18 Oct

18 October 2022 21:01

Recursion:-

TOH

for loop

[1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

i

Pre-ORDER

Post-ORDER

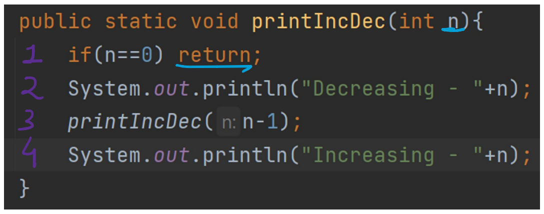
```
n= 10
sol(n) {
if(n==0) return;
Sout(n);
Sol(n-1)
}
```

Pre-Order

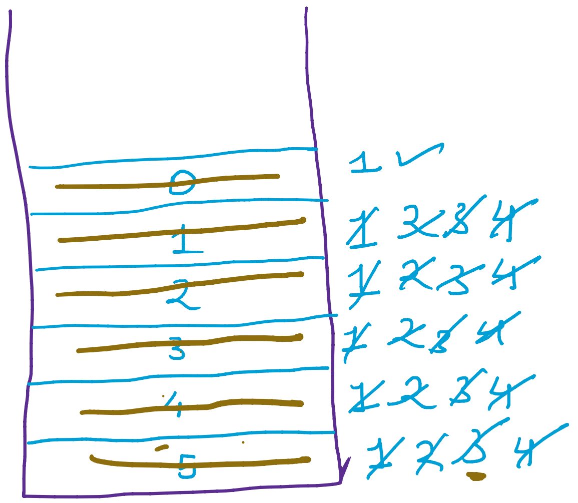
```
sol(n) {
if(n==0) return;

Sol(n-1)
sout(n)
}
```

Post



```
public static void printIncDec(int n){
1   if(n==0) return;
2   System.out.println("Decreasing - "+n);
3   printIncDec( n: n-1);
4   System.out.println("Increasing - "+n);
}
```



D - 5
D - 4
D - 3
D - 2
D - 1
I - 1
I - 2
I - 3
I - 4
I - 5

Power- https://leetcode.com/problems/powx-n/

x , n
↓
2.0, 10 => 1024.0

$2^{10} = \underline{2^5} \times 2^5$

```
Pow(x, n) {
if(n==0) return 1;
half = Pow(x, n/2);

if(n%2==0) {
```

$$2^{10} = \underline{2^5} \times 2^5$$

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

$$2^3 = 2^2 \times 2^1$$

$$2^2 = 2^1 \times 2^1$$

```
if(n%2==0){
  return half × half;
else
  return (half×half) × x
```

$$2^{-2} = \frac{1}{2^2} = \frac{1}{4} = \underline{0.25}$$

$$2^{-5} = 2^{-3} \times 2^{-2}$$

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

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

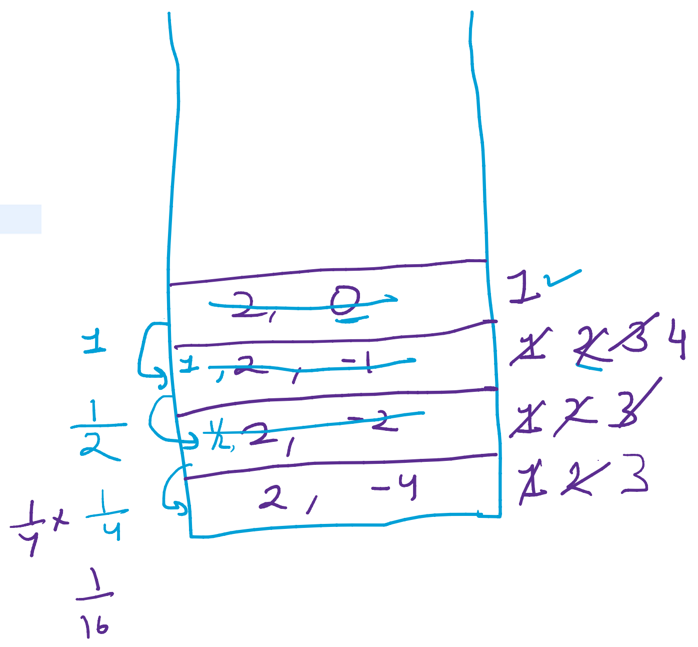
$$2^{0} = 1$$

```
public double myPow(double x, int n) {
1  if(n==0)return 1;
2   double half=myPow(x,n/2);
3   if(n%2==0){
        return half*half;
    }else{
4       if(n<0) return (half*half)/x;
5       else return half*half*x;
    }
}
```

2^{-4}

$$\frac{1\times1}{2} = \frac{1}{2}$$

$$\frac{1}{2}\times\frac{1}{2} = \frac{1}{4}$$



Print Triplets:-

n = 1, O/P = 1 1 1

n = 2, O/P = 2 1 1 1 2 1 1 1 2

n = 3, O/P = 3 2 1 1 1 2 1 1 1 2 3 2 1 1 1 2 1 1 1 2 3