# Programming in C

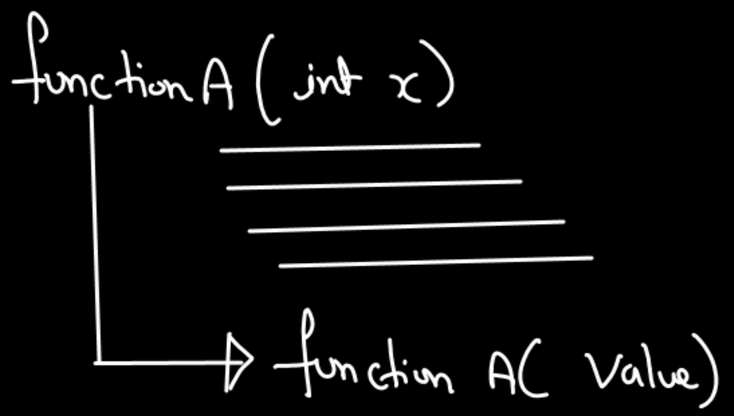
## Lecture - 29

## Recursion

# Recursion

↳ It is a process where a function calls itself directly or indirectly to solve a problem.

⇒ This technique is used to solve a problem that can be broken down into smaller sub-problems.



```
int fun ( int x ) {
    if (x == 1) {
        return (x);
    }
    else {
        return (x + fun (x - 1));
    }
}

int main ( ) {
    printf("%d", fun(5));
    return 0;
}
```

5 → x = 5

Base Case

5 + fun(4)

5 + fun(x-1)

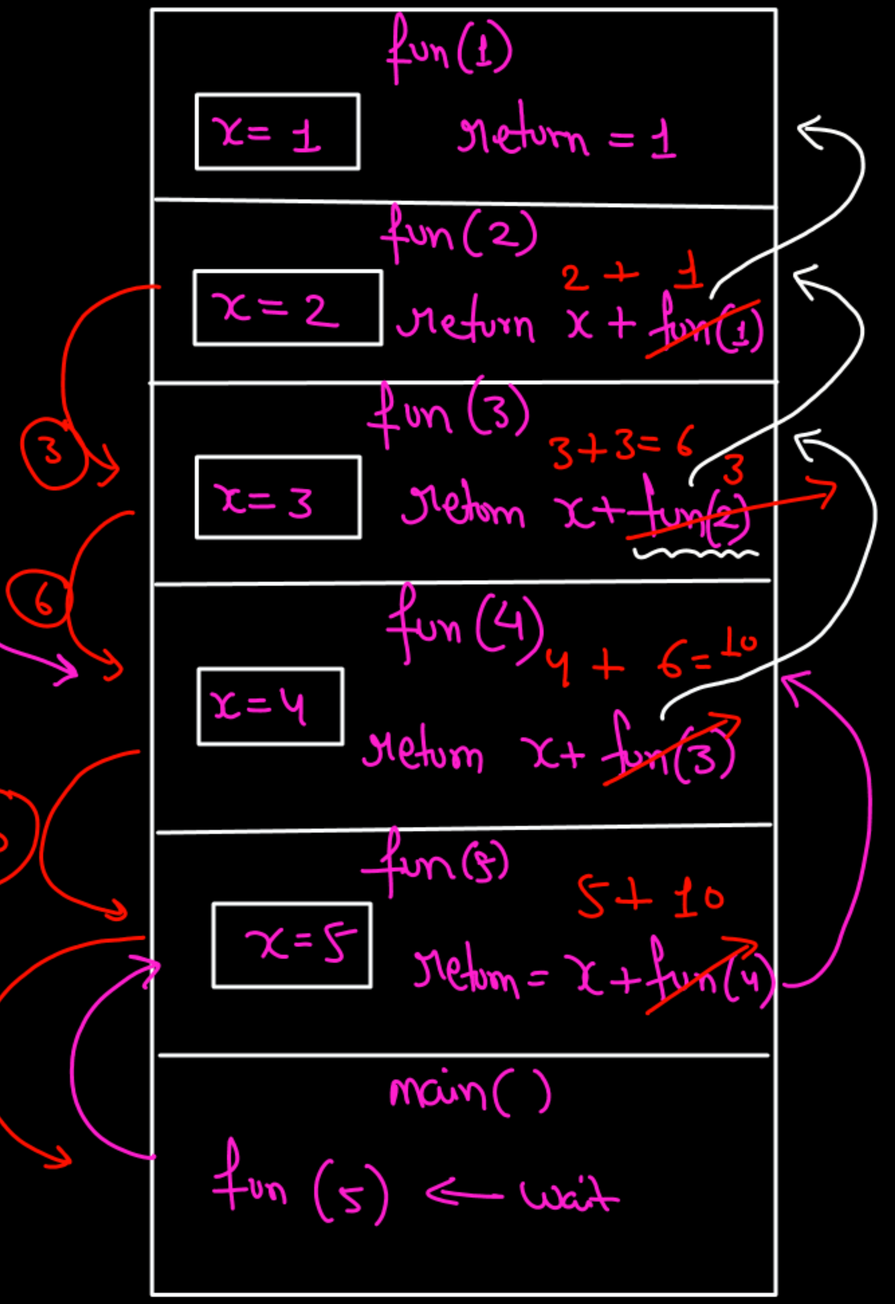
Recursive Case

Recursion

entry

output

15

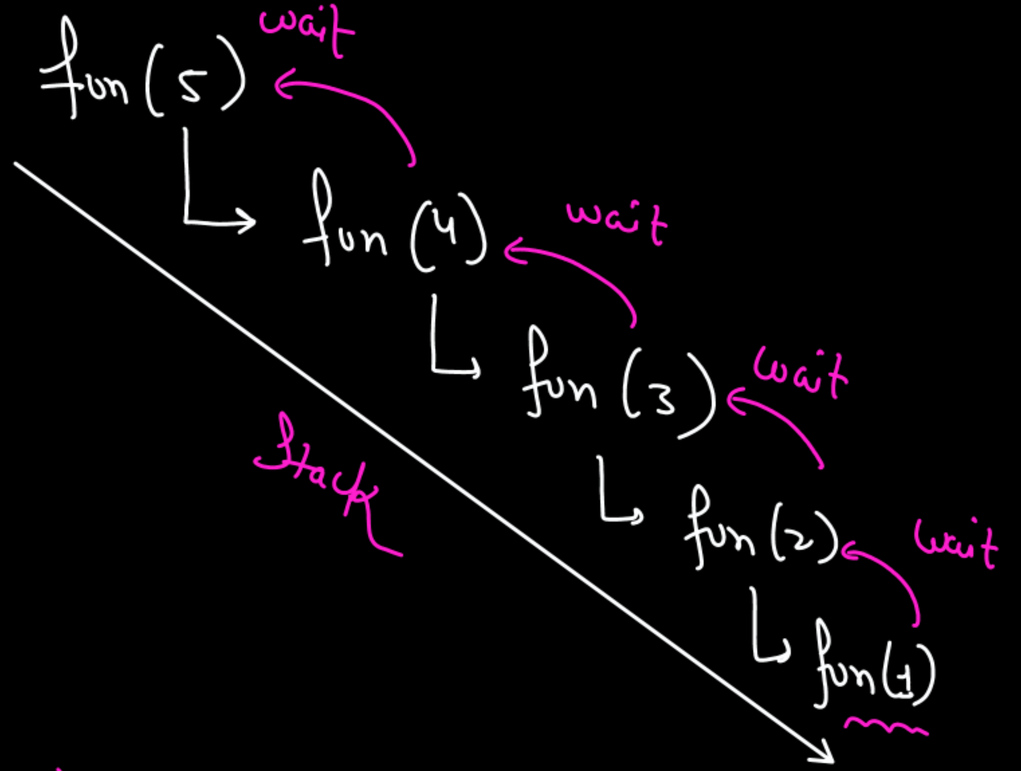


# How Recursion Works :

**Base Case** : A condition under which the function stop calling itself.

⇒ Without Base Case, the recursion continue infinitly. Causing Stack overflow.

**Recursive Case** : Part of a function that breaks the problem down and calls itself with a smaller / modified parameter.



```
int fact(int n){
    if (n==0){
        return 1;
    }
    else {
        return n * fact(n-1);
    }
}
```

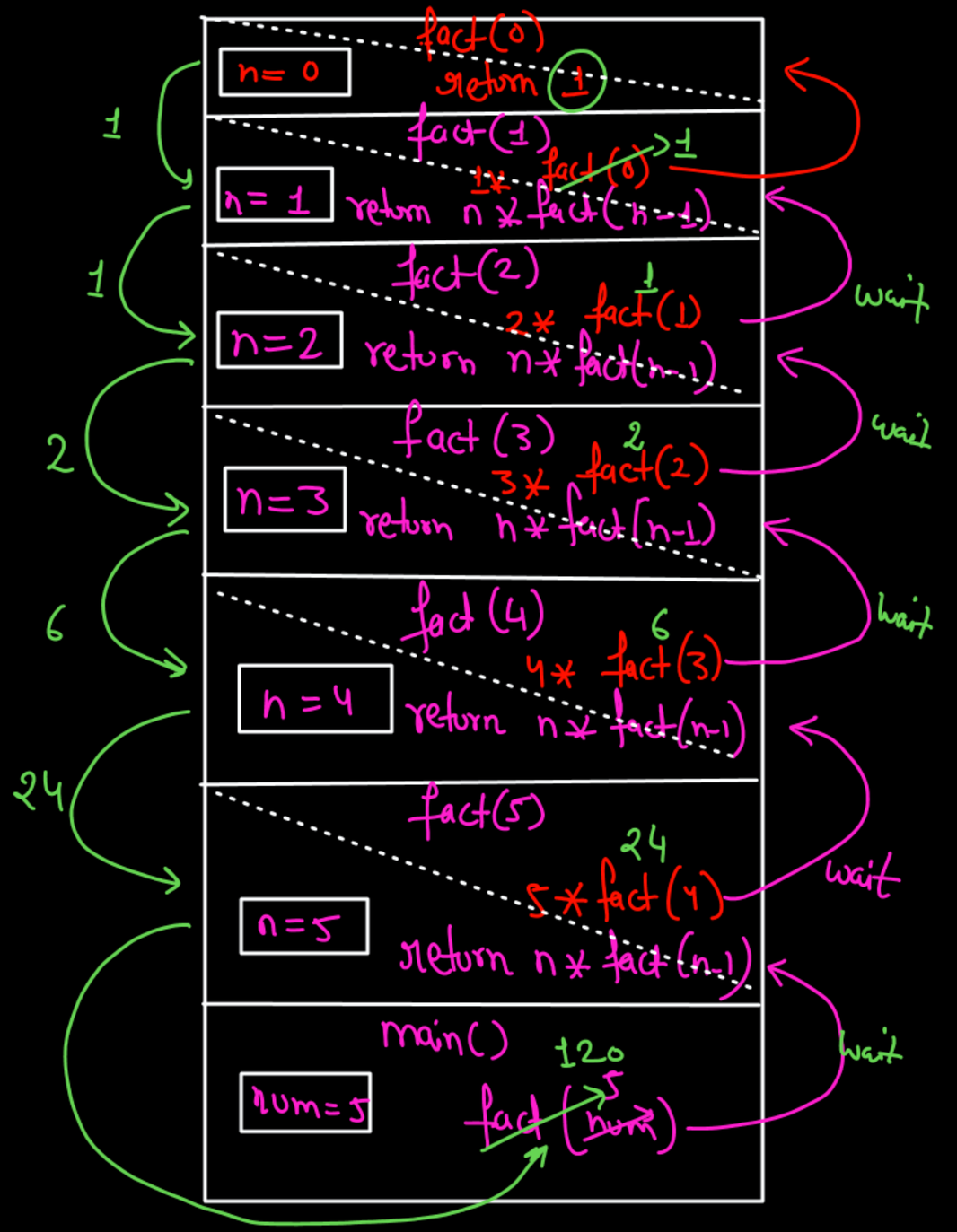
```
int main(){
    int num = 5;
    printf("%d", fact(num));
    return 0;
}
```

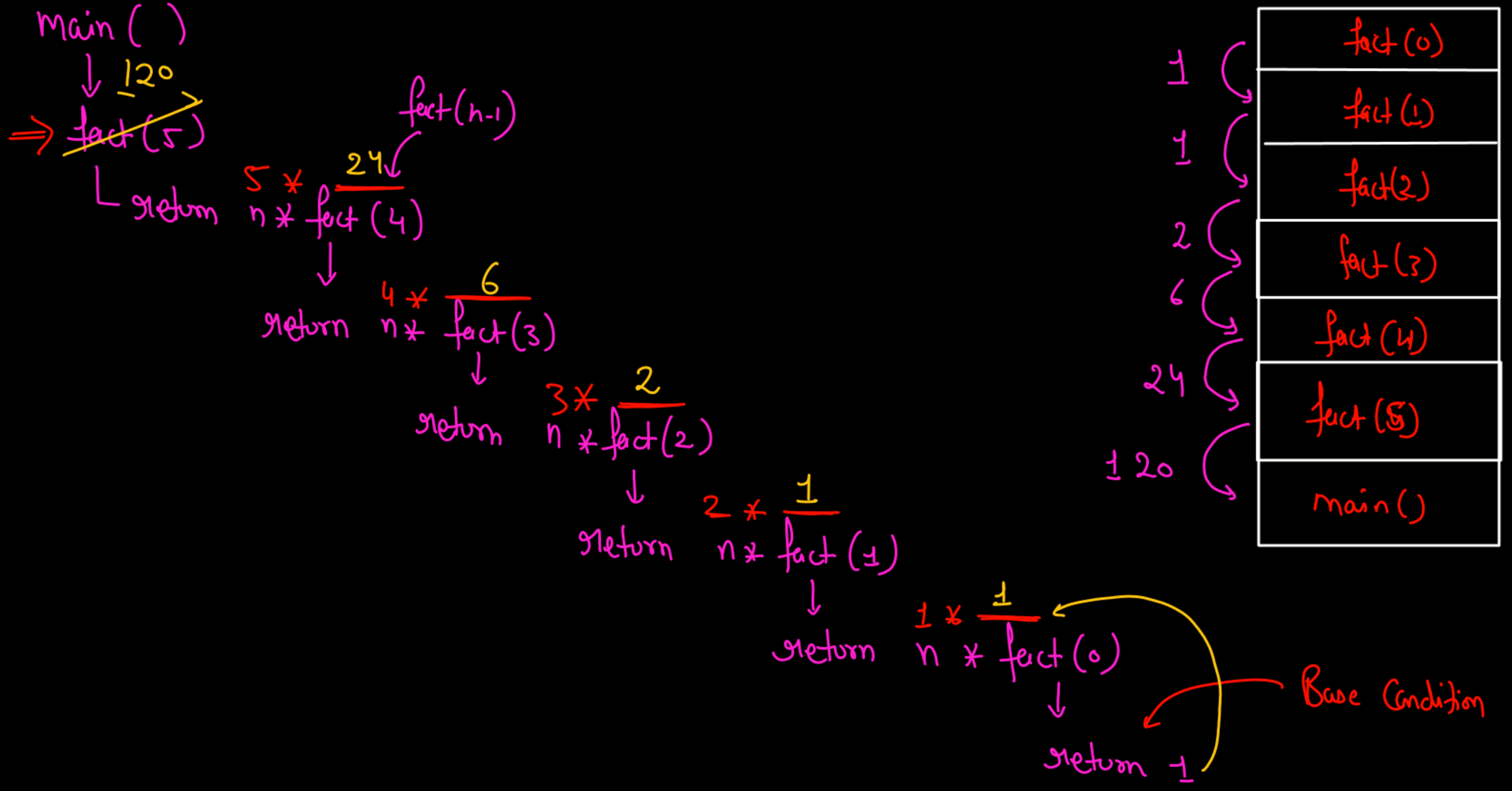
entry

120

int → 120

wait




main ( )
120
fact(5)
return
5 * fact(4)
fact(n-1)
24
return
4 * fact(3)
6
return
3 * fact(2)
2
return
2 * fact(1)
1
return
1 * fact(0)
1
return 1
Base Condition
1
1
2
6
24
120
fact(0)
fact(1)
fact(2)
fact(3)
fact(4)
fact(5)
main()

Tomorrow

↳ MCQ → 100 ← 2 videos

↓

[functions / Storage class / Recursion]

⇒ [Pointers & Arrays → Multidimensional Array] time