7 - Recursion

Sunday, 9 August 2026 14:30

A function that repeatedly calls itself.

eg: Factorial

$$n! = n\,(n-1)\,(n-2)\,\ldots\,(1)$$

$$\Rightarrow n! = n\,(n-1)!$$

$$\Rightarrow fact(n) = n \cdot fact(n-1)$$

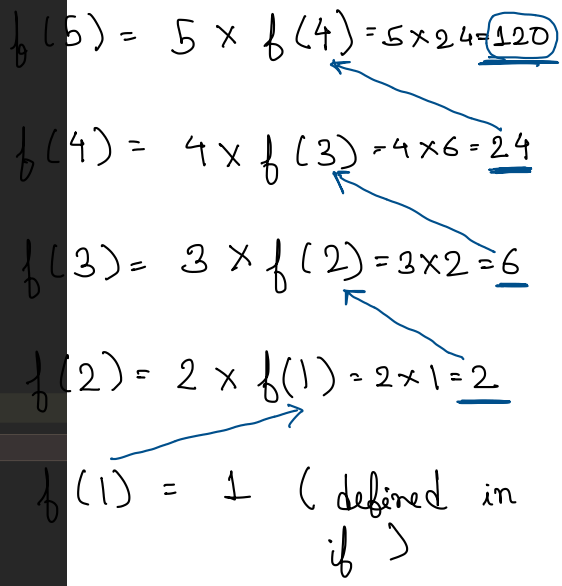
recursion/factorial.c

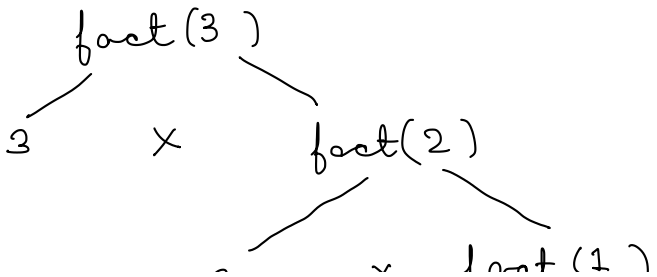
```c
#include <stdio.h>

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

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

```

recursion — zsh

```
(base) arneshbanerjee@Arneshs-MacBook-Air recursion % gcc factorial.c
(base) arneshbanerjee@Arneshs-MacBook-Air recursion % ./a.out
120%
(base) arneshbanerjee@Arneshs-MacBook-Air recursion %
```

Visual Rep





2 ... 1

Questions

(i)

recursion/countup.c › void countup( )

```c
#include<stdio.h>

void countup(int n){
    if(n==0){
        return;
    }
    countup(n-1);
    printf("%d", n);
}


int main(){
    countup(5);
    return 0;
}
```

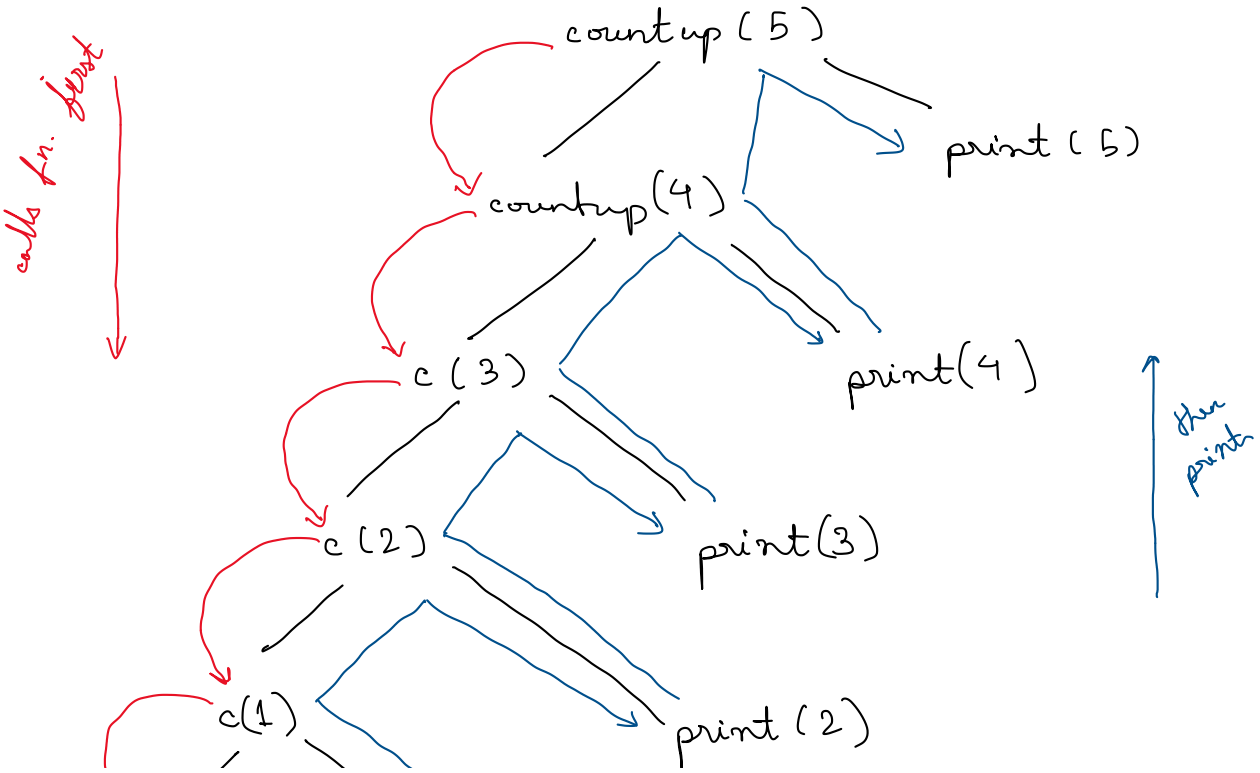
recursion — zsh

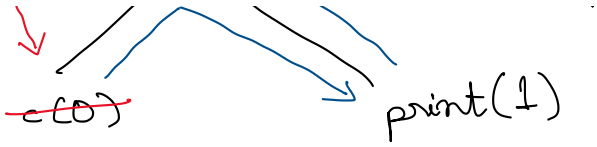
```
(base) arneshbanerjee@Arneshs-MacBook-Air recursion % gcc countup.c
(base) arneshbanerjee@Arneshs-MacBook-Air recursion % ./a.out
12345%
(base) arneshbanerjee@Arneshs-MacBook-Air recursion %
```

The recursion gets called first & then print.

So the print only happens once all the recursions are over

And since last countup(0) finishes first, the countup(1) print happens first.





o/p : 1 2 3 4 5

(ii)

```c
#include<stdio.h>

void func(){
    static int n = 5;

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


}

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

o/p → 0 0 0 0 0

∴ at last step n = 0.

And since n is a static variable.

So n = 0 persists for all print values.

(iii)

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

    if (n <= 0)
        return count;

    count++;
```

rF(5)
/
rF(4)
/
rF(3)
/

```
    return recursiveFunc(n - 1);
}

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

}
```

rf(2)
/
rf(1)
/
rf(0)
5

(iv)

```
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);
}
```

rF(3) c = 3
/
rf(2) c = 5
/
rf(1) c = 6
/
rF(0) c = 6
↓
returns count

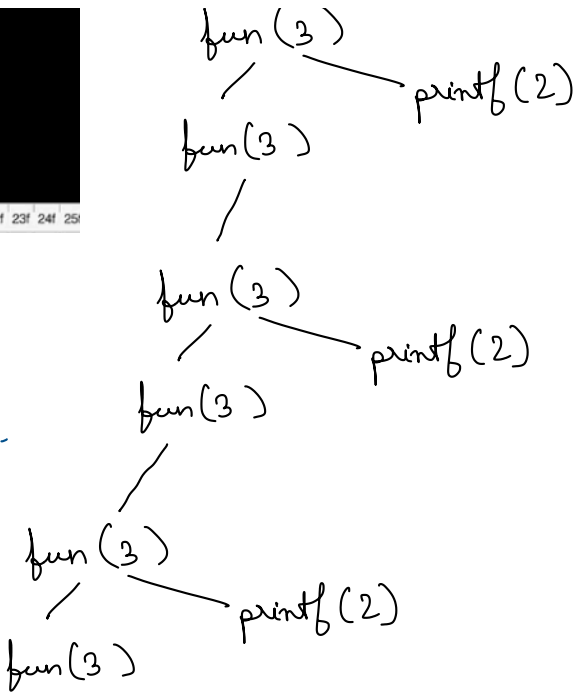
(v)

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

Start with fun(3)

→ fun(3) get passed again since it's post decrement operator

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



So this runs infinitely & results in stack overflow.

(vi)

```
#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;
}
```

3

func1: 3
func2: 2
func1: 1

(vi) Gate CSE 2004

```
int f(int n)
{
    static int i = 1;
```

• f(1)
n = 1, i = 1 → n = 2, i = 2

• f(n) → n = 2, i = 2 → n = 4, i = 2

```
    if(n >= 5) return n;
    n = n+i;
    i++;
    return f(n);
}
```

f(2) -> n = 2, i = 2 -> n = 4, i = 3

- f(4) => n = 4, i = 3 → n = 7

↓

return(n)

Ans) 7

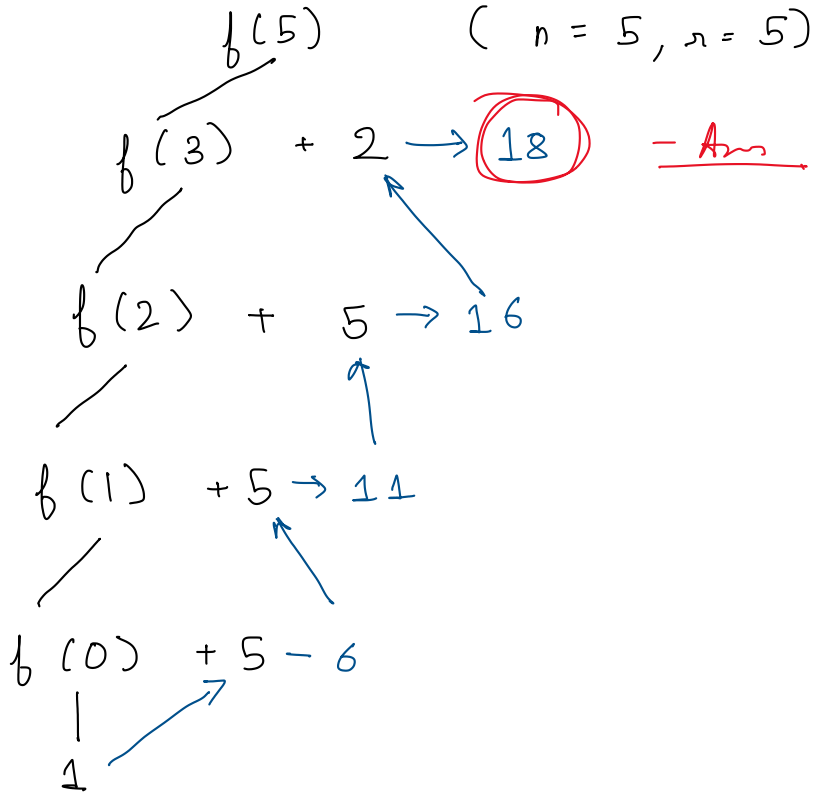
Q: What will be value returned by f(1).

(viii)

```
int f(int n)
{
    static int r = 0;
    if (n <= 0) return 1;
    if (n > 3)
    {   r = n;
        return f(n-2) + 2;
    }
    return f(n-1) + r;
}
```

What is the value of $f(5)$?

A. 5
B. 7
C. 9
D. 18



viii

```
void count (int n) {
    static int d=1;

    printf ("%d",n);
    printf ("%d",d);
    d++;
    if (n>1) count (n-1);
    printf ("%d",d);


}

void main(){
    count (3);
}
```

A. 312213444
B. 312111222
C. 3122134
D. 3121112

3 1 2 2 1 3 4 4 4

ix

Gate CS 2017

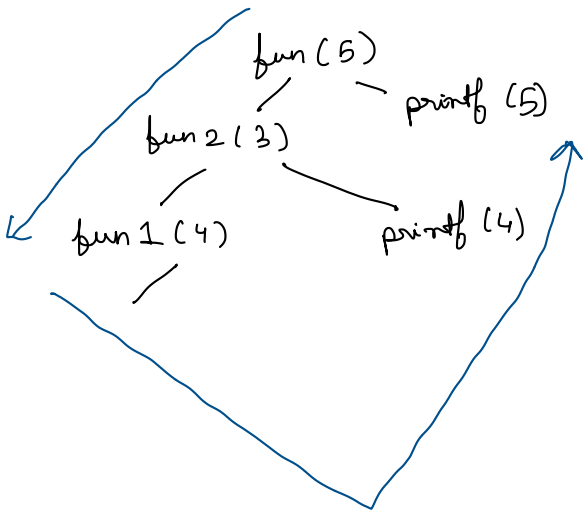
Consider the following two functions.

```
void fun1(int n) {
    if(n == 0) return;
    printf("%d", n);
    fun2(n - 2);
    printf("%d", n);
}
void fun2(int n) {
    if(n == 0) return;
    printf("%d", n);
    fun1(++n);
    printf("%d", n);
}
```

The output printed when fun1(5) is called is

A. 53423122233445
B. 53423120112233
C. 53423122132435

5 3 4 2 3 1 2 2 2 3 3 4 4 5



So last 2 digits are '45'

So answer is (A)