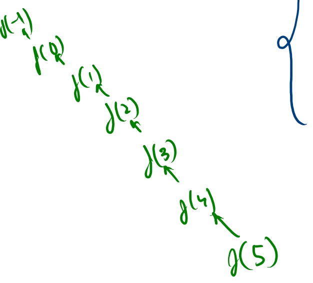
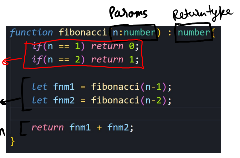


Params　Returntype



```
function fibonacci(n:number) : number{
    if(n == 1) return 0;
    if(n == 2) return 1;

    let fnm1 = fibonacci(n-1);
    let fnm2 = fibonacci(n-2);

    return fnm1 + fnm2;
}
```

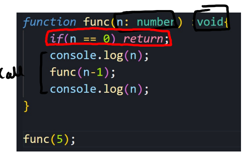
Base cases

Calls

Return

1) Params
2) Return
3) Base Case
4) Calls
5) Return

Call



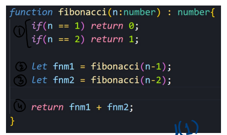
```
function func(n: number) :void{
    if(n == 0) return;
    console.log(n);
    func(n-1);
    console.log(n);
}

func(5);
```

factorial calc.

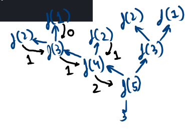
n!

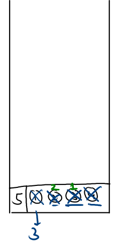
5! ⇒ 1·2·3·4·5

n=5 ⇒ 120



```
function fibonacci(n:number) : number{
  ① if(n == 1) return 0;
    if(n == 2) return 1;

  ② let fnm1 = fibonacci(n-1);
  ③ let fnm2 = fibonacci(n-2);

  ④ return fnm1 + fnm2;
}
```





4! = 4·(3·2·1)

5! = 5·4!

5! = 5·4!

fact(n) = n· fact(n-1)

funct. fact(n): number {
  if(n == 1) return 1
  let fnm1 = fact(n-1)
  return n·fnm1;
}

String with difference of Every two chars

a c d e b a A

⇓

a2c1d1e3b-1a-32A

a c d e b a A

1) ith char
if(i+1 < n){
2) ascii diff add
}

ans = a2c1d1e-3b-1a-32A

S = ~~XXXXS~~
t = nnaam

m=15

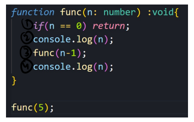
a=97
m=109
n=110

→ aamnn
sort
→ aamnn

if(s.len != t.len) return false

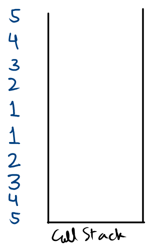
ascii(　) - 97 = idx

freqs =	2	_	_	_	_	......	1	2	......	_
	a	b	c	d	e	......	m	n	......	z
	0	1	2	3	4	......	12	13	......	25
freqt =	2	_	_	_	_	......	1	2	......	_



```
function func(n: number) :void{
  ①if(n == 0) return;
  ②console.log(n);
  ③func(n-1);
  ④console.log(n);
}

func(5);
```

•) Call → add to callstack
•) return → remove from callstack
•) Execution always happens on top of the call stack



flower　flow　flo?　float

0th　1st　2nd ...

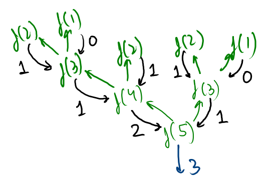
ans = f l o

for(ith char in arr[0])
　for(jth string)　2
　　arr[0][i] == arr[j][i]
　　　0 2　　　1 2

fibonacci Numbers　fib(8) = fib(7) + fib(6)

0	1	1	2	3	5	8	13
1st	2nd	3rd	4th	5th	6th	7th	8th

→ faith
→ expectation



f(2)
f(1)
f(3)
f(2)
f(2)
f(1)
f(4)
f(3)
f(5)
1
0
1
1
1
0
1
2
1
3