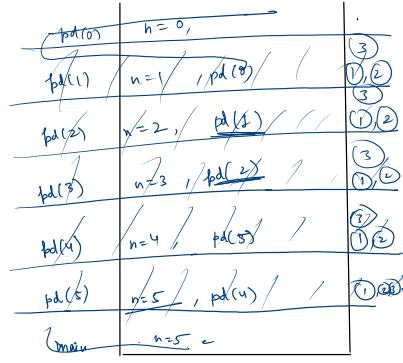
Base case :- if (n == 0) return



4 3 2 1

```
void printDecreasing(int n){
if (n == 0) return; -(1)
System.out.println(n); -(2)
printDecreasing(n-1); -(3)
}
```



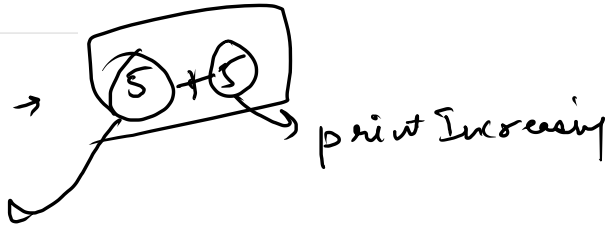


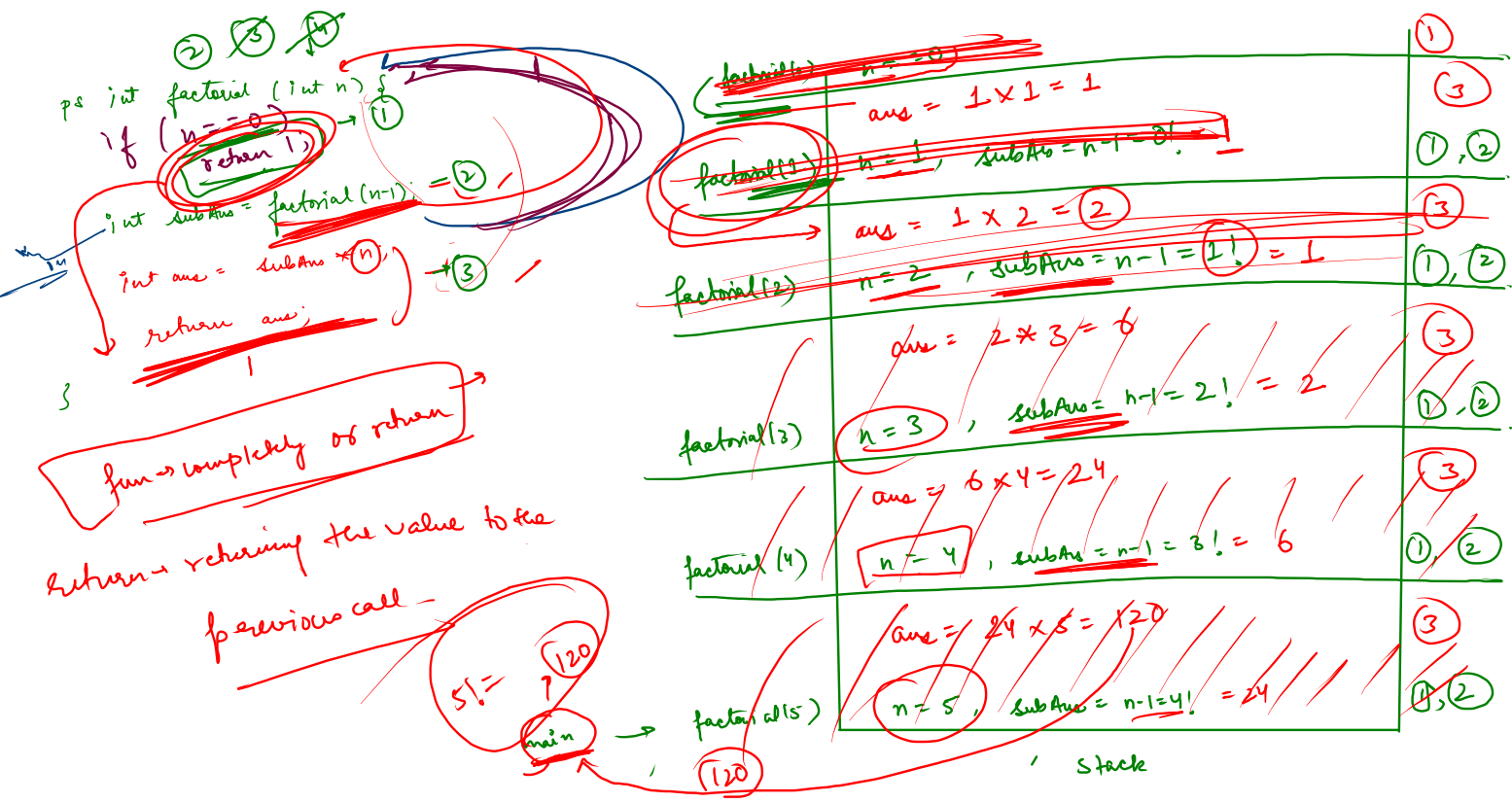
```java
import java.io.*;
import java.util.*;
import java.text.*;
import java.math.*;
import java.util.regex.*;

public class Solution {
    public static void printDecreasing(int n){
        if(n == 0) return;

        System.out.println(n);
        printDecreasing(n-1);
    }

    public static void main(String[] args) {
        /* Enter your code here. Read input from STDIN. Print ou
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();
        printDecreasing(n);
    }
}
```




int factorial (int n) {
if (n == 0)
return 1;
int subAns = factorial(n-1);
int ans = subAns * n;
return ans;
}
fun → completely or return
Return → returning the value to the previous call
5! = ?
120
main
factorial(5)
factorial(4)
factorial(3)
factorial(2)
factorial(1)
factorial(0)
n = 0
ans = 1 x 1 = 1
n = 1, subAns = n-1 = 0!
ans = 1 x 2 = 2
n = 2, subAns = n-1 = 1! = 1
ans = 2 * 3 = 6
n = 3, subAns = n-1 = 2! = 2
ans = 6 x 4 = 24
n = 4, subAns = n-1 = 3! = 6
ans = 24 x 5 = 120
n = 5, subAns = n-1 = 4! = 24
stack

Print Increasing   n=5

1
2
3
4
5

① expectation → Print increasing order of n i.e ⇒ n=5   1 2 3 4 5

② Faith :– → Print increasing order of n-1 ⇒ n=4   1 2 3 4

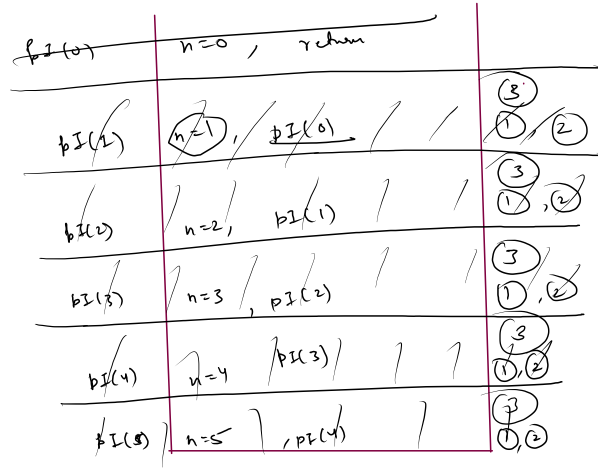
③ My work → Sysout (n);

④ Base case   if (n==0) return;

```
ps void pI ( int n) {
    if (n==0)
        return;         — ①
    pI ( n-1);          — ②
    Sysout ( n);        — ③
```

5

1
2
3
4
5



```java
import java.io.*;
import java.util.*;
import java.text.*;
import java.math.*;
import java.util.regex.*;

public class Solution {

    public static void printIncreasing(int n){
        if(n == 0) return;
        printIncreasing(n-1);
        System.out.println(n);
    }
    public static void main(String[] args) {
        /* Enter your code here. Read input from STDIN. Prir
        Scanner scn = new Scanner(System.in);
        int n = scn.nextInt();
        printIncreasing(n);
    }
}
```

Power-linear (Day 23)

Problem | Submissions | Leaderboard | Discussions

1. You are given a number x.
2. You are given another number n.
3. You are required to calculate x raised to the power n.

Sample Input 0

```
2
5
```

Sample Output 0

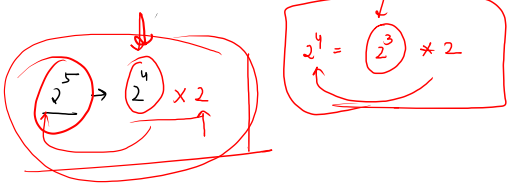
```
32
```

$x = 2$

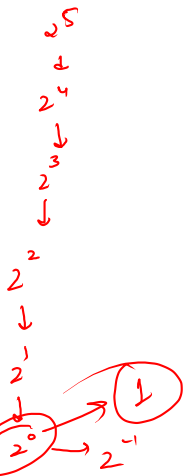
$n = 5$

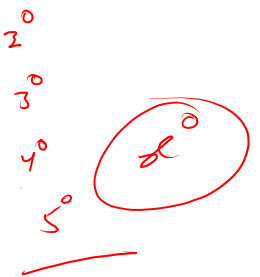
x^n

$2^5 = 32$



1. Expectation → Find x^4
2. Faith/Assumption → Find x^{n-1}
3. My work → $x^{n-1} * x \Rightarrow x^n$
4. base case → if ($x == 0$) → return 1

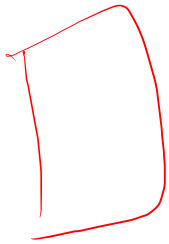




Tomorrow

6 pm

↳ Photo on whatsapp



```java
import java.io.*;
import java.util.*;
import java.text.*;
import java.math.*;
import java.util.regex.*;

public class Solution {
    public static int powerLinear(int x,int n){
        if(n == 0) return 1;
        int subAns = powerLinear(x,n-1);
        int ans = subAns * x;
        return ans;
    }

    public static void main(String[] args) {
        /* Enter your code here. Read input from STDIN. Print output to STD
        Scanner scn = new Scanner(System.in);
        int x = scn.nextInt();
        int n = scn.nextInt();

        System.out.println(powerLinear(x,n));
    }
}
```

→ How?

Dry Run of powerLinear