5. Multiple Recursion Calls

Multiple Recursion Calls :

$$f()\ \{\ \ \left.\begin{array}{l} f() \\ f() \end{array}\right\} \text{2 times}\ \ \}$$

Fibonacci

$$\begin{array}{ccccccccccc} 0 & 1 & 1 & 2 & 3 & \overset{f(5)}{5} & 8 & 13 & 21 & 34 & \dots \\ \swarrow & \swarrow & \swarrow & \swarrow & & & & & & & \\ 0^{th} & 1^{st} & 2^{nd} & 3^{rd} & \dots & & & & & & \end{array}$$

$\underline{N.} \longrightarrow \quad f(N) \longrightarrow N^{th}$ fibonacci nos.

$f(3) \longrightarrow 2$

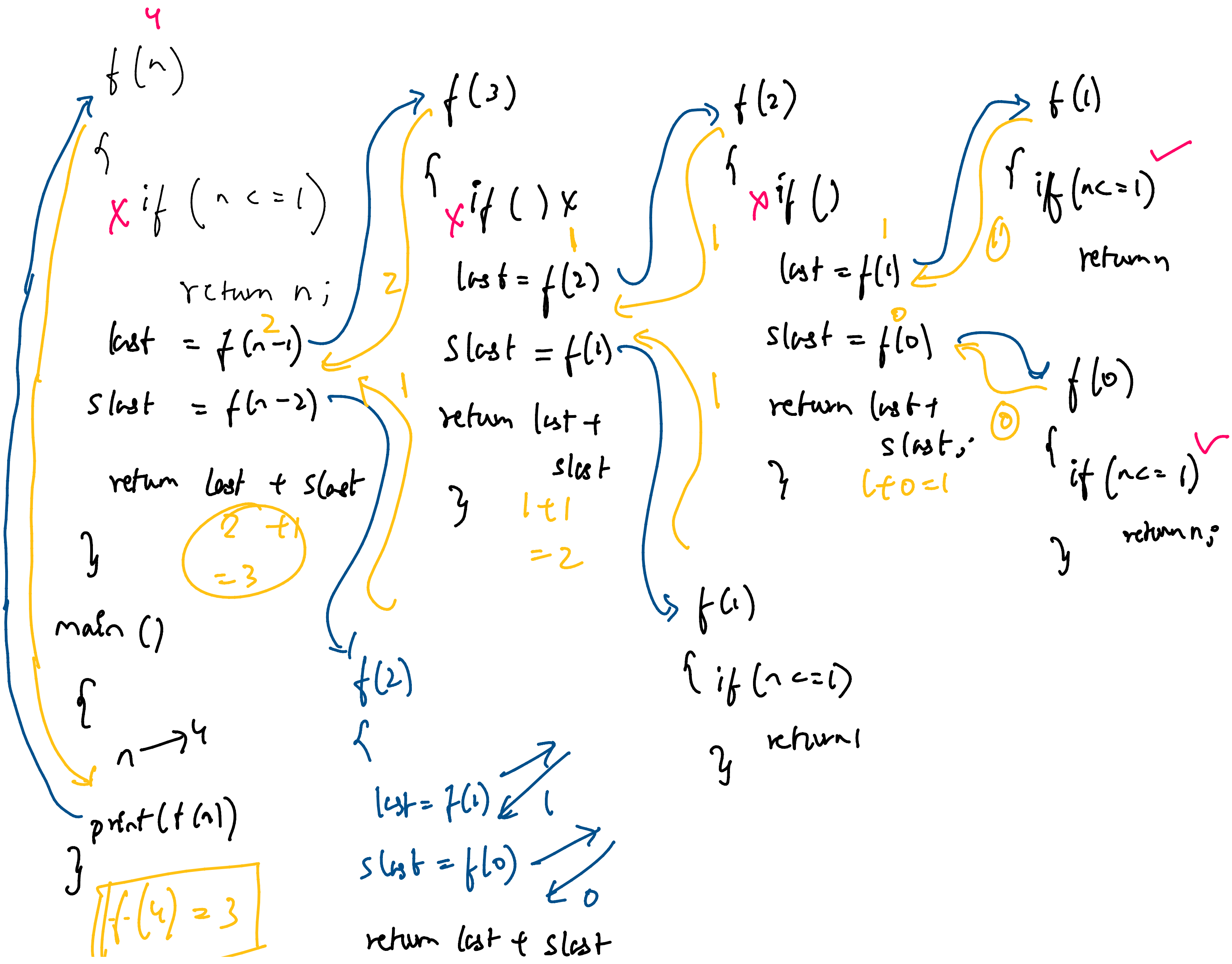
$f(4) \longrightarrow 3$

$f(5) \Rightarrow f(4) + f(3)$

Iterative Approach

```
f[0] = 0      f[1] = 1
for ( int i = 2 → n)
        f[i] = f(i-1) + f(i-2)
```

$$f(n) = f(n-1) + f(n-2)$$



```
f(n)
{
  if (n <= 1)
     return n;
  last = f(n-1)
  slast = f(n-2)
  return last + slast
}
main ()
{
  n → 4
  print(f(n))
}
```

```
f(3)
{
  if ( )
  last = f(2)
  Slast = f(1)
  return last +
         slast
}
```

```
f(2)
{
  if ( )
  last = f(1)
  slast = f(0)
  return last +
         slast;
}
```

```
f(1)
{ if (n<=1)
     return n
```

```
f(0)
{
  if (n<= 1)
}   return n;
```

```
f(1)
{ if (n<=1)
}   return 1
```

```
f(2)
{
  last = f(1)
  slast = f(0)
  return last + slast
```

$\boxed{f(4) = 3}$

O/P

← 0

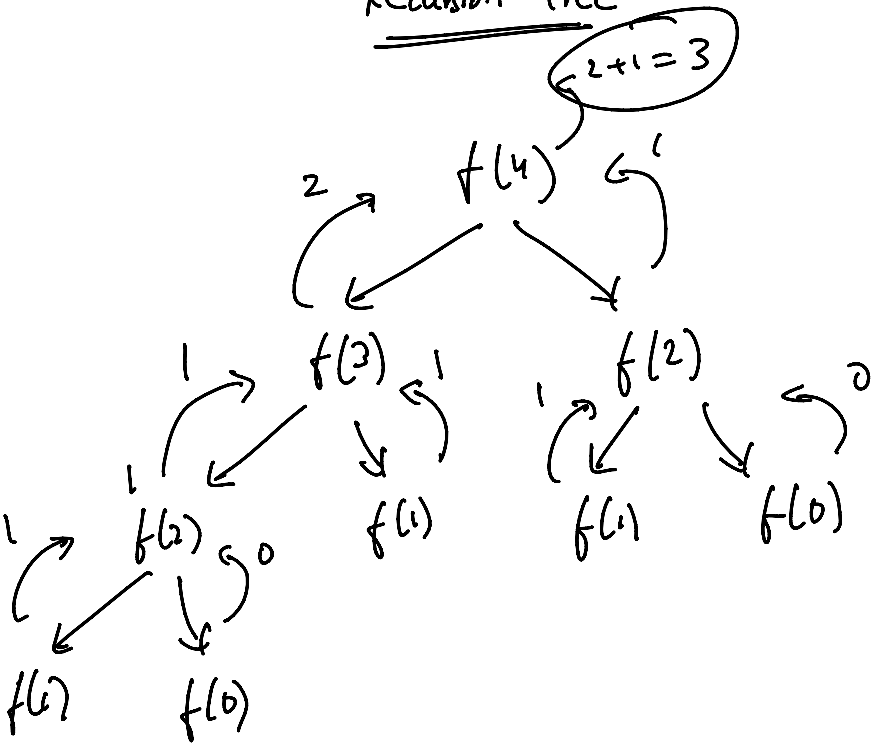
```
return last + slast
}
```

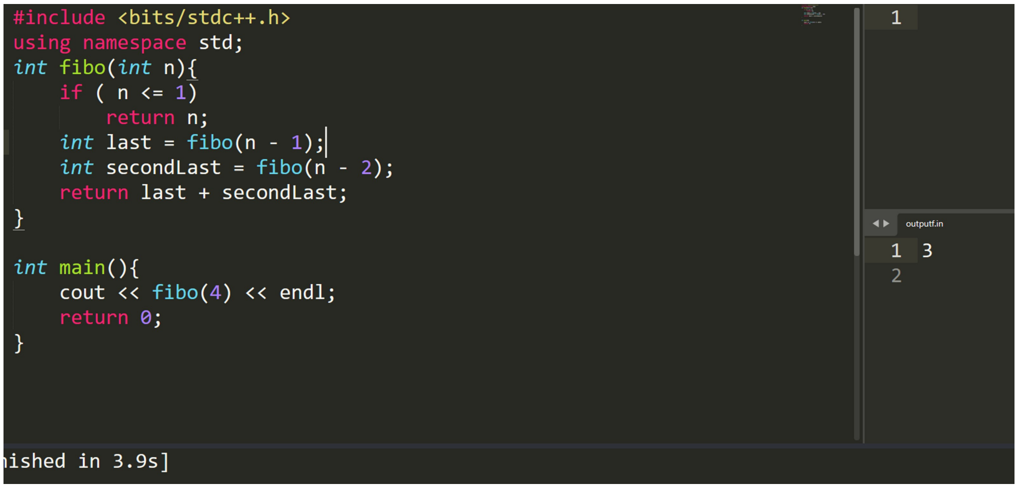
1 + 0 = 1

```
f()
{
    f()
    f()
    f()
}
```

One is ended, next comes.

Recursion Tree





```cpp
#include <bits/stdc++.h>
using namespace std;
int fibo(int n){
    if ( n <= 1)
        return n;
    int last = fibo(n - 1);
    int secondLast = fibo(n - 2);
    return last + secondLast;
}

int main(){
    cout << fibo(4) << endl;
    return 0;
}
```

```
3
```

```
ished in 3.9s]
```

T.C ⇒ $O(2^n)$ Exponential in nature