Recursion Pseudocode

Recursion means a function calls itself.

Required parts

Every recursive function needs:

1. Base case: when recursion stops.
2. Recursive case: function calls itself with smaller/simpler input.

Factorial

```
FUNCTION fact(n)
    IF n = 0 THEN
        RETURN 1
    ELSE
        RETURN n * fact(n-1)
    END IF
END FUNCTION
DISPLAY fact(4)
```

Trace:

```
fact(4) = 4 * fact(3)
fact(3) = 3 * fact(2)
fact(2) = 2 * fact(1)
fact(1) = 1 * fact(0)
fact(0) = 1
```

Output:

```
24
```

Fibonacci

```
FUNCTION fib(n)
    IF n = 0 THEN RETURN 0
    IF n = 1 THEN RETURN 1
    RETURN fib(n-1) + fib(n-2)
END FUNCTION
```

Values:

```
fib(0)=0
fib(1)=1
fib(2)=1
fib(3)=2
fib(4)=3
fib(5)=5
fib(6)=8
```

Sum recursion

```
FUNCTION sum(n)
    IF n = 0 THEN RETURN 0
    RETURN n + sum(n-1)
END FUNCTION
```

`sum(5) = 5+4+3+2+1+0 = 15`

Recursive output order

Print before recursive call

```
FUNCTION fun(n)
    IF n = 0 THEN RETURN
    DISPLAY n
    fun(n-1)
END FUNCTION
fun(3)
```

Output:

```
3 2 1
```

Print after recursive call

```
FUNCTION fun(n)
    IF n = 0 THEN RETURN
    fun(n-1)
    DISPLAY n
END FUNCTION
fun(3)
```

Output:

```
1 2 3
```

Common recursion traps

- Missing base case causes infinite recursion.
- Print before call gives descending order.
- Print after call gives ascending order.
- Recursive functions use stack memory.
- `fib(n)` without memoization repeats many calls.