RECURSION

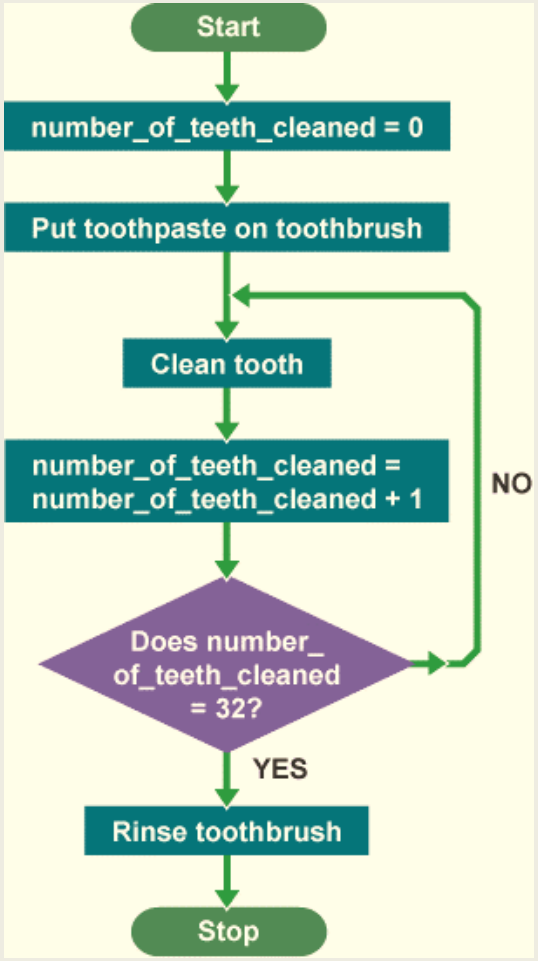
Mahsa Ziraksima
ACIBADEM UNIVERSITY
Fall 2025

SUMMARY

- Iterative algorithm
- Recursive algorithm

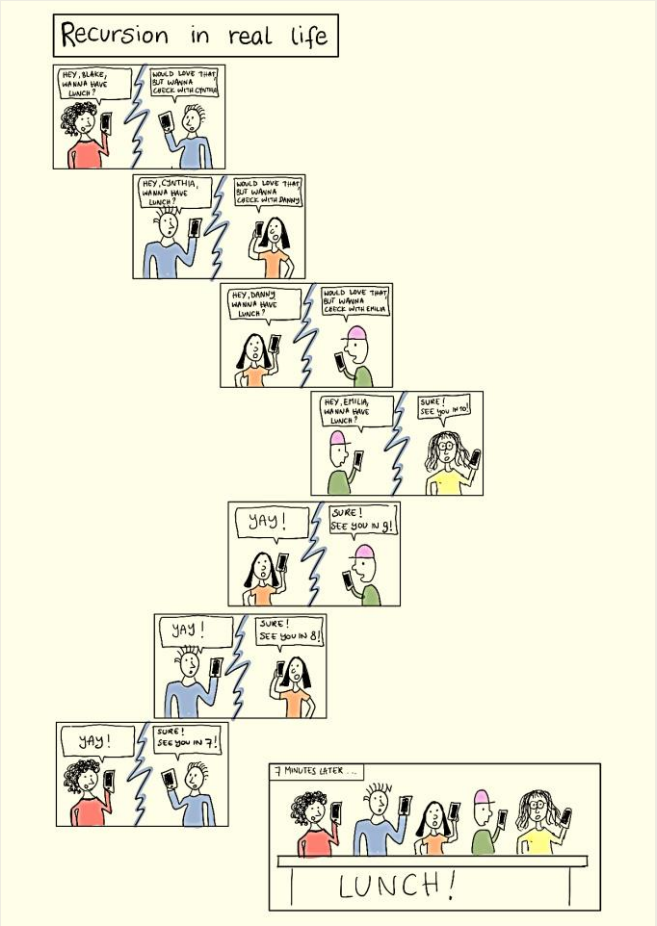
ITERATIVE ALGORITHMS

- It solves problems by repeating steps over and over, typically using a loop.



RECURSIVE ALGORITHMS

- It is a method of problem-solving where you solve smaller instances of the problem until you arrive at a solution.



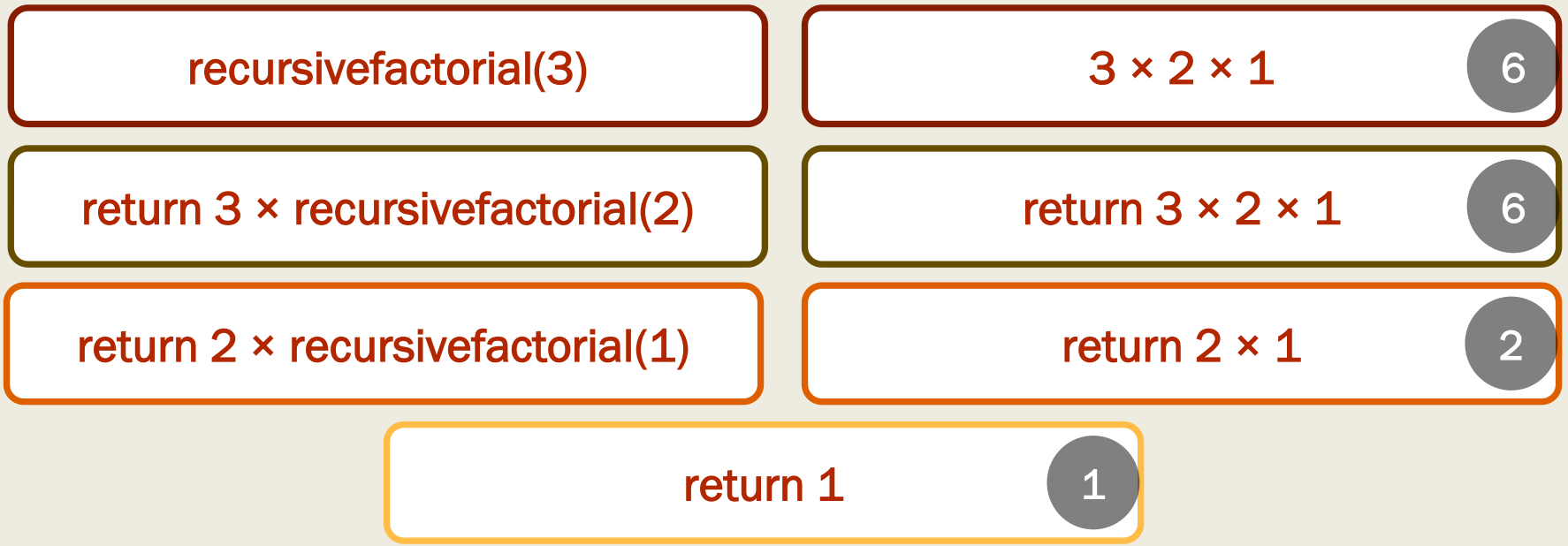
RECURSIVE ALGORITHMS

- Recursive algorithms rely on functions that call themselves.
- Any problem you can solve with an iterative algorithm, you can also solve with a recursive one.
- **Base case**: a condition that ends a recursive algorithm to stop it from continuing forever.
- The three laws of recursion:
  - *A recursive algorithm must call itself recursively.*
  - *A recursive algorithm must have a base case.*
  - *A recursive algorithm must change its state and move toward the base case.*

RECURSIVE ALGORITHMS

- Factorial:

$$3! = 3 * 2 * 1$$



```python
def iterative_factorial(n):
    print(f"Starting iterative_factorial({n})")

    result = 1
    for i in range(1, n + 1):
        print(f"Multiplying result ({result}) by {i}")
        result *= i

    print(f"Final result after loop: {result}")
    return result


num = int(input("Enter a number: "))
print(f"\nFinal result: {iterative_factorial(num)}")
```

```
Enter a number: 6
Starting iterative_factorial(6)
Multiplying result (1) by 1
Multiplying result (1) by 2
Multiplying result (2) by 3
Multiplying result (6) by 4
Multiplying result (24) by 5
Multiplying result (120) by 6
Final result after loop: 720

Final result: 720
```

```python
def recursive_factorial(n, depth=0):
    indent = "  " * depth
    print(f"{indent}Entering recursive_factorial({n})")

    if n == 1:
        print(f"{indent}Base case reached, returning 1")
        return 1

    result = n * recursive_factorial(n - 1, depth + 1)
    print(f"{indent}Returning {n} * factorial({n - 1}) = {result}")
    return result


num = int(input("Enter a number: "))
print(f"\nFinal result: {recursive_factorial(num)}")
```

```
Enter a number: 6
Entering recursive_factorial(6)
  Entering recursive_factorial(5)
    Entering recursive_factorial(4)
      Entering recursive_factorial(3)
        Entering recursive_factorial(2)
          Entering recursive_factorial(1)
          Base case reached, returning 1
        Returning 2 * factorial(1) = 2
      Returning 3 * factorial(2) = 6
    Returning 4 * factorial(3) = 24
  Returning 5 * factorial(4) = 120
Returning 6 * factorial(5) = 720

Final result: 720
```