Recursion

1 to 5

```
for(i=1; i<=5; i++)
    S.o.p(i)
```

1 to n

5 to 1

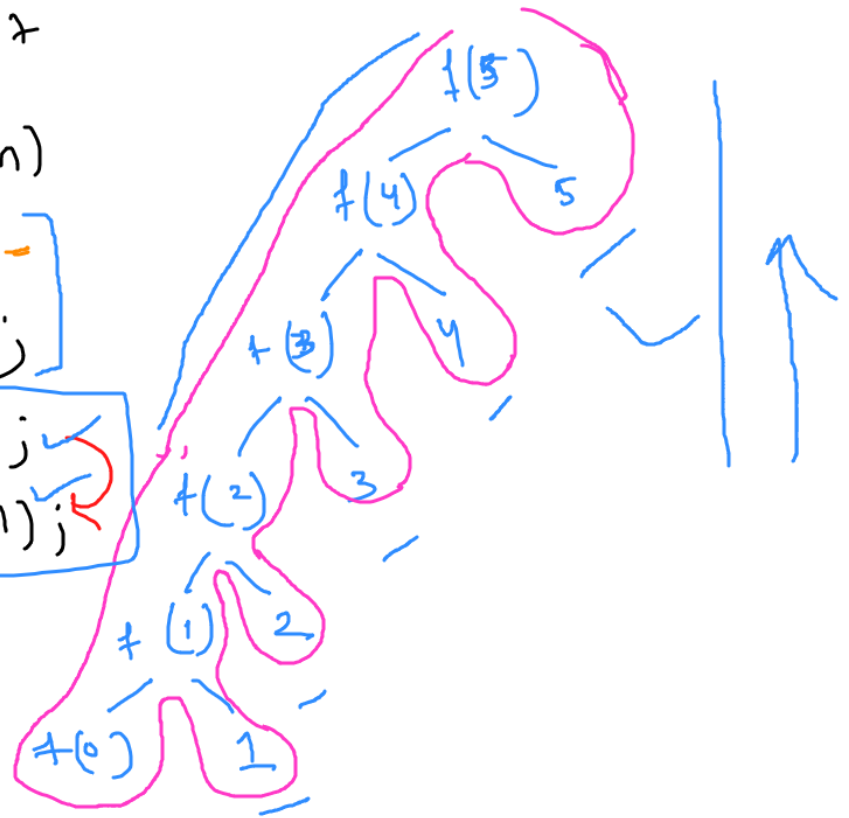
```
for(i=5; i>=1; i--)
    S.o.p(i)
```

```
S.o.p(n)
fun(n-1);
```

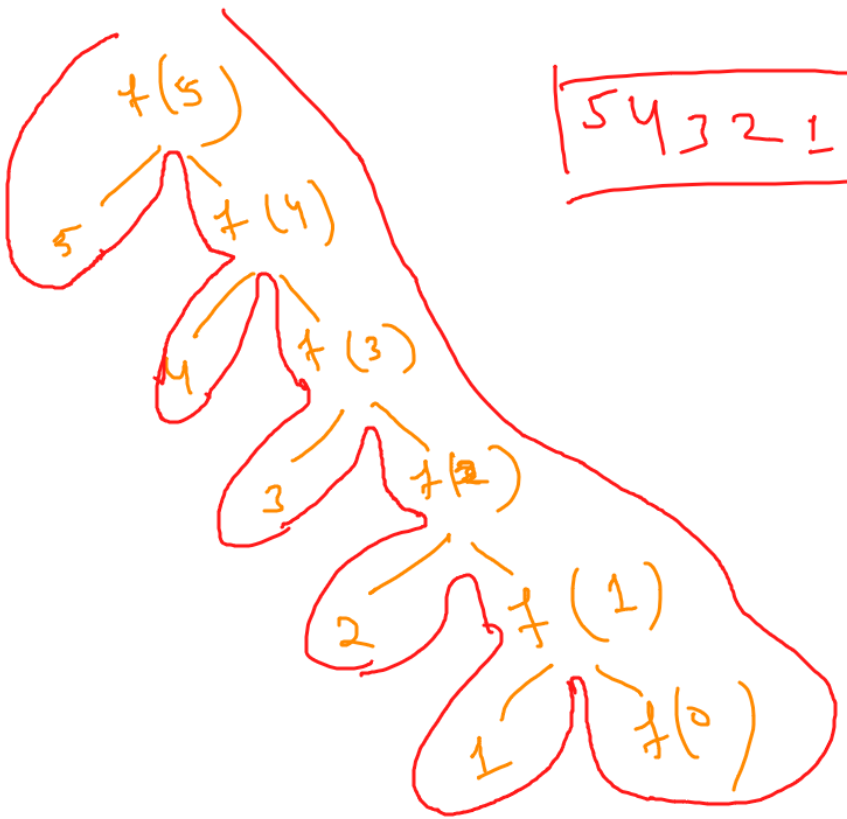
1 to 7 n=7

```
void fun(int n)
{
    if(n==0)
        return;
    fun(n-1);
    S.o.p(n);
}
```

1 2 3 4 5 6 7

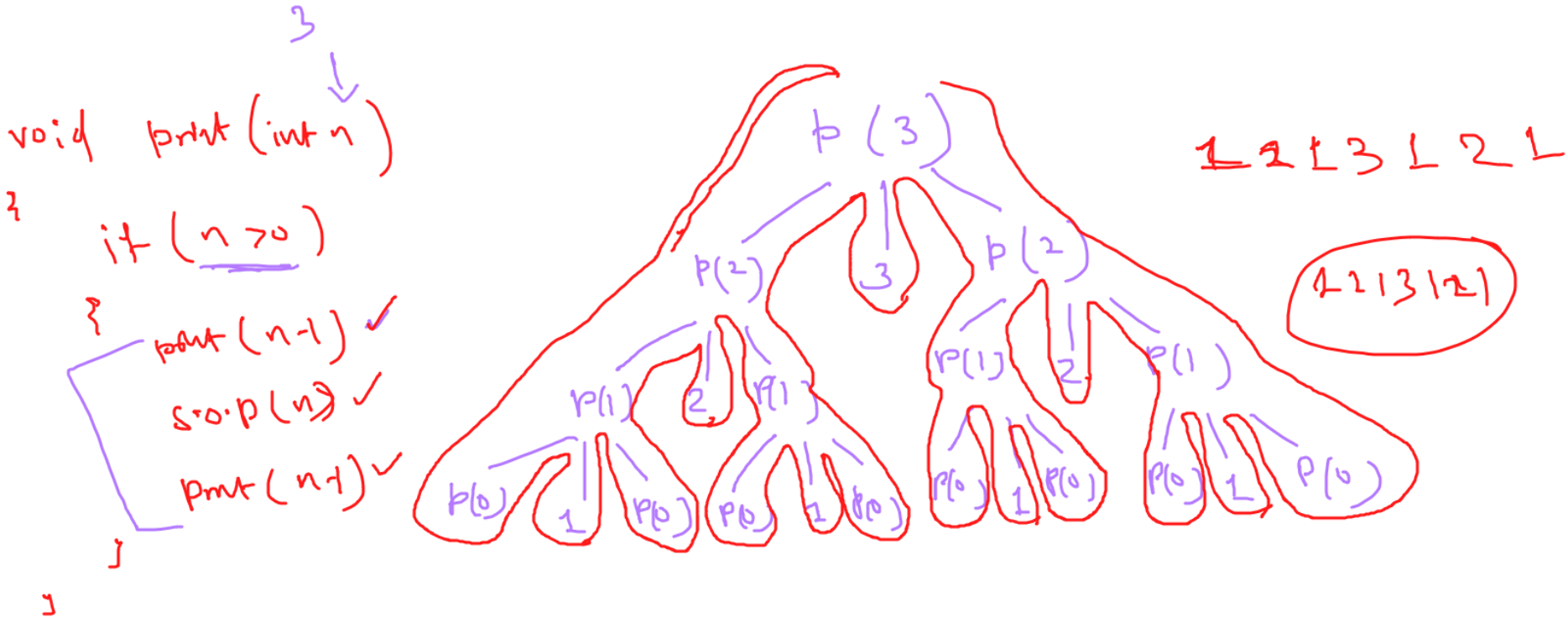


Head ✓
Tail
Double order.

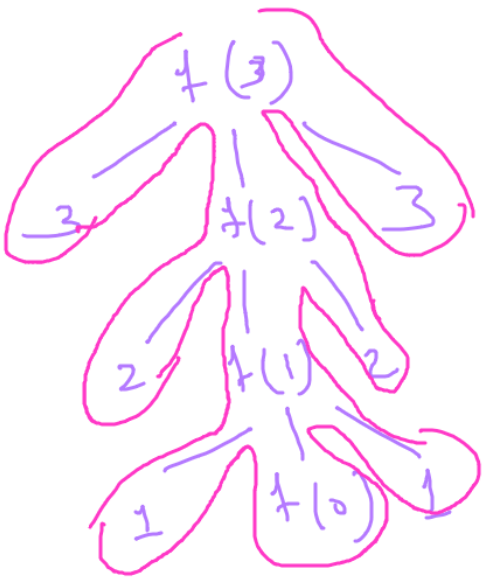


5 4 3 2 1

```
void fun(n)
{
    if(n==0)
        return;
    S.O.P(n);
    fun(n-1);
}
```

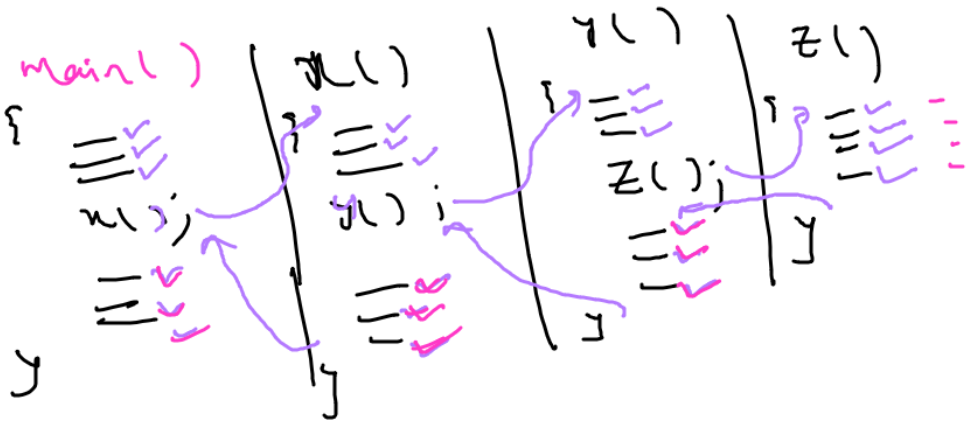

3
void print(int n)
{
if (n > 0)
{
print(n-1)
s.o.p(n)
print(n-1)
}
}
p(3)
p(2)
3
p(2)
p(1)
2
p(1)
p(1)
2
p(1)
p(0)
1
p(0)
p(0)
1
p(0)
p(0)
1
p(0)
p(0)
1
p(0)
1 2 1 3 1 2 1
1 2 1 3 1 2 1

```
void fun(int n)
{ if(n>0)
  {
    printf(n)
    fun(n-1)
    printf(n)
  }
}
```



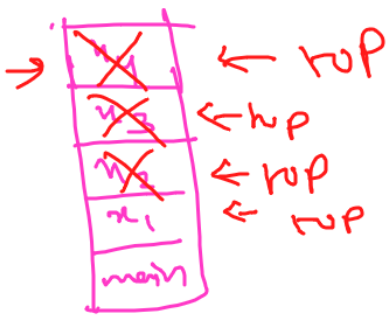
3 2 1 1 2 3

Usage of stack memory in function calls



```
n()
{
  ==
  n()
  ==
}
```

StackOverflow



Iteration

Recursion

```
int gcd(int a, int b)
{
    if(b == 0)
        return a;
    return gcd(b, a%b);
}
```

gcd(5, 7)

gcd(7, 5)

gcd(5, 2)

gcd(2, 1)

gcd(1, 0)

↳ 1

LCM * HCF = a * b

$\frac{a*b}{HCF}$ → LCM

Factorial (n).

```
f = 1;
for ( i = 1; i <= n ; i++ )
{
    f *= i;
}
```

```
int fact (int n)
{
    if (n == 0)
        return 1;

    return n * fact(n-1);
}
```

5! = 5 * 4!

n! = n * (n-1)!

0! = 1

1! = 1