Recursion - Theory

When a function calls itself until a specified condition is met

can be single or multiple (base condition)

```
void f(){
    cout<<1;
    f(); }

main(){
    f(); ②
}
```

①

```
void f(){
    cout<<1;
    f();
}
```

keeps going forever
"infinite recursion

```
void f(){
cout<<1;
f()
}
```

Keeps printing 1 until it runs out of memory.

stack

f(), l2
f(), l2
f(), l2
f(), l2
f(), l2

"Stack overflow" — numerous function calls waiting due to recursion

Segmentation flow
All these functions are not complete

```
count = 0
f() {
    cout << (count++);   → 0
    count++; #1
    f();
}
main() {
    f();
}
```

→ 1 #2 → 2 #3

```
count = 0
f() {
    if (count == 3)
        return;
    cout << (count++);
    count++;
    f();
}
main() {
    f();
}
```

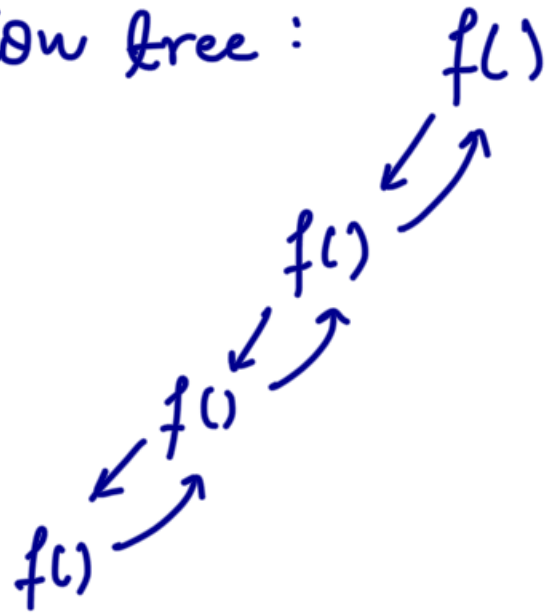
```
f() {
    if (count == 3)
        return;
    ---- x
    ---- x
    ---- x
}
```

output = 0
1
2

function gets terminated

f(),f5 → 2
f(),f5 → 2
f(),f5 → 3

Recursion tree :



Stack space : Stores yet to be completed functions