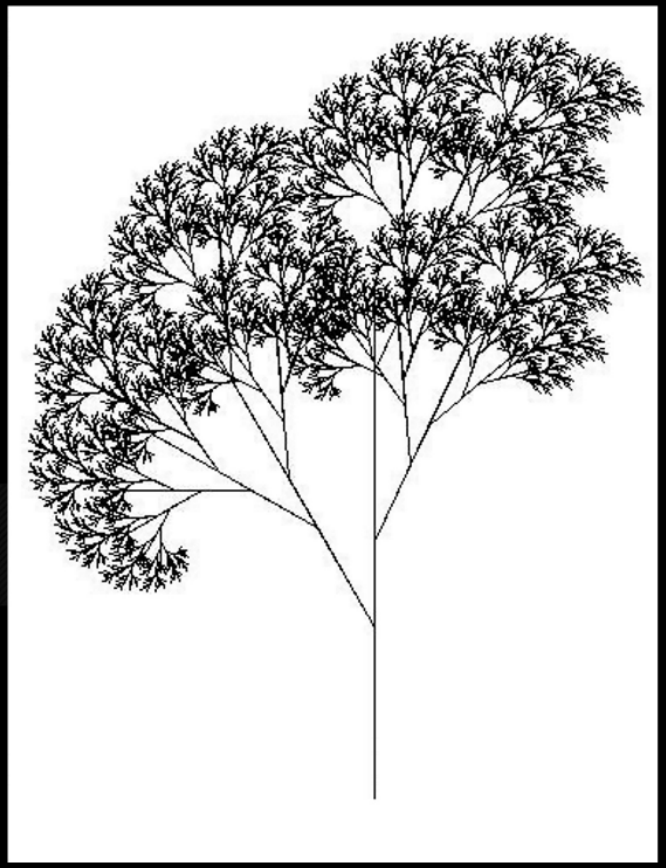
Recursion

The Loopless Loop

Recursion

Functions defined in terms of itself are called recursive functions




fun ( )
{
fun( );
}
Recursive call.

$n! = n \times (n-1) \times (n-2) \times (n-3) \cdots \times 1$

$n! = n \times (n-1)!$

fact (n) = n x fact (n-1)

```
factorial(int n)
```

$$factorial(n) = \begin{cases} 1 & n = 1 \text{ or } 0 \\ n \times factorial(n-1) & \text{o.w.} \end{cases}$$

Base case

Recursive step

```
factorial(int n)
{
    int fact;

    if (n==1)
        return(1);
    else
        fact = n*factorial(n-1);

    return(fact);
}
```

Base case

Recursive case

$$factorial(n) = \begin{cases} 1 & n = 1 \\ n \times factorial(n-1) \end{cases}$$

```
factorial(int n)
{
    int fact;

    if (n==1 || n==0)
        return(1);
    else
        fact = n*factorial(n-1);

    return(fact);
}
```

How does this work?

factorial(2) → 1

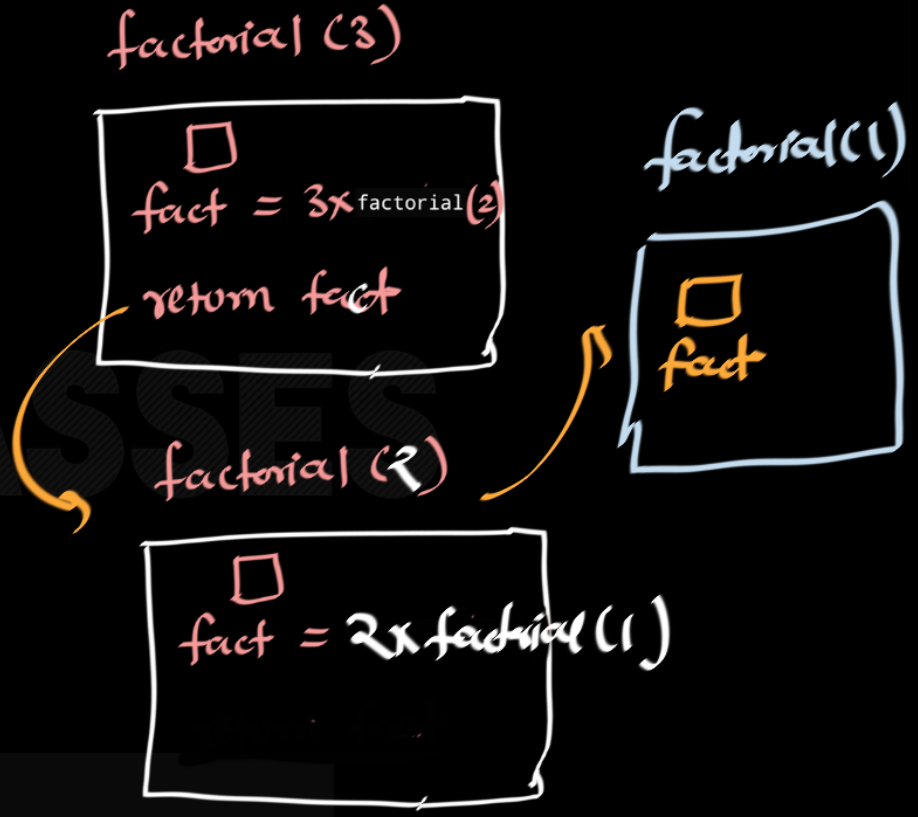
2 (factorial(2)

↳ 2 * factorial(1)

↖ 1

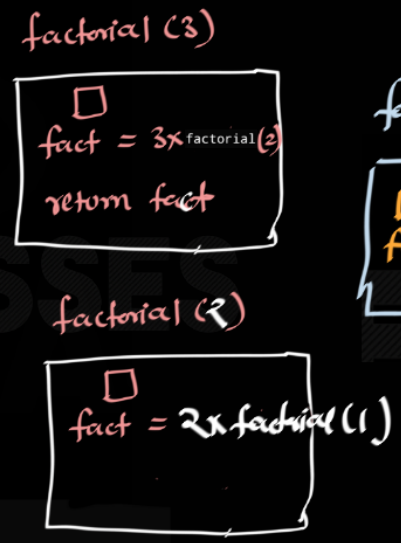
```
factorial(int n)
{
    int fact;

    if (n==1 || n==0)
        return(1);
    else
        fact = n*factorial(n-1);

    return(fact);
}
```

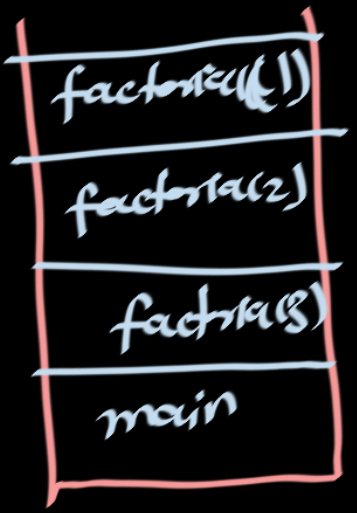
How does this work?



```
factorial(int n)
{
    int fact;

    if (n==1 || n==0)
        return(1);
    else
        fact = n*factorial(n-1);

    return(fact);
}
```

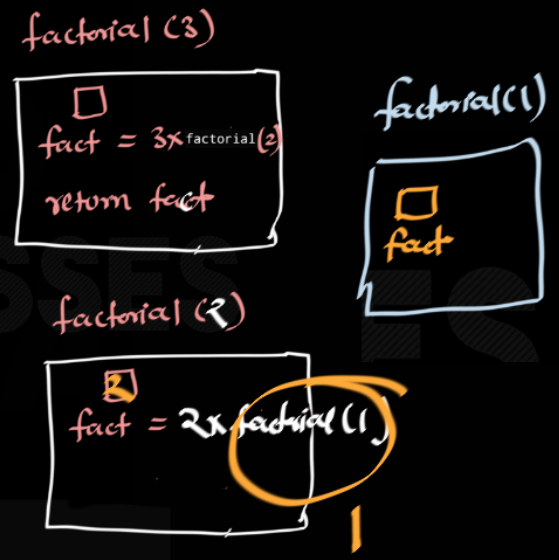
How does this work?





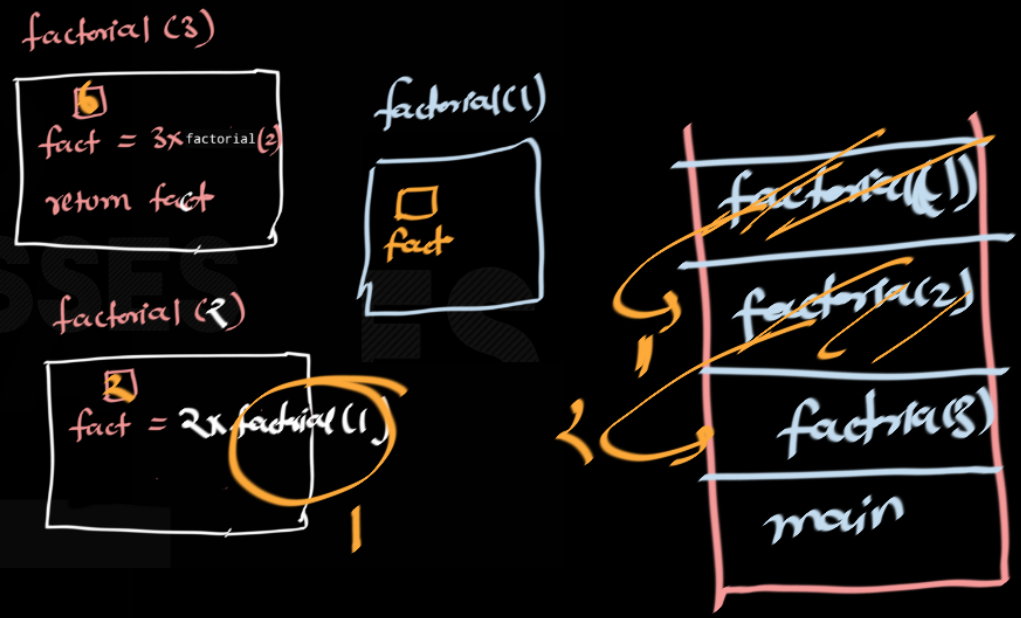
```
factorial(int n)
{
    int fact;

    if (n==1 || n==0)
        return(1);
    else
        fact = n*factorial(n-1);

    return(fact);
}
```

How does this work?



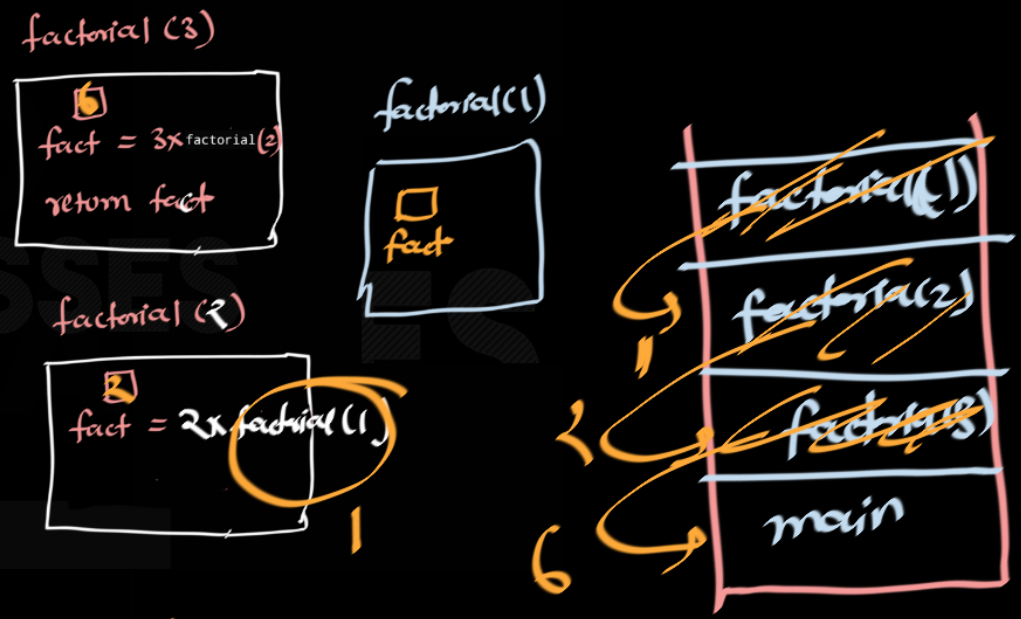
```
factorial(int n)
{
    int fact;

    if (n==1 || n==0)
        return(1);
    else
        fact = n*factorial(n-1);

    return(fact);
}
```

How does this work?



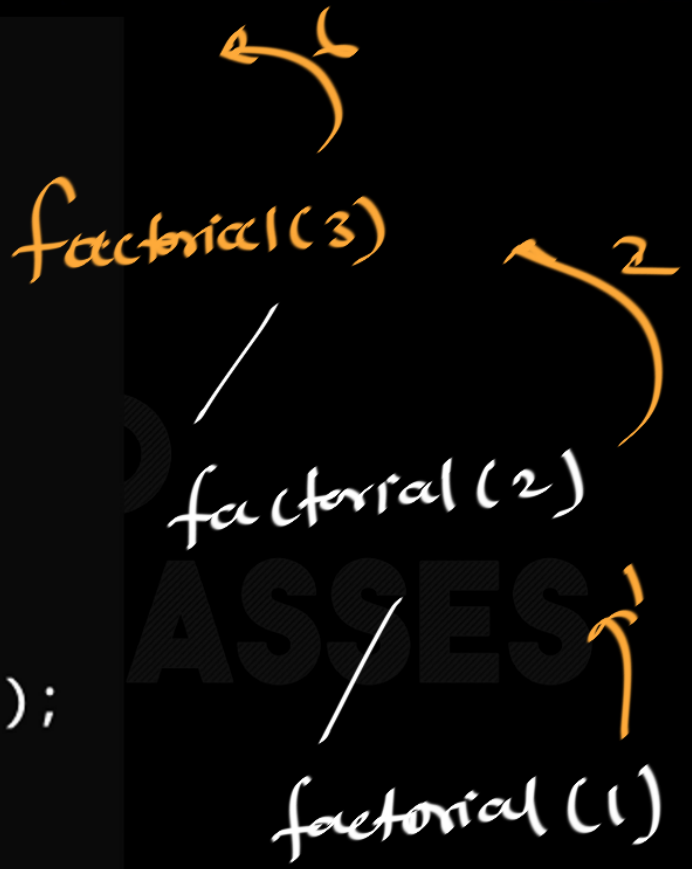
```
factorial(int n)
{
    int fact;

    if (n==1 || n==0)
        return(1);
    else
        fact = n*factorial(n-1);

    return(fact);
}
```

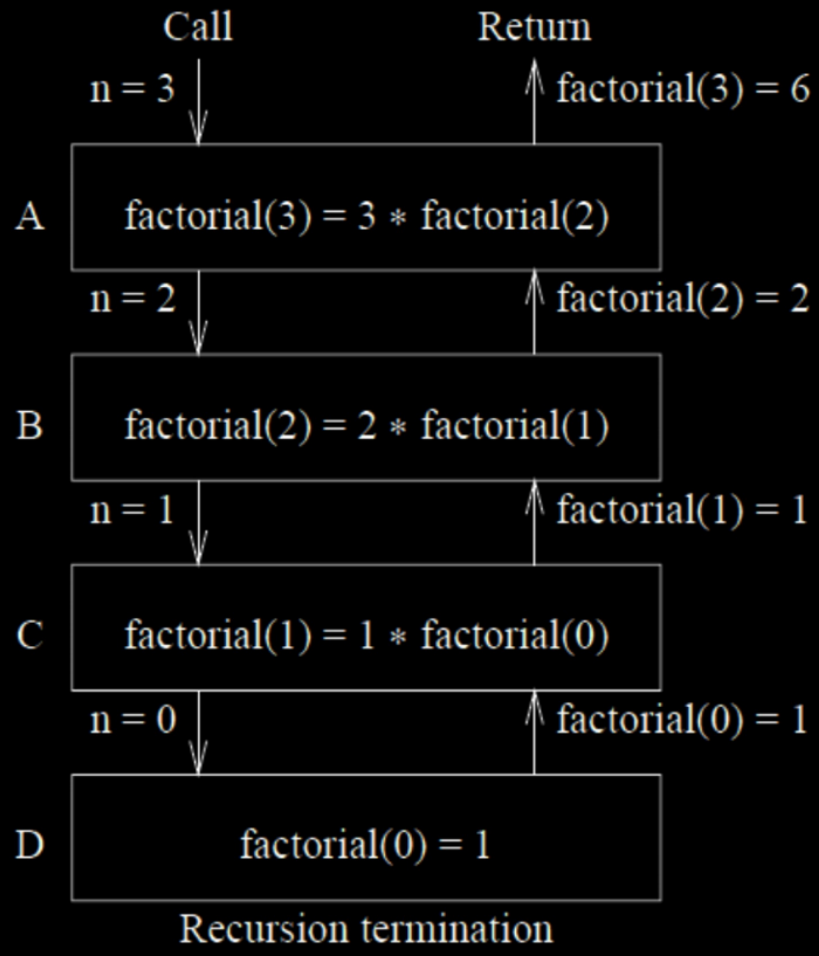
How does this work?



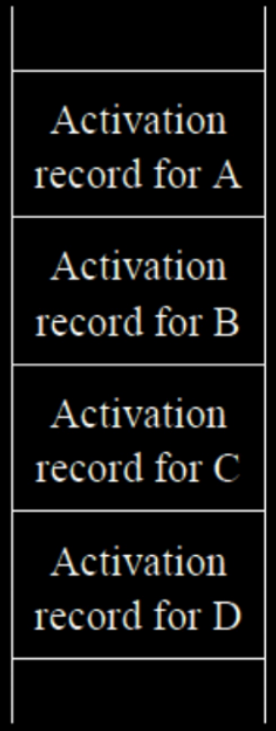
3x2x1 = 6

```
factorial(int n)
{
    int fact;

    if (n==1)
        return(1);
    else
        fact = n*factorial(n-1);

    return(fact);
}
```




Call
Return
n = 3
factorial(3) = 6
A
factorial(3) = 3 * factorial(2)
n = 2
factorial(2) = 2
B
factorial(2) = 2 * factorial(1)
n = 1
factorial(1) = 1
C
factorial(1) = 1 * factorial(0)
n = 0
factorial(0) = 1
D
factorial(0) = 1
Recursion termination

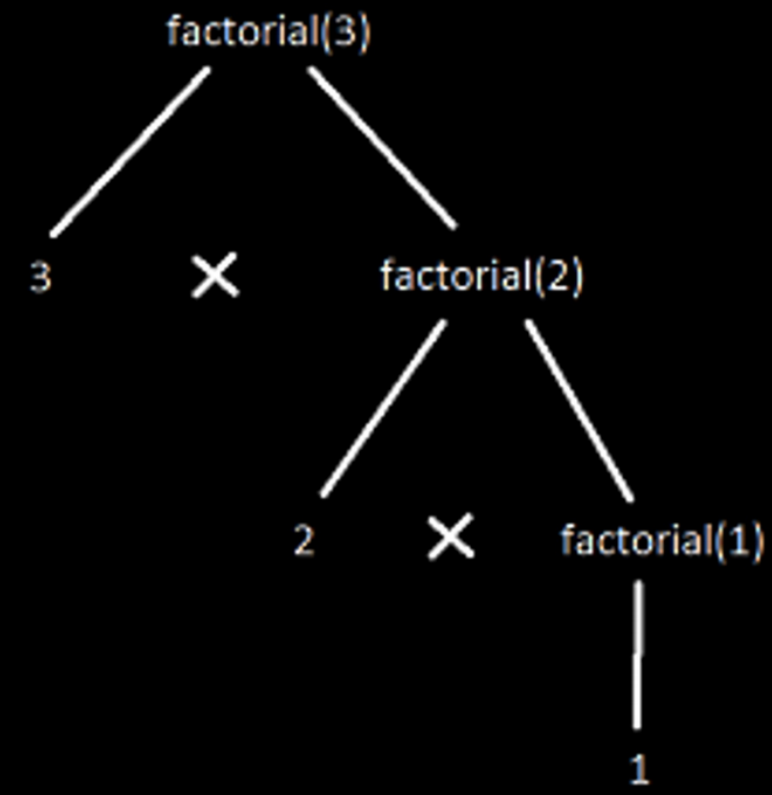
(a)


Activation record for A
Activation record for B
Activation record for C
Activation record for D

(b)

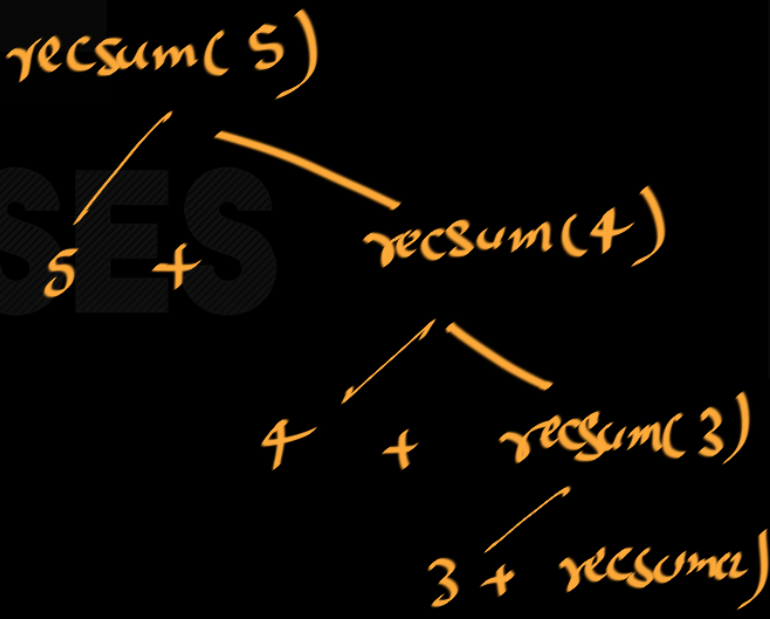
```
factorial(int n)
{
    int fact;

    if (n==1)
        return(1);
    else
        fact = n*factorial(n-1);

    return(fact);
}
```

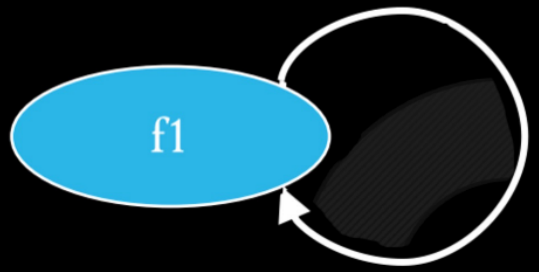
Preferred

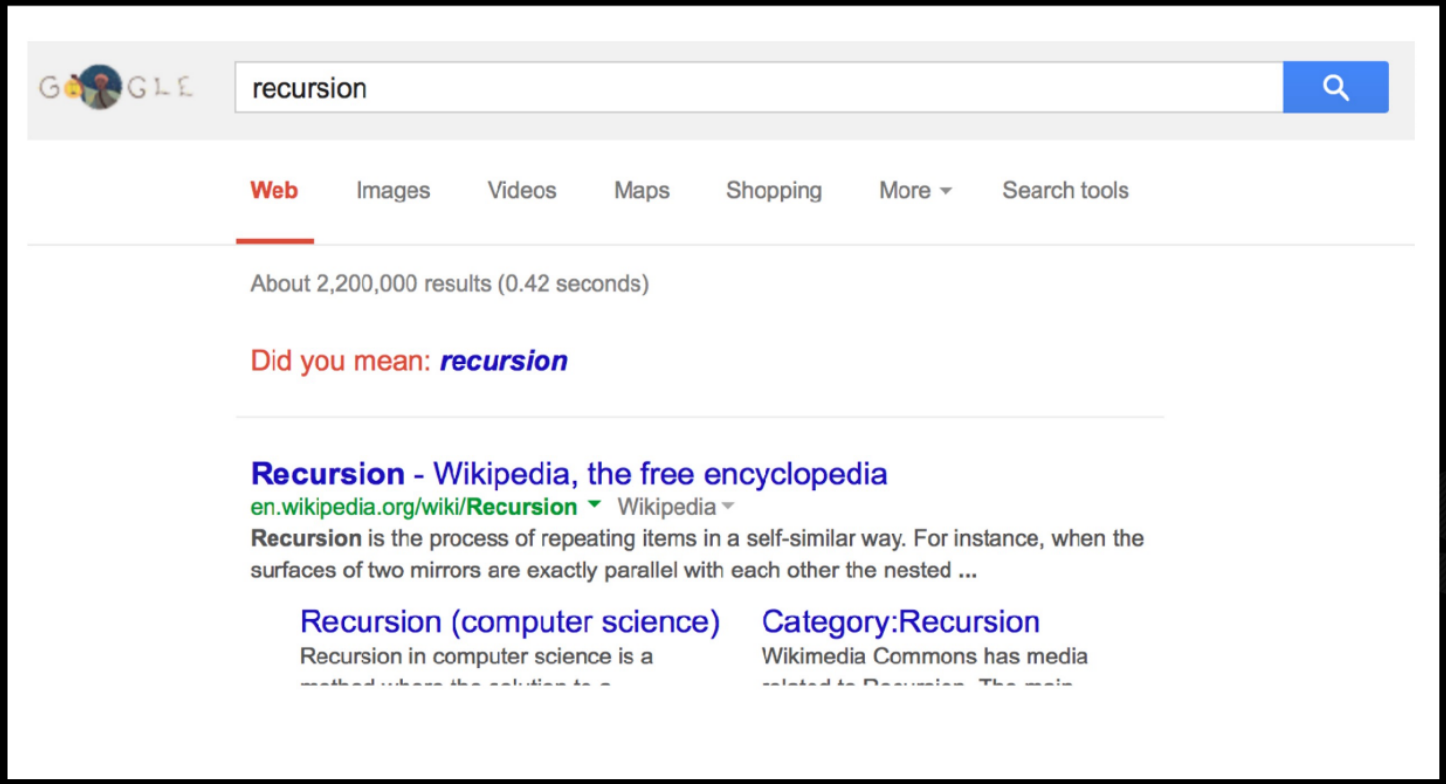


```
int recsum(x) {
    if (x === 1) {
        return x;
    } else {
        return x + recsum(x - 1);
    }
}
```

```
recsum(5)
5 + recsum(4)
5 + (4 + recsum(3))
5 + (4 + (3 + recsum(2)))
5 + (4 + (3 + (2 + recsum(1))))
5 + (4 + (3 + (2 + 1)))
15
```




f1


recursion
Web
Images
Videos
Maps
Shopping
More
Search tools
About 2,200,000 results (0.42 seconds)
Did you mean: recursion
Recursion - Wikipedia, the free encyclopedia
en.wikipedia.org/wiki/Recursion
Wikipedia
Recursion is the process of repeating items in a self-similar way. For instance, when the surfaces of two mirrors are exactly parallel with each other the nested ...
Recursion (computer science)
Recursion in computer science is a
Category:Recursion
Wikimedia Commons has media

Let's recurse on *you*.

How many students total are directly behind you in your "column" of the classroom?

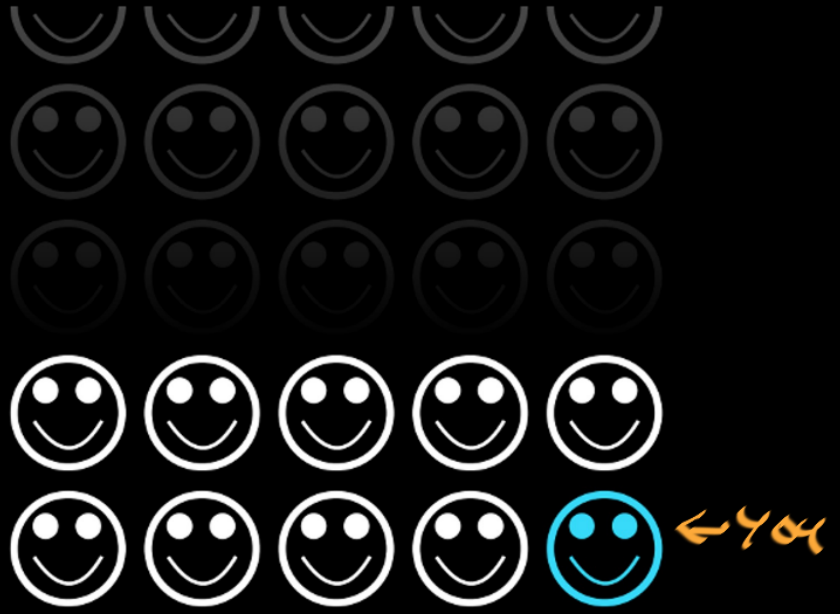
Rules:

1. You can see only the people directly in front and behind you. So, you can't just look back and count.
2. You *are* allowed to ask questions of the people in front / behind you.

How can we solve this problem *recursively*?

Recursive Student Counting

CS220 students in the front row



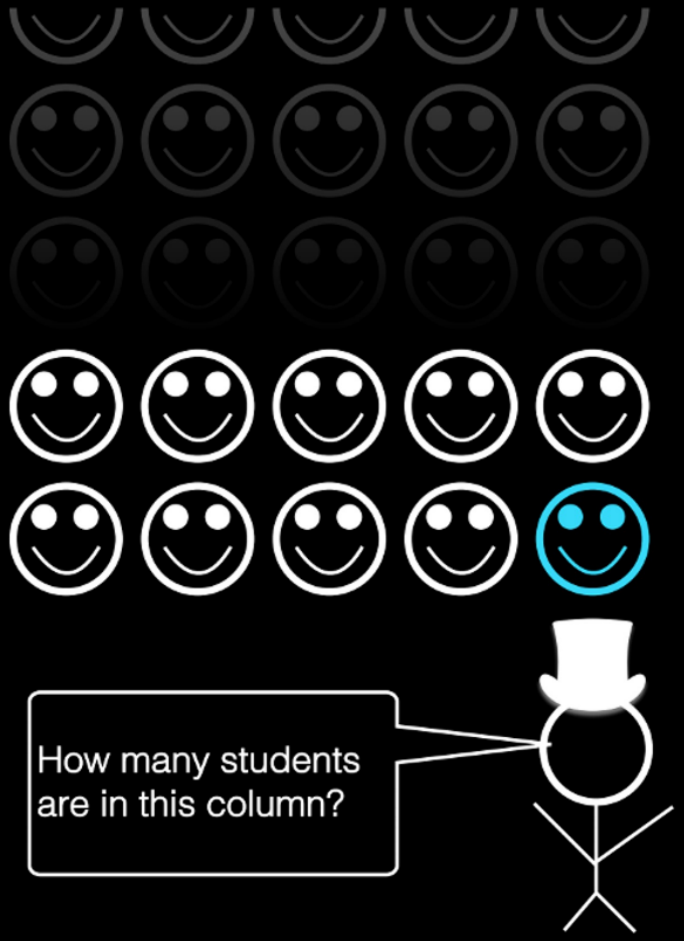
Professor with a question

Recursive Student Counting

Constraints:

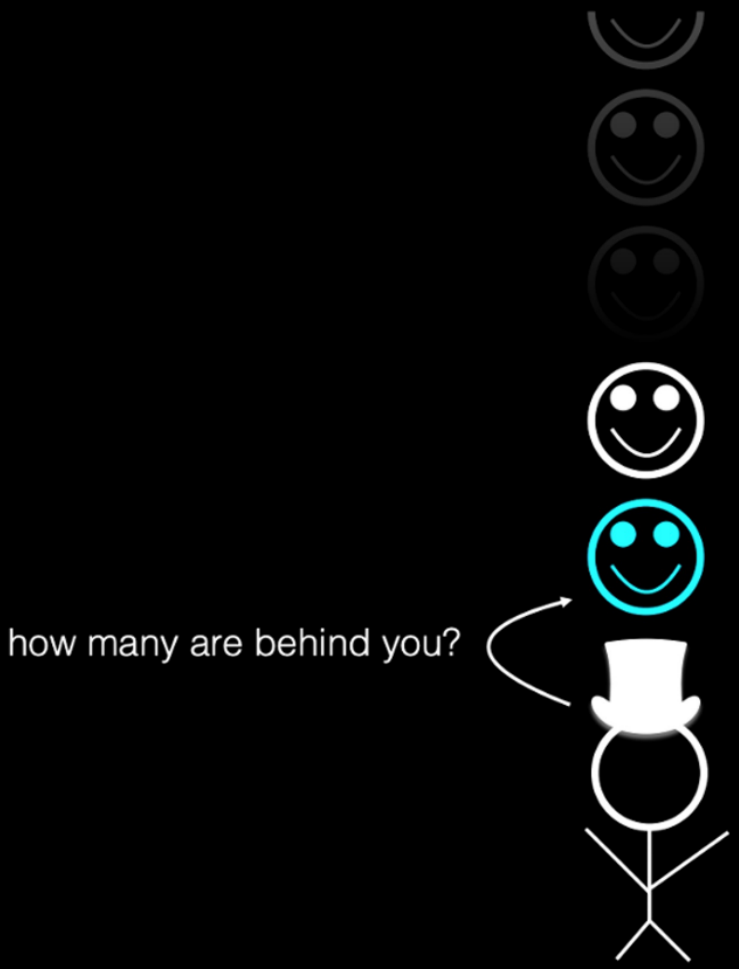
- You can only talk to the student behind / in front of you

What should each student ask the person behind them?



Recursive Student Counting

Strategy: reframe question as "how many students are behind you?"



Process:
if nobody is behind you: say 0
else: ask them, say their answer+1

Recursive Student Counting

Strategy: reframe question as "how many students are behind you?"

Process:
if nobody is behind you: say 0
else: ask them, say their answer+ I

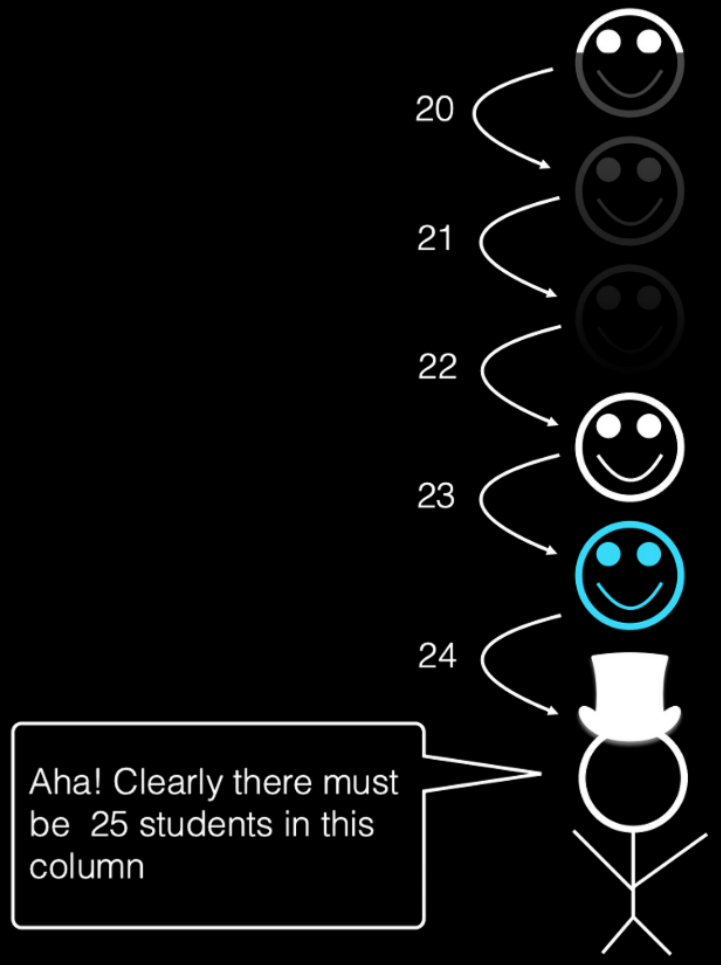
Recursive Student Counting

Strategy: reframe question as "how many students are behind you?"

Process:
if nobody is behind you: say 0
else: ask them, say their answer+ 1

Observations:

- Each student runs the same "code"
- Each student has their own "state"

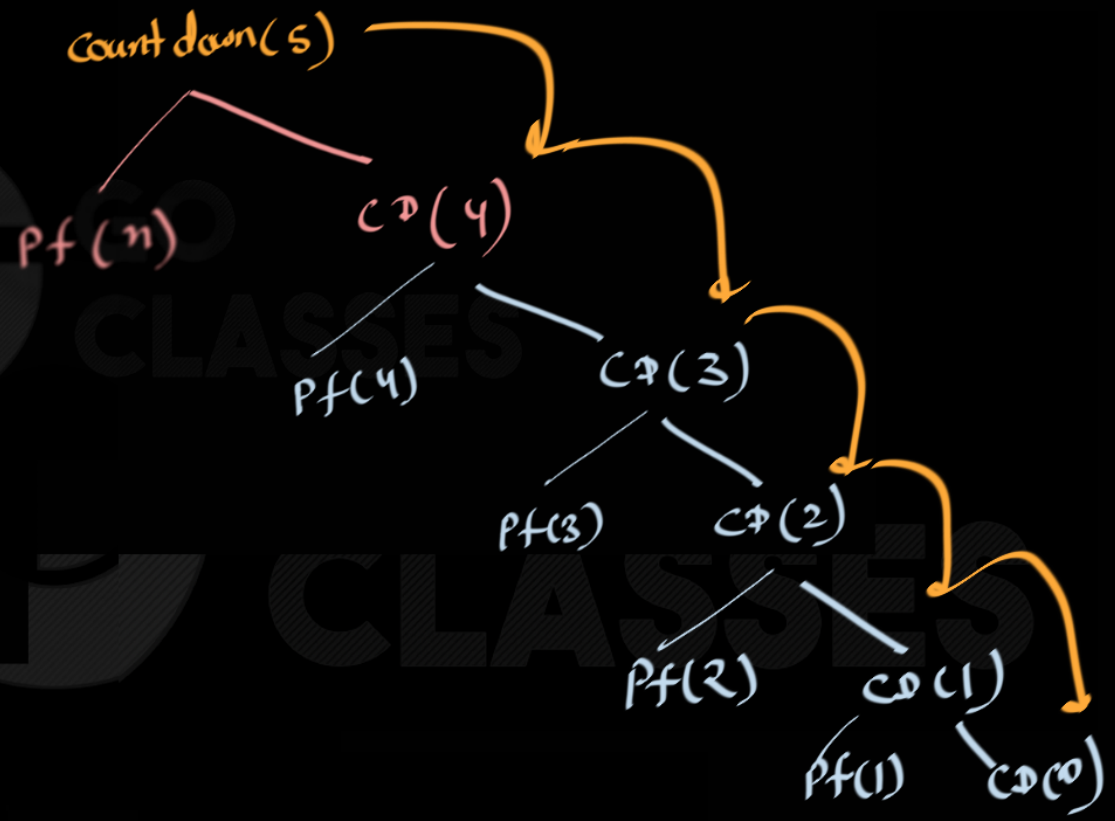


GO CLASSES


Hi, I am Jake
the Time traveller
Oh, really?
then tell me something
that happens in the future
BROOKLYN
99 - HOLTPOSTING
This

Question 1

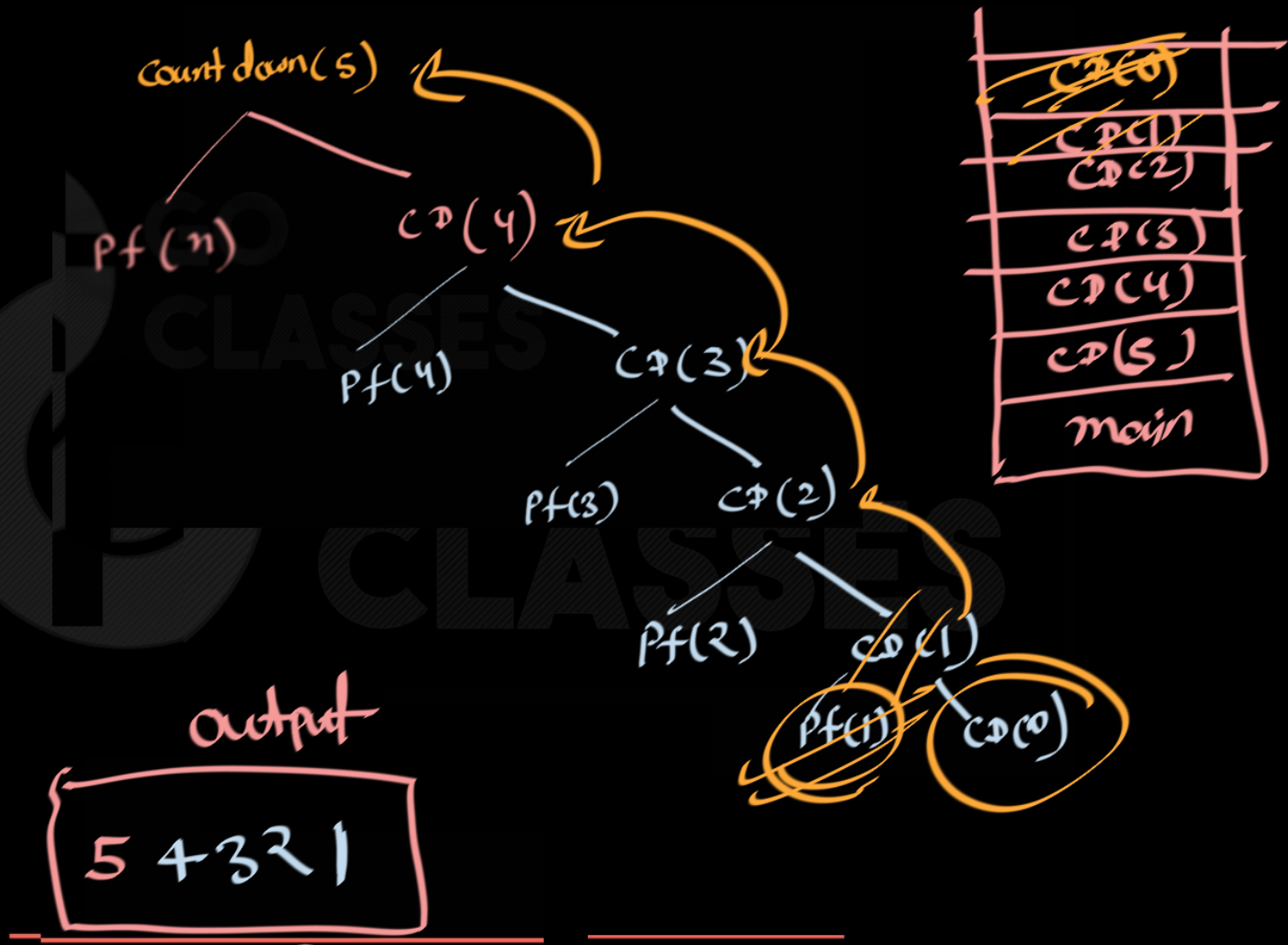
```c
#include<stdio.h>


void countdown(int n){
    if (n==0)
        return;

    printf("%d ", n);
    countdown(n-1);
}


int main()
{
    countdown(5);
}
```



5 4 3 2 1

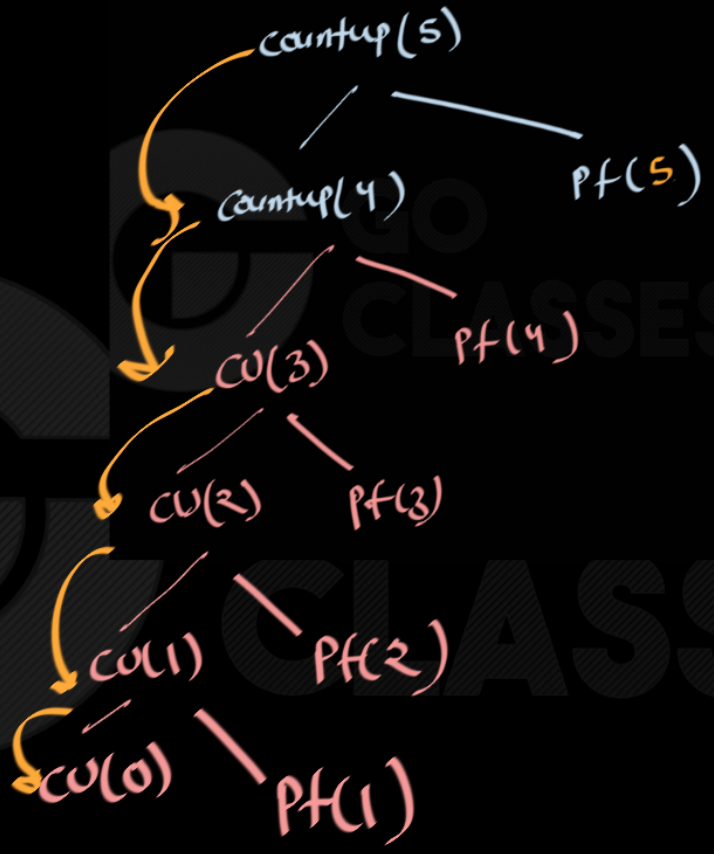
Question 1

```c
#include<stdio.h>


void countdown(int n){
    if (n==0)
        return;

    printf("%d ", n);
    countdown(n-1);
}


int main()
{
    countdown(5);
}
```



Question 2

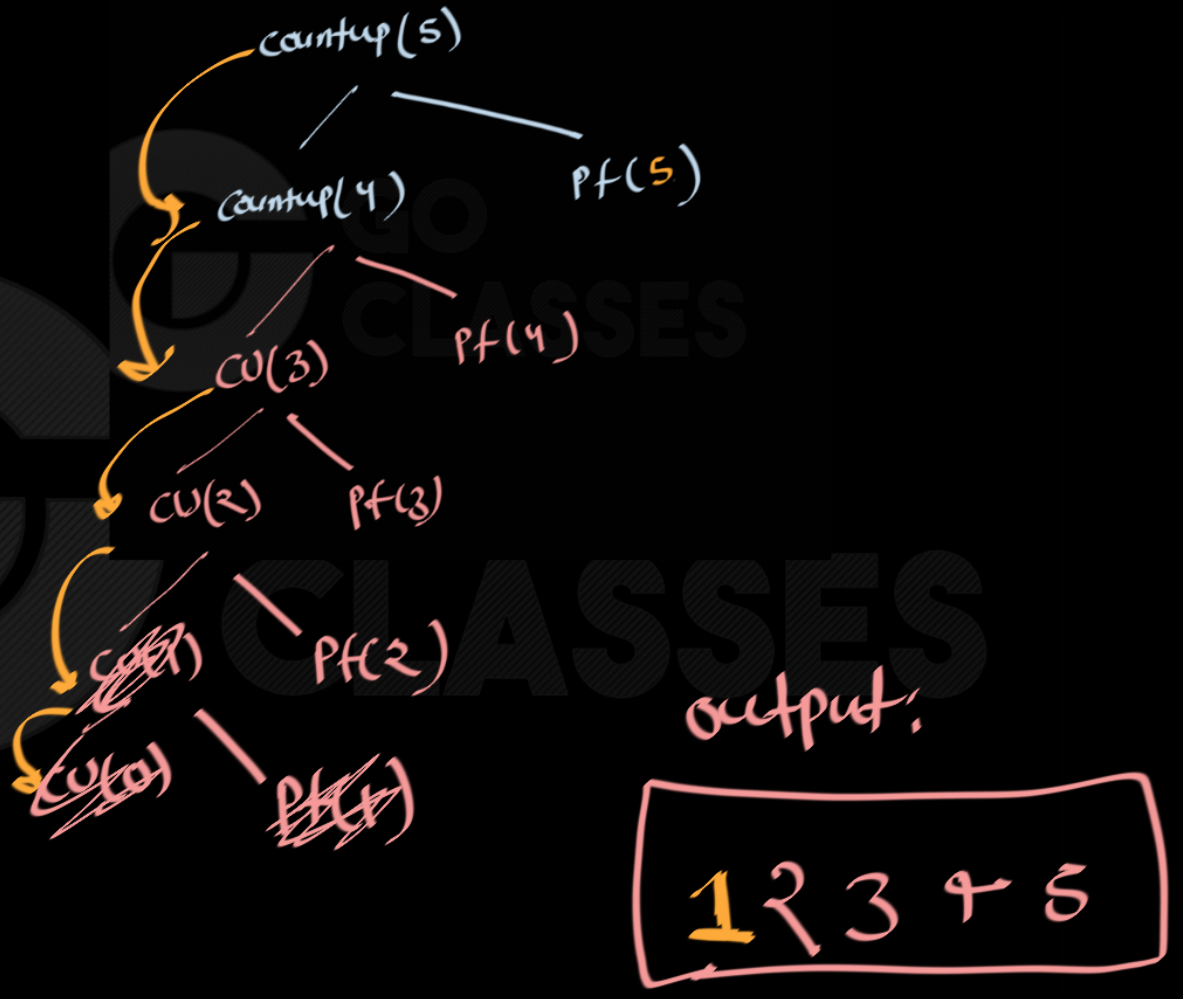
```c
#include<stdio.h>


void countup(int n){
    if (n==0)
        return;

    countup(n-1);
    printf("%d ", n);

}

int main()
{
    countup(5);
}
```



1

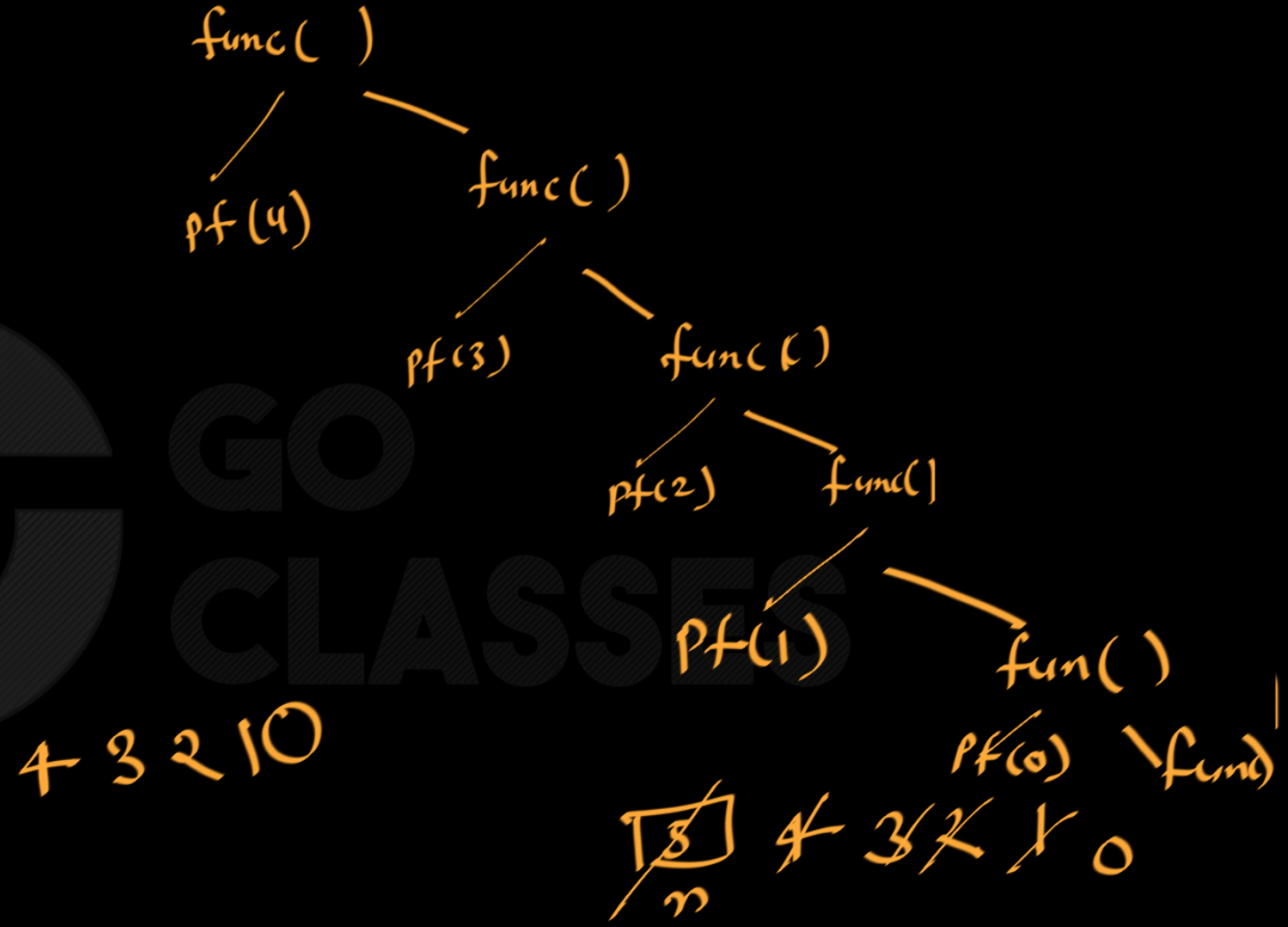
Question 2

```c
#include<stdio.h>


void countup(int n){
    if (n==0)
        return;

    countup(n-1);
    printf("%d ", n);

}

int main()
{
    countup(5);
}
```



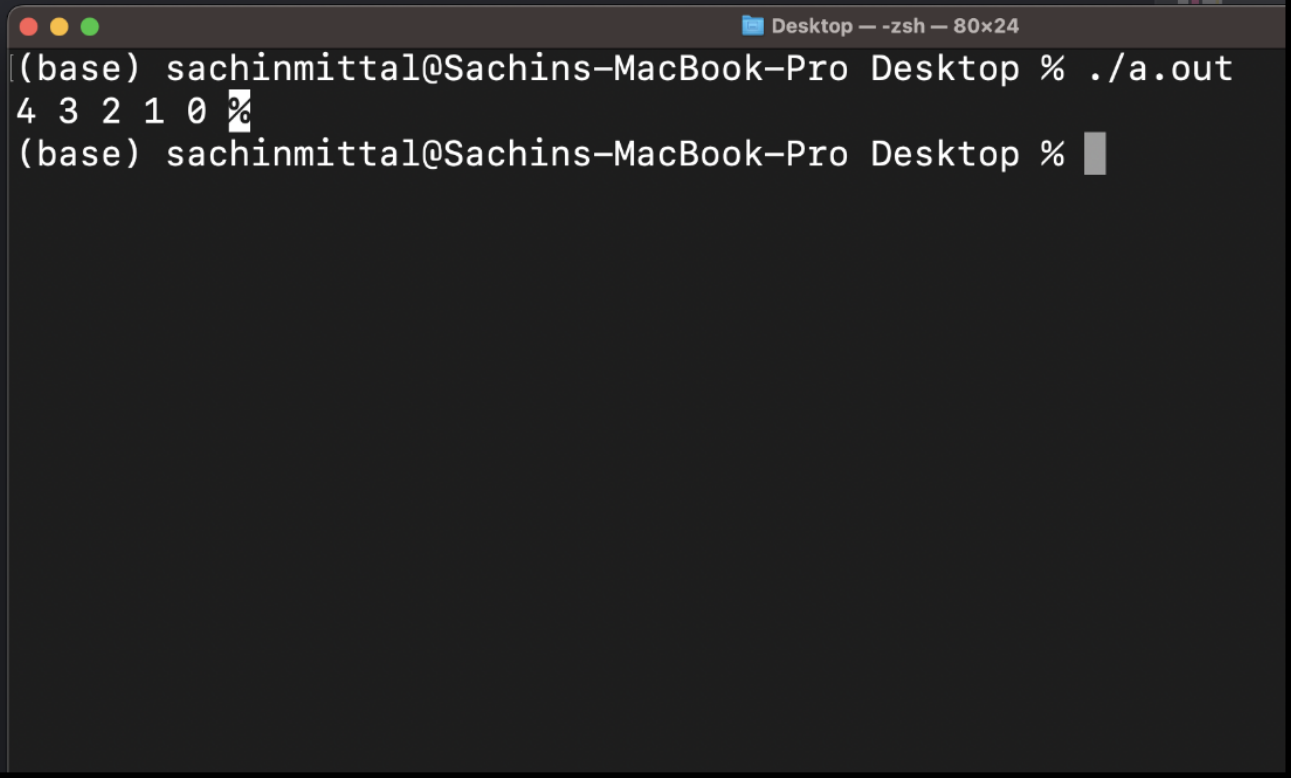
GO CLASSES

```c
#include<stdio.h>

void func(){
    static int n = 5;

    if (n==0)
        return;
    n--;
    printf("%d ", n);
    func();

}

int main()
{
    func();
}
```

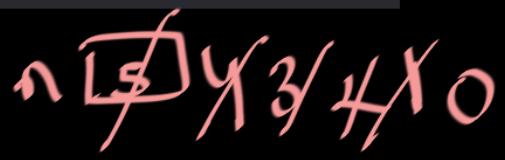


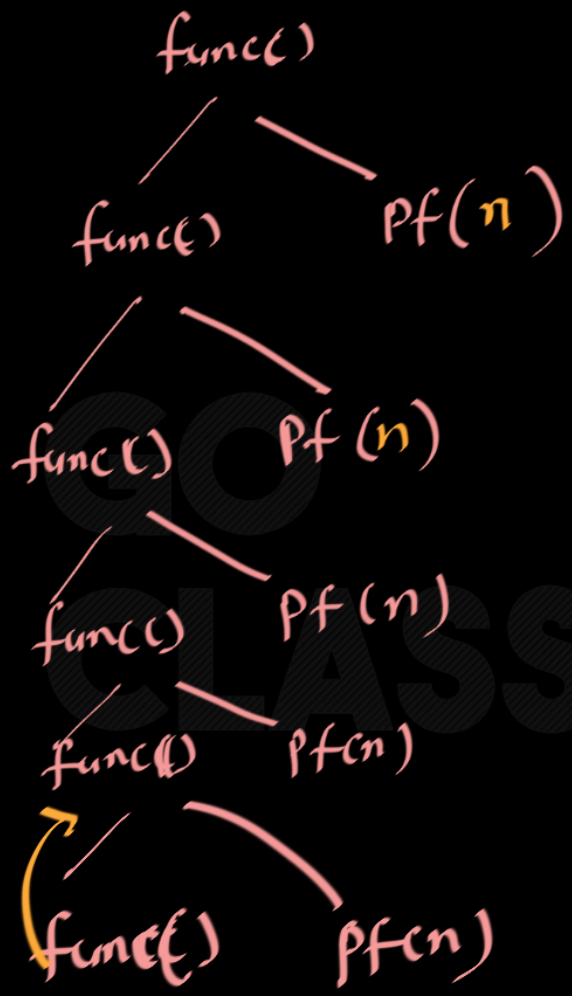
GO CLASSES

```c
#include<stdio.h>

void func(){
    static int n = 5;

    if (n==0)
        return;
    n--;
    printf("%d ", n);
    func();

}

int main()
{
    func();
}
```



```c
#include<stdio.h>

void func(){
    static int n = 5;

    if (n==0)
        return;
    n--;
    func();
    printf("%d ", n);


}

int main()
{
    func();
}
```





Question 3

```c
int recursiveFunc(int n)
{
    static int count = 0;

    if (n <= 0)
        return count;

    count++;
    return recursiveFunc(n - 1);
}

int main()
{
    int result = recursiveFunc(5);
    printf("Result: %d\n", result);

}
```

What will be the output of the above code?

A) Result: 0

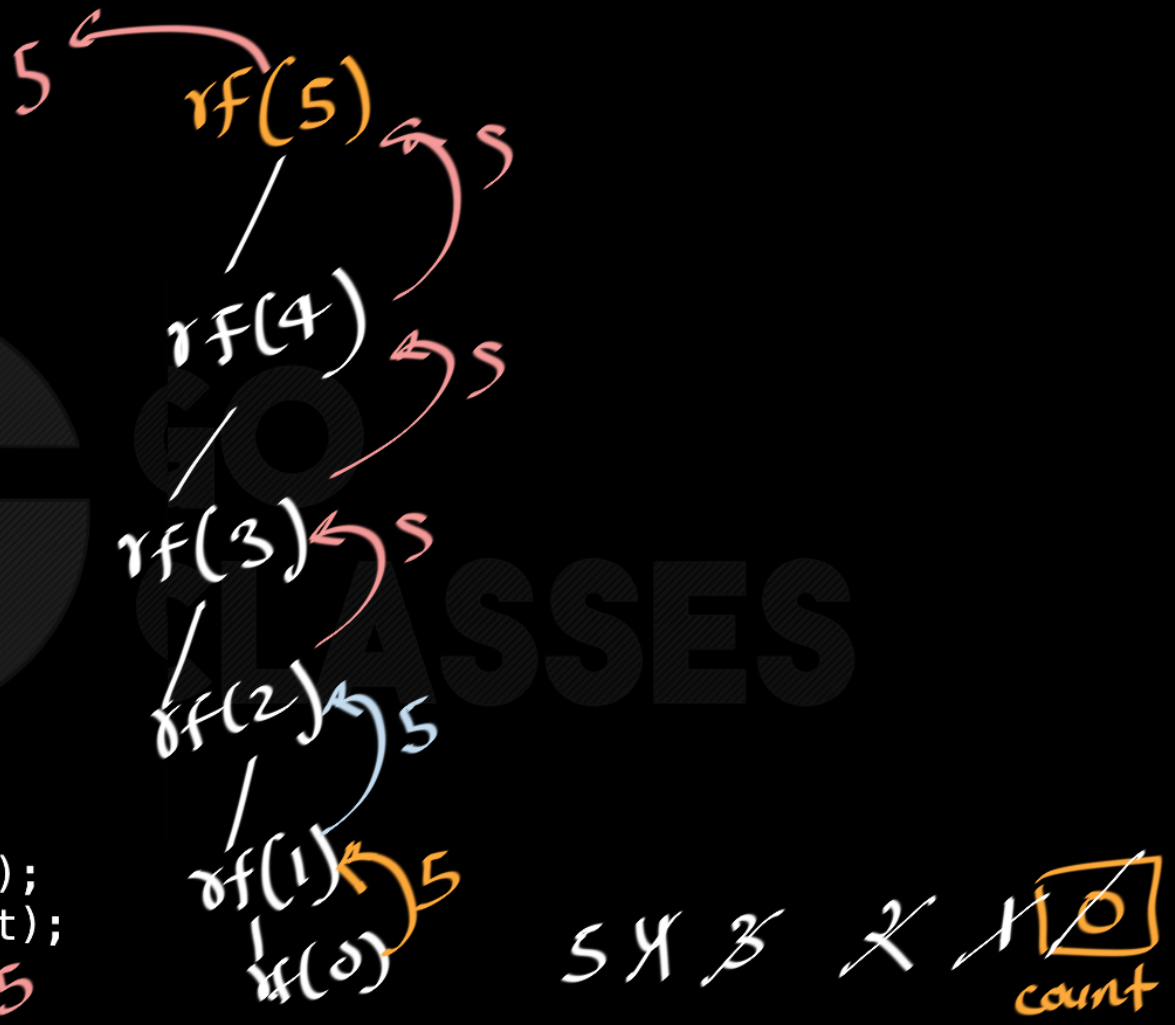
B) Result: 1

C) Result: 4

D) Result: 5

Question 3

```
int recursiveFunc(int n)
{
    static int count = 0;

    if (n <= 0)
        return count;

    count++;
    return recursiveFunc(n - 1);
}

int main()
{
    int result = recursiveFunc(5);
    printf("Result: %d\n", result);

}
```



Question 4

```c
int recursiveFunc(int n)
{
    static int count = 0;

    if (n <= 0)
        return count;

    count += n;
    return recursiveFunc(n - 1);
}


int main()
{
    int result = recursiveFunc(3);
    printf("Result: %d\n", result);
}
```

What will be the output of the above code?

A) Result: 0

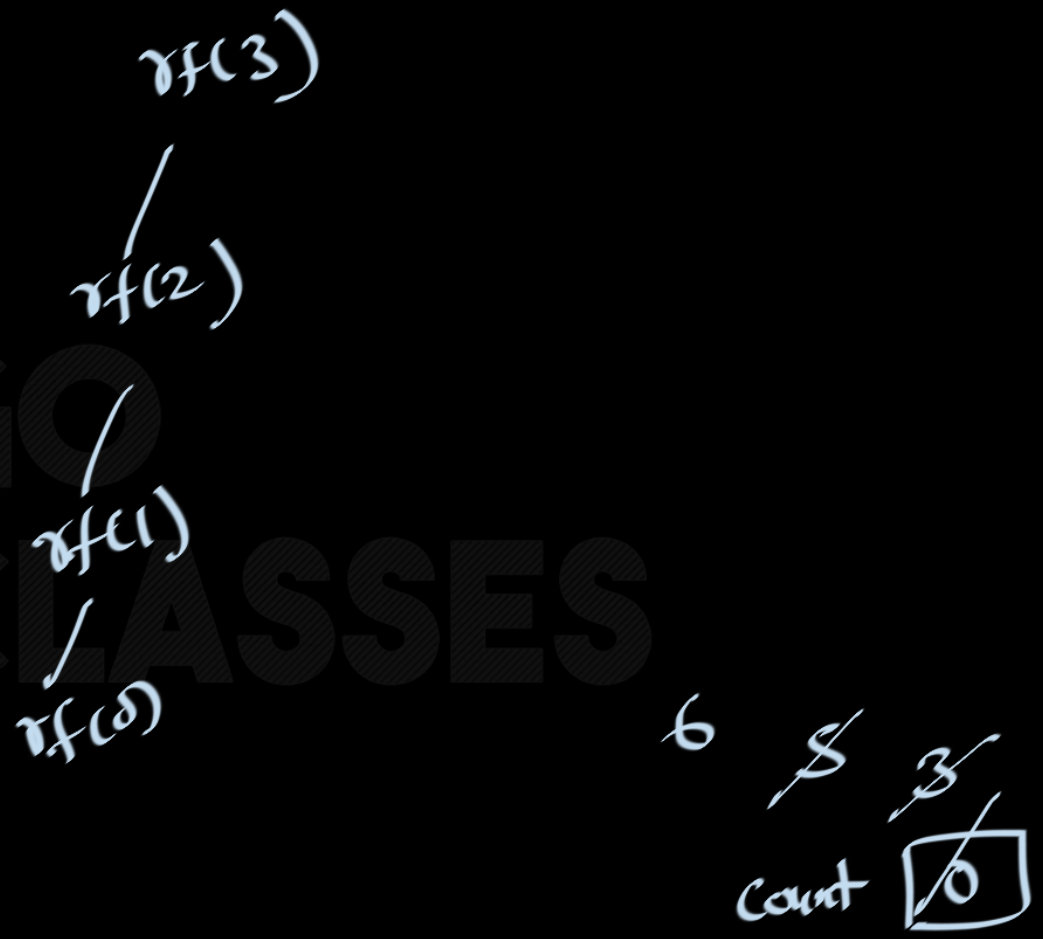
B) Result: 3

C) Result: 6

D) Result: 10

Question 4

```
int recursiveFunc(int n)
{
    static int count = 0;

    if (n <= 0)
        return count;

    count += n;
    return recursiveFunc(n - 1);
}


int main()
{
    int result = recursiveFunc(3);
    printf("Result: %d\n", result);
}
```



Question 5

H.W.

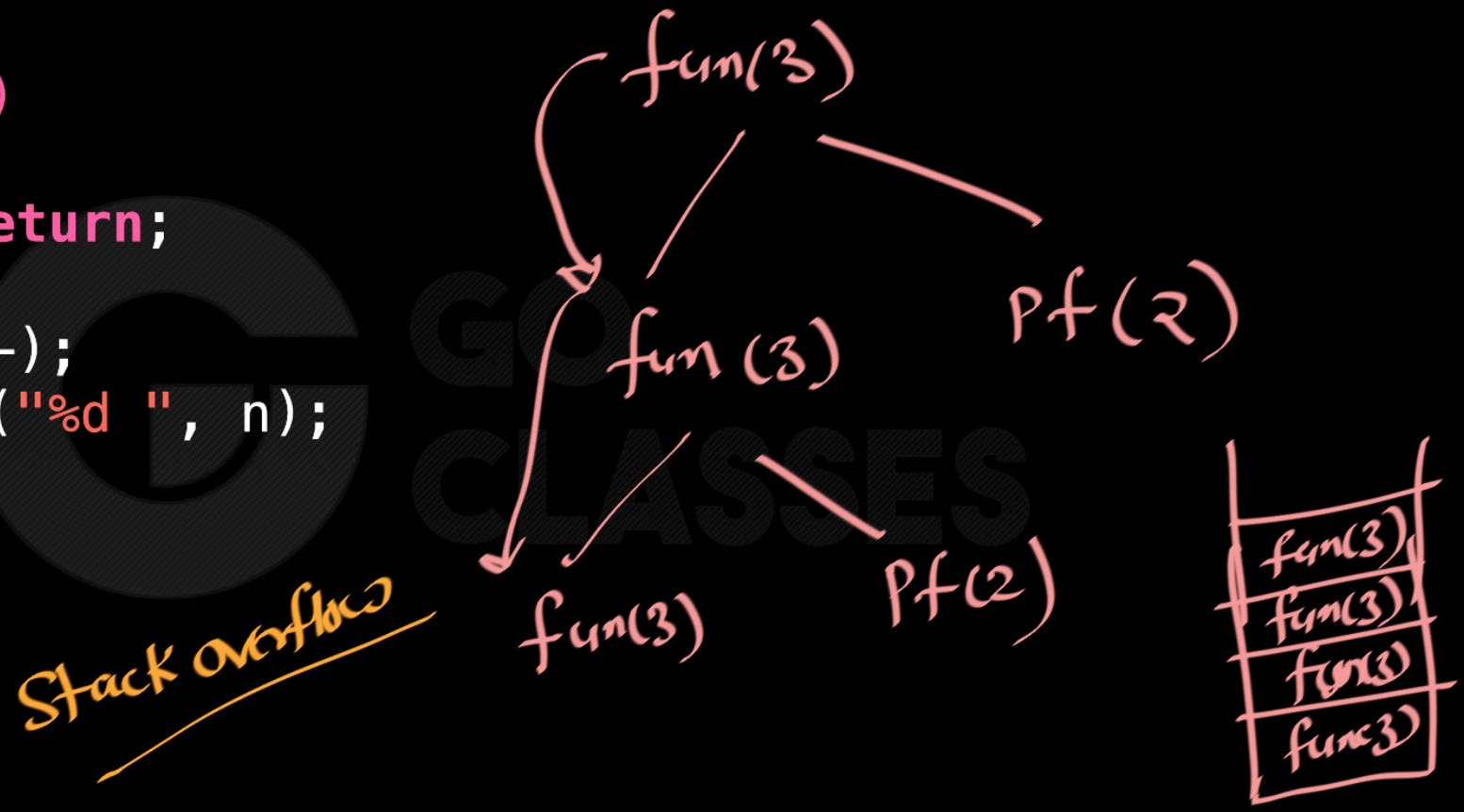
```c
#include <stdio.h>
void increase(int x)
{
    static int i = 0;
    x = i + x;
    i++;
    printf("x=%d, i=%d\n", x, i);
}

int main(void)
{
    int i = 0;
    for (; i < 4; ++i)
    {
        increase(i);
    }
}
```

Question 6

```
int func(int a)
{
    static int j=0;
    j++;
    return a*j;
}

int main(void)
{
    int x=10;

    while(x>0){
        printf("%d\n",func(x));
        --x;
    }

    return (0);
}
```

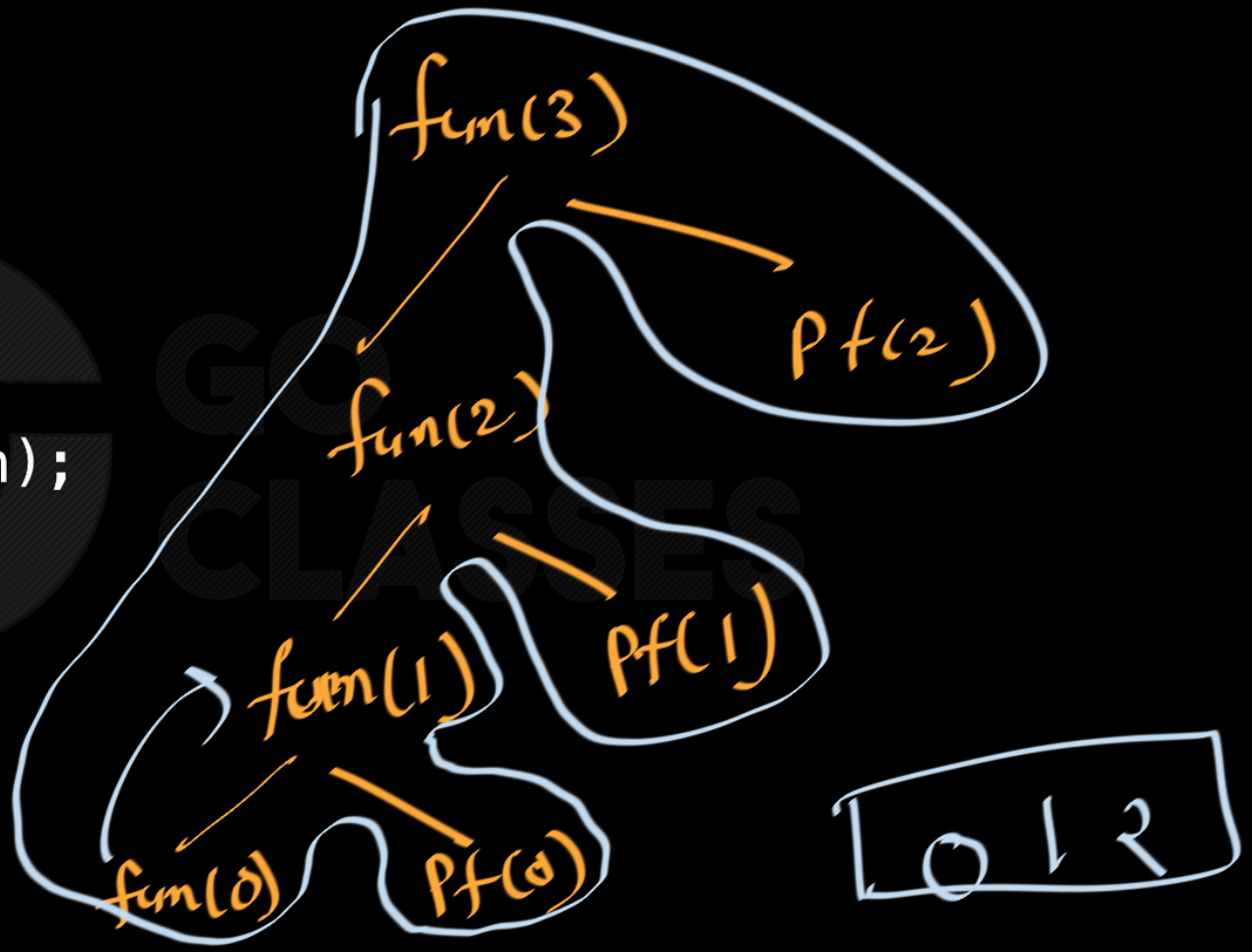
Question 7

```
void fun(int n)
{
    if(n==0) return;
    else {
        fun(n--);
        printf("%d ", n);
    }
}

main()
{
    fun(3);
}
```



Question 8

```
void fun(int n)
{
    if(n==0) return;
    else {
        fun(--n);
        printf("%d ", n);
    }
}

 main()
{
    fun(3);
}
```



Question 9-12

Program 1

```
void recurse() {
    static int i = 4;
    if (--i) {
        recurse();
        printf("%d", i);
    }
}
```

Program 2

```
void recurse() {
    static int i = 4;
    printf("%d", i);
    if (--i) {
        recurse();
    }
}
```

Program 3

```
void recurse(int i) {
    if (--i) {
        printf("%d", i);
        recurse(i);
    }
}
```

Program 4

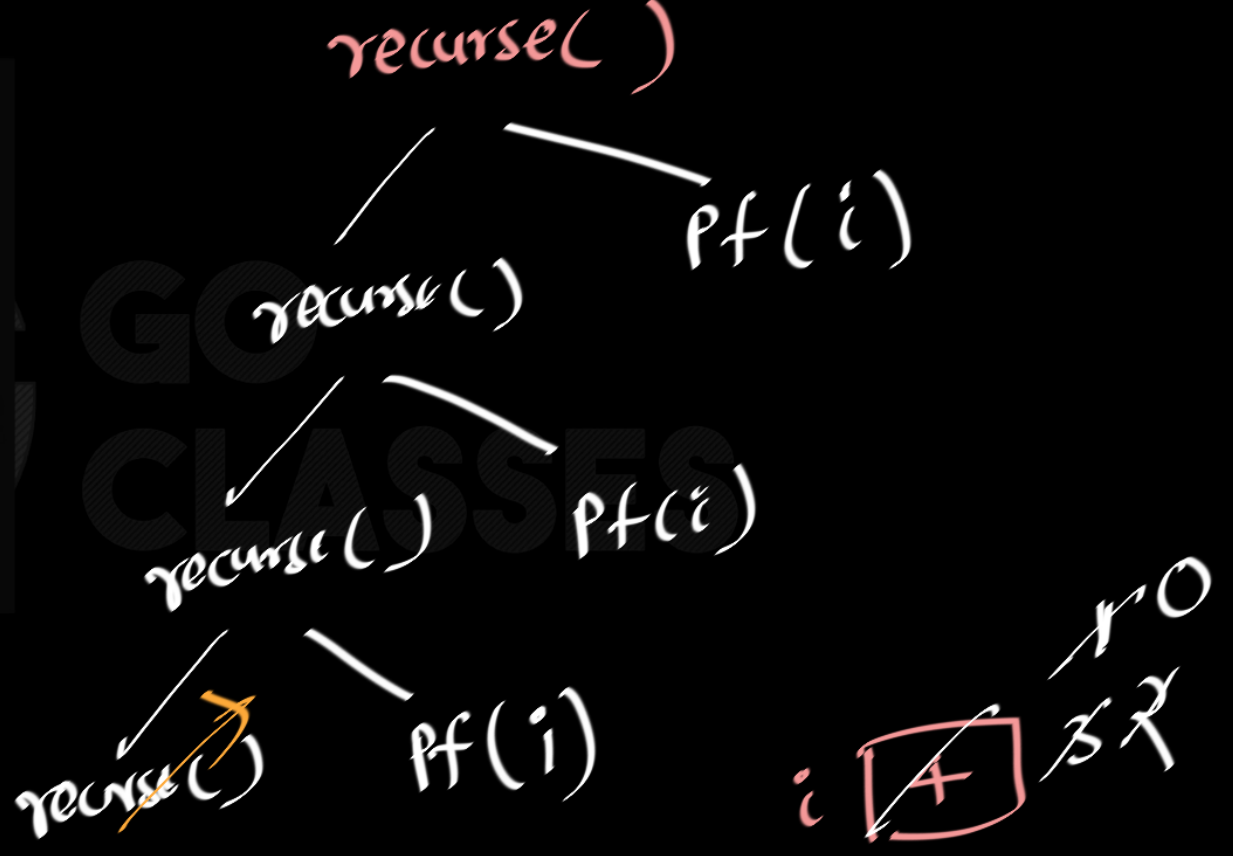
```
void recurse(int i) {
    printf("%d", i);
    if (--i) {
        recurse(i);
    }
}
```

Program 1

```
void recurse() {
    static int i = 4;
    if (--i) {
        recurse();
        printf("%d", i);
    }
}
```

output:

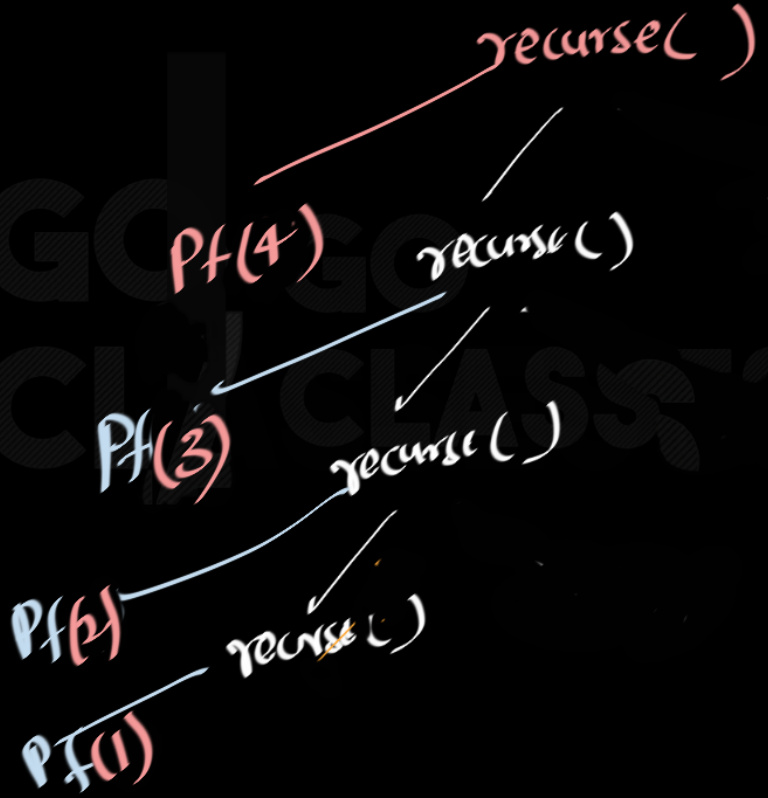
000

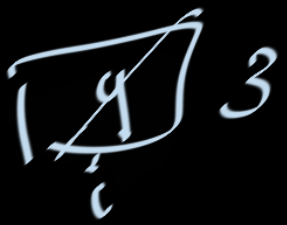


Program 2

```
void recurse() {
    static int i = 4;
    printf("%d", i);
    if (--i) {
        recurse();
    }
}
```

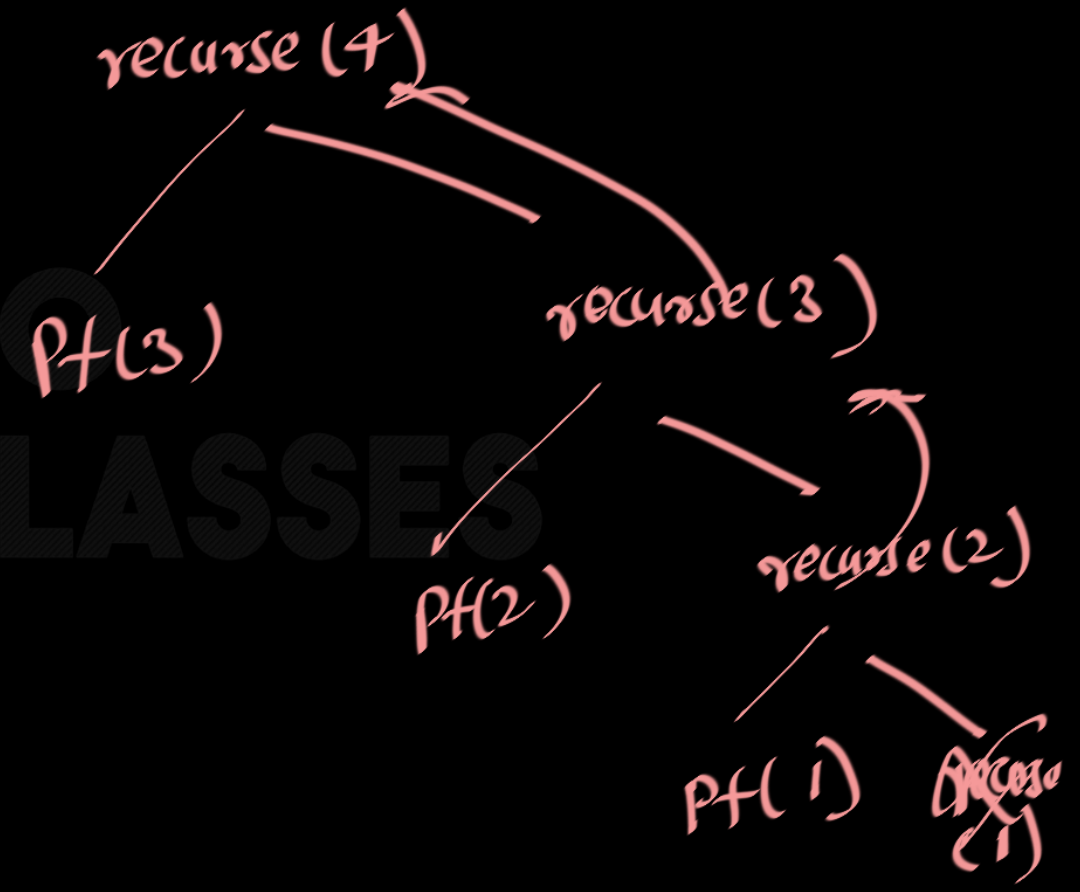
4 3 2 1





Program 3

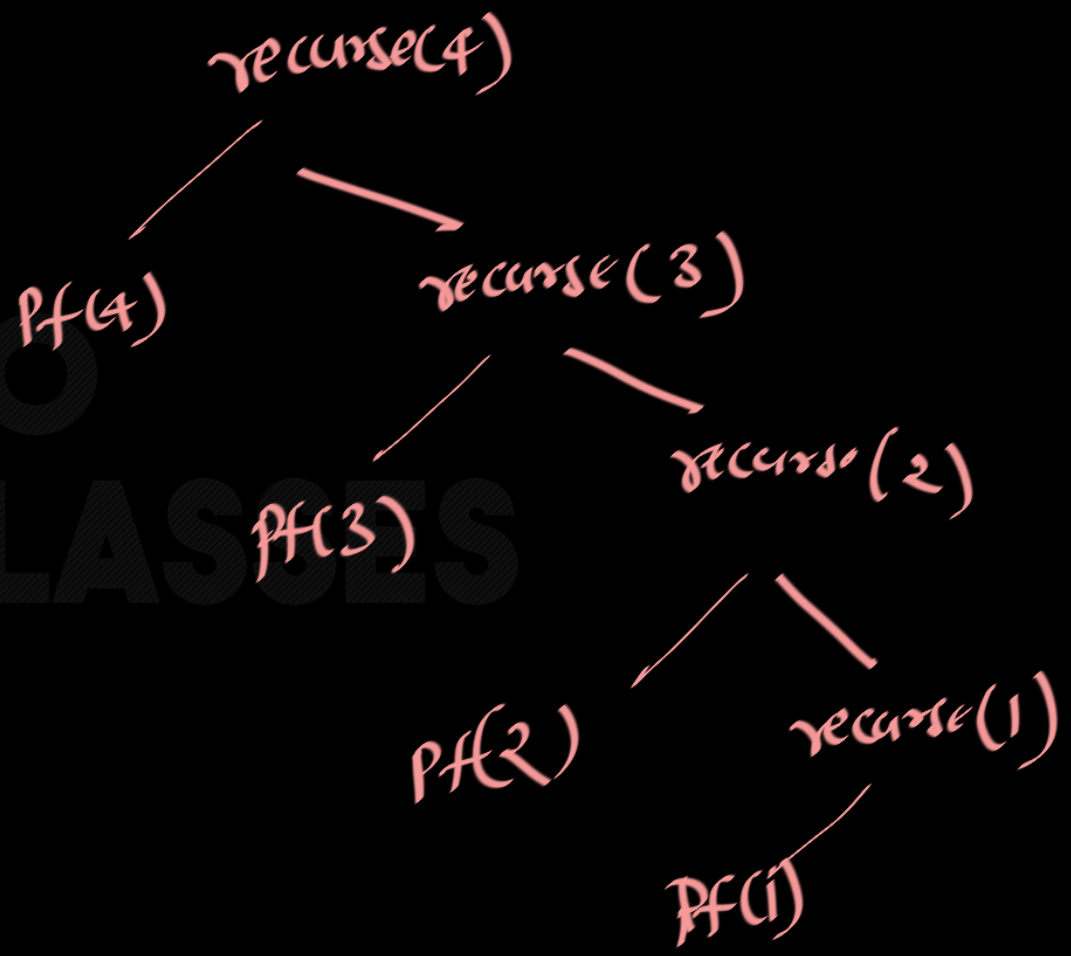
```
void recurse(int i) {
    if (--i) {
        printf("%d", i);
        recurse(i);
    }
}
```



Program 4

```
void recurse(int i) {
    printf("%d", i);
    if (--i) {
        recurse(i);
    }
}
```

4321



Program 1

000

```
void recurse() {
    static int i = 4;
    if (--i) {
        recurse();
        printf("%d", i);
    }
}
```

Program 2

4321

```
void recurse() {
    static int i = 4;
    printf("%d", i);
    if (--i) {
        recurse();
    }
}
```

Program 3

321

```
void recurse(int i) {
    if (--i) {
        printf("%d", i);
        recurse(i);
    }
}
```

Program 4

4321

```
void recurse(int i) {
    printf("%d", i);
    if (--i) {
        recurse(i);
    }
}
```

Question 13

```c
double power(double x, int n)
{
    if (n == 1)
        return x;
    else
        return x * power(x, n - 1);
    --n;
}

int main()
{
    int a = power(2, 5);
    printf("%d", a);
}
```

http://web.mit.edu/10.001/Web/Homework/Solutions/10_001F02midterm_soln.pdf

Question 14

```c
#include <stdio.h>
void func2(int n);  // Forward declaration
void func1(int n)
{
    if (n > 0)
    {
        printf("func1: %d\n", n);
        func2(n - 1);
    }
}
void func2(int n)
{
    if (n > 0)
    {
        printf("func2: %d\n", n);
        func1(n - 1);
    }
}
int main()
{
    func1(3);
    return 0;
}
```

Question 15

```c
#include <stdio.h>
void r();
int i = 20;

int main() {
    int i = 5;
    printf("%d ", i);
    r();
    printf("%d ", i);
    r();
    return 0;
}

void r() {
    static int j;
    if (i < 10) {
        i = 3;
        printf("%d ", i);
    }
    printf("%d ", i + j);
    i++;
    j++;
}
```