Print 1 to N without Loop

I/p : N = 5

O/p : 1 2 3 4 5

I/p : N = 3

O/p : 1 2 3

I/p : N = 0

O/p :

NAIVE

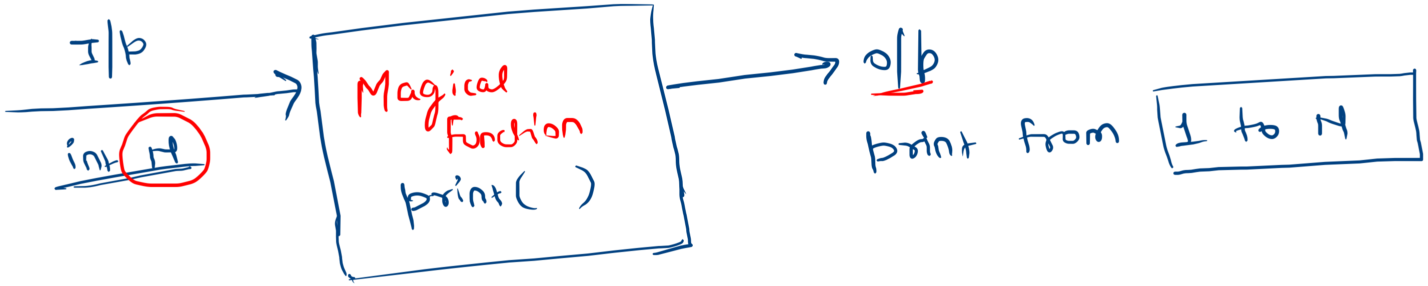
N = 5

o/p: 1 2 3 4 5

```
for( int i=1 ; i<=5 ; i++)
{
    cout << i << endl;
}
```

1 2 3 4 5

Recursive Solution



I/p

int N

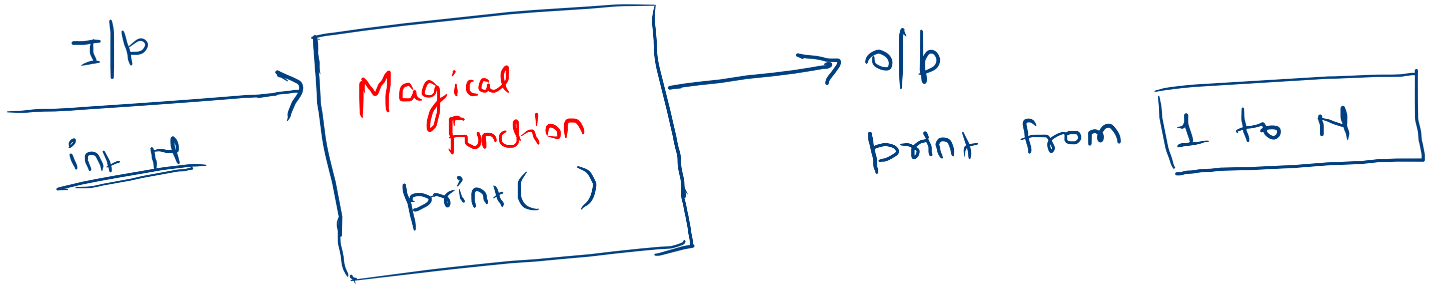
Magical Function

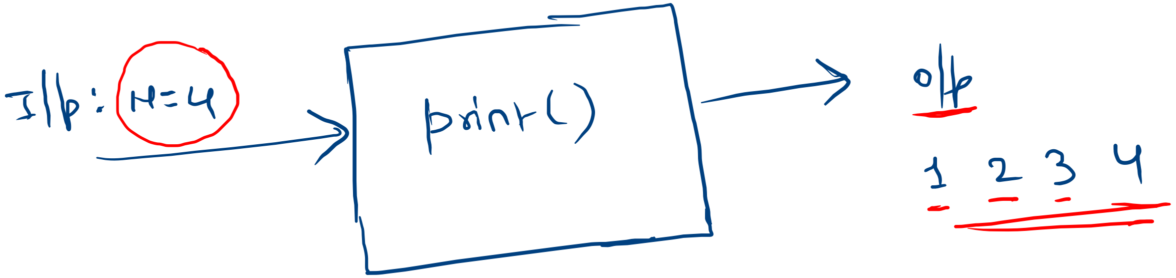
print( )

o/p

print from 1 to N

Recursive solution





Logical part

I/p: N = 5

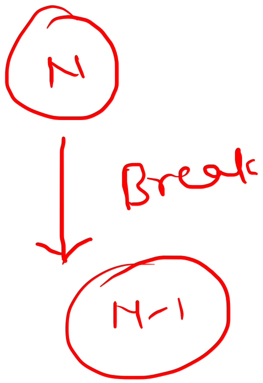
O/p: 1 2 3 4 5

↓

print(4)   cout << 5;

↓

1 2 3 4 5



Logical part

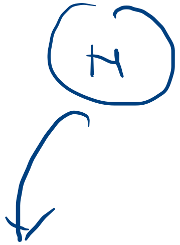
I/p : N = 5

O/p : 1 2 3 4 $\boxed{5}$

$\boxed{\texttt{print(4)}}$ $\boxed{\texttt{cout << 5 ;}}$

generalise

$\boxed{\texttt{print(N-1)}}$ $\boxed{\texttt{cout << N ;}}$



Recursive function

```
void print ( int N )   5
{
    // Base condition ✓
    print (N-1);
    cout<<N;
}
```

print(4)

1 2 3 4

70%

Base Condⁿ :

Smallest Valid Input → Check output

```
int print(int n)
{

}
```

→ 0, 1, 2, 3 . . .

↓

Smallest Valid I/p

Void

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

Recursive function

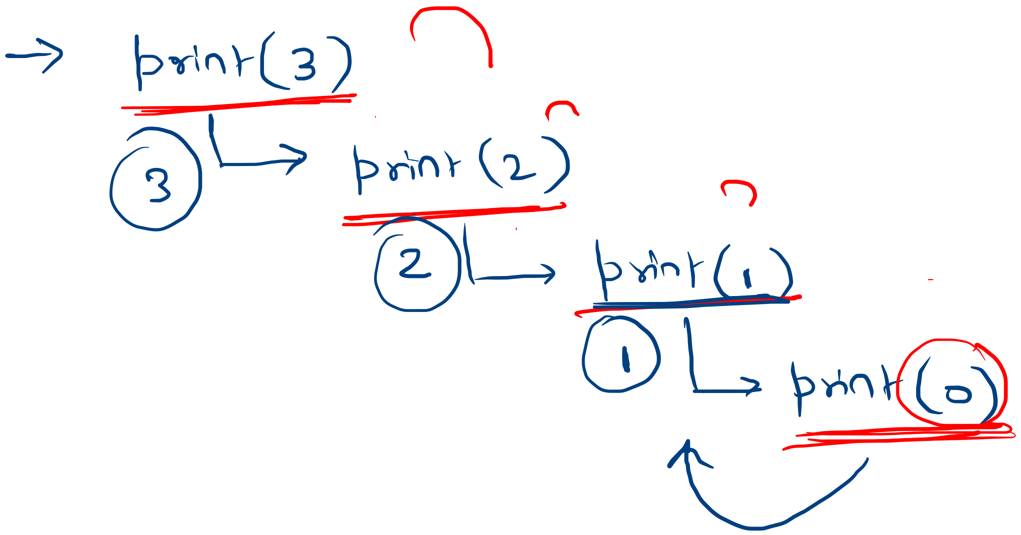
```
Void print ( int N )
{
    if (N==0) return;  ✓
 →  print (N-1);  ✓
    cout<<N;
}
```

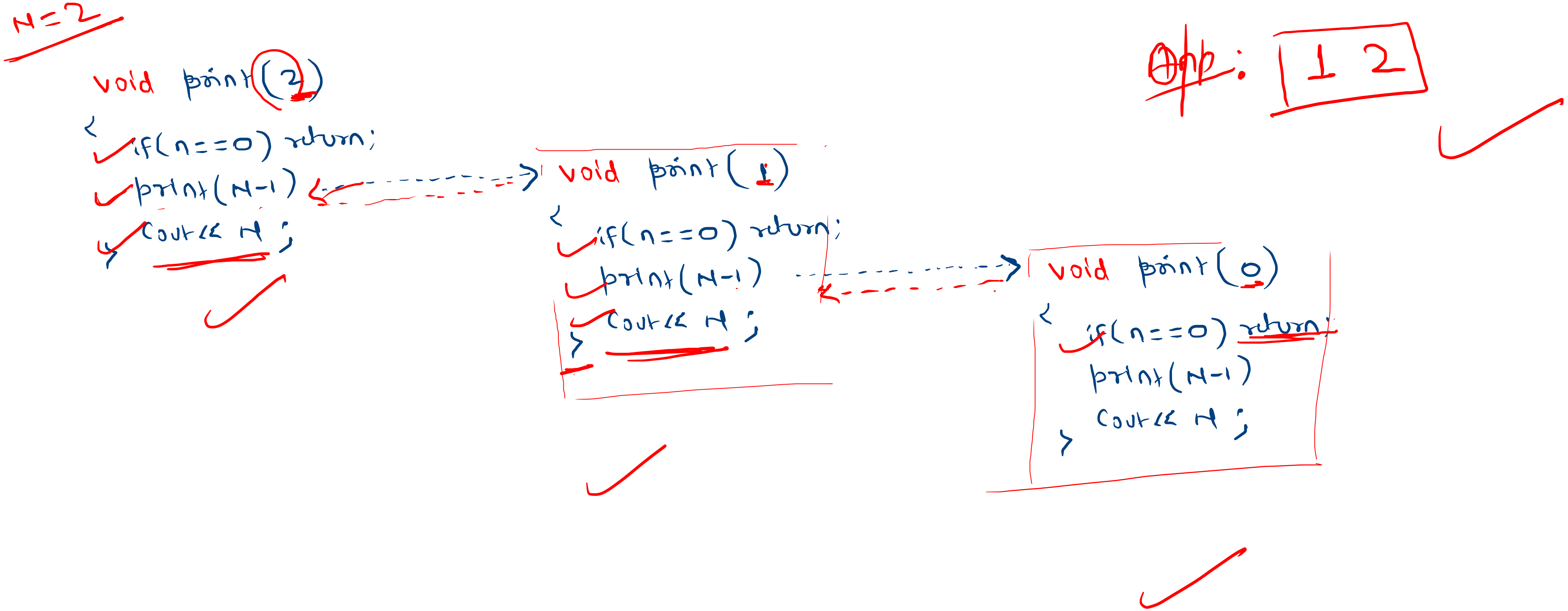
1   N-1

Recursive Tree

```
void print ( int N )
{
    if (N==0) return;

    print (N-1);
    cout<<N;
}
```





N=2

```
void print(2)
{
if(n==0) return;
print(N-1)
cout<< N ;
}
```

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

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

O/p : 1 2