Strings in C++

Recap from Last Time

Another View of Factorials

$$n! = \begin{cases} 1 & \text{if } n = 0 \\ n \times (n - 1)! & \text{otherwise} \end{cases}$$

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

Another View of Factorials

$$n! = \begin{cases} 1 & \text{if } n = 0 \\ n \times (n - 1)! & \text{otherwise} \end{cases}$$

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

Thinking Recursively

- Solving a problem with recursion requires two steps.
- First, determine how to solve the problem for simple cases.
  - This is called the ***base case***.
- Second, determine how to break down larger cases into smaller instances.
  - This is called the ***recursive step***.

Thinking Recursively

```
if (The problem is very simple) {
   Directly solve the problem.
   Return the solution.
 } else {
   Split the problem into one or more
   smaller problems with the same
   structure as the original.
   Solve each of those smaller problems.
   Combine the results to get the overall
   solution.
   Return the overall solution.
}
```

These simple cases are called *base cases*.

These are the *recursive cases*.

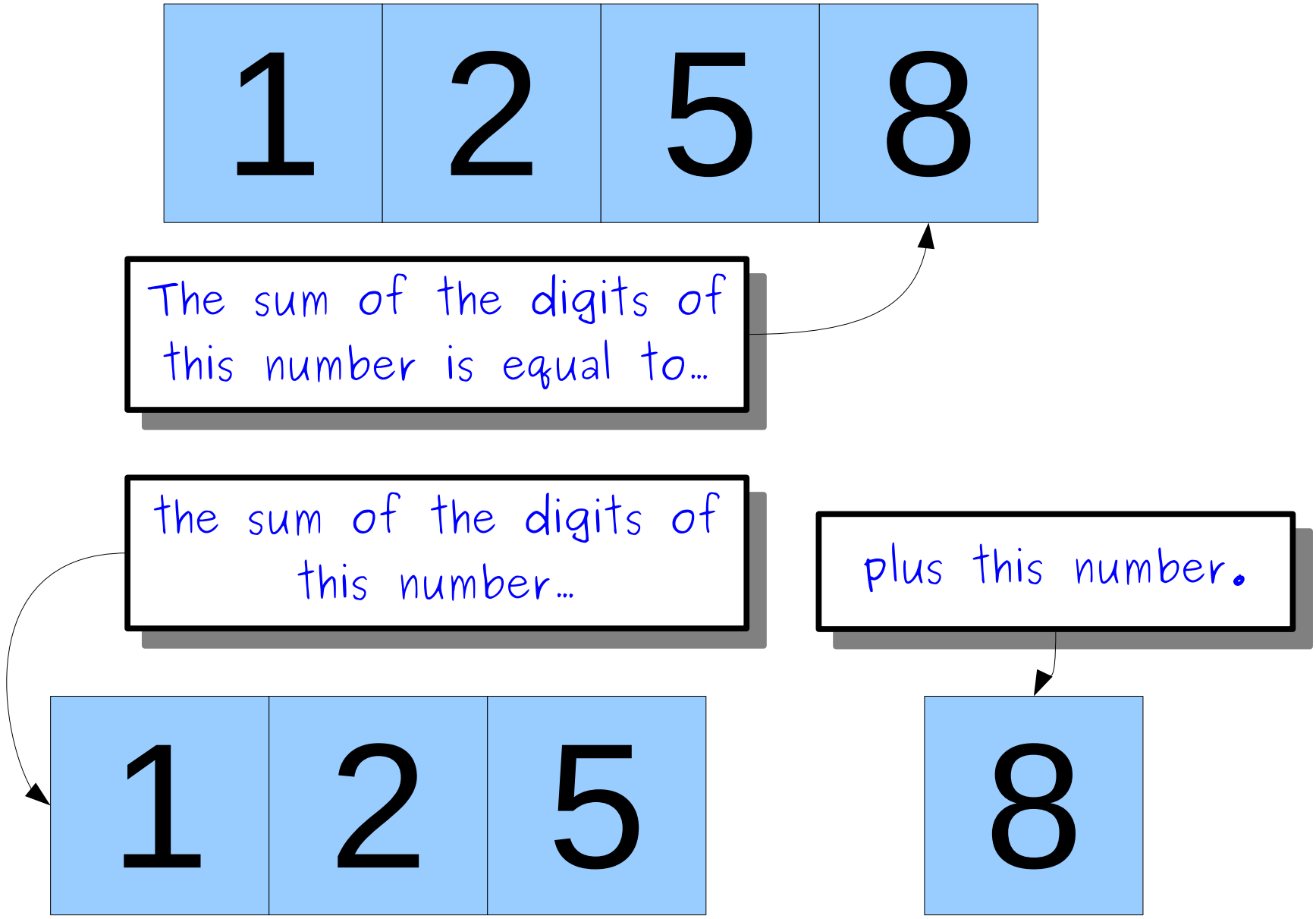
Summing Up Digits

- On Wednesday, we wrote this function to sum up the digits of a nonnegative integer:

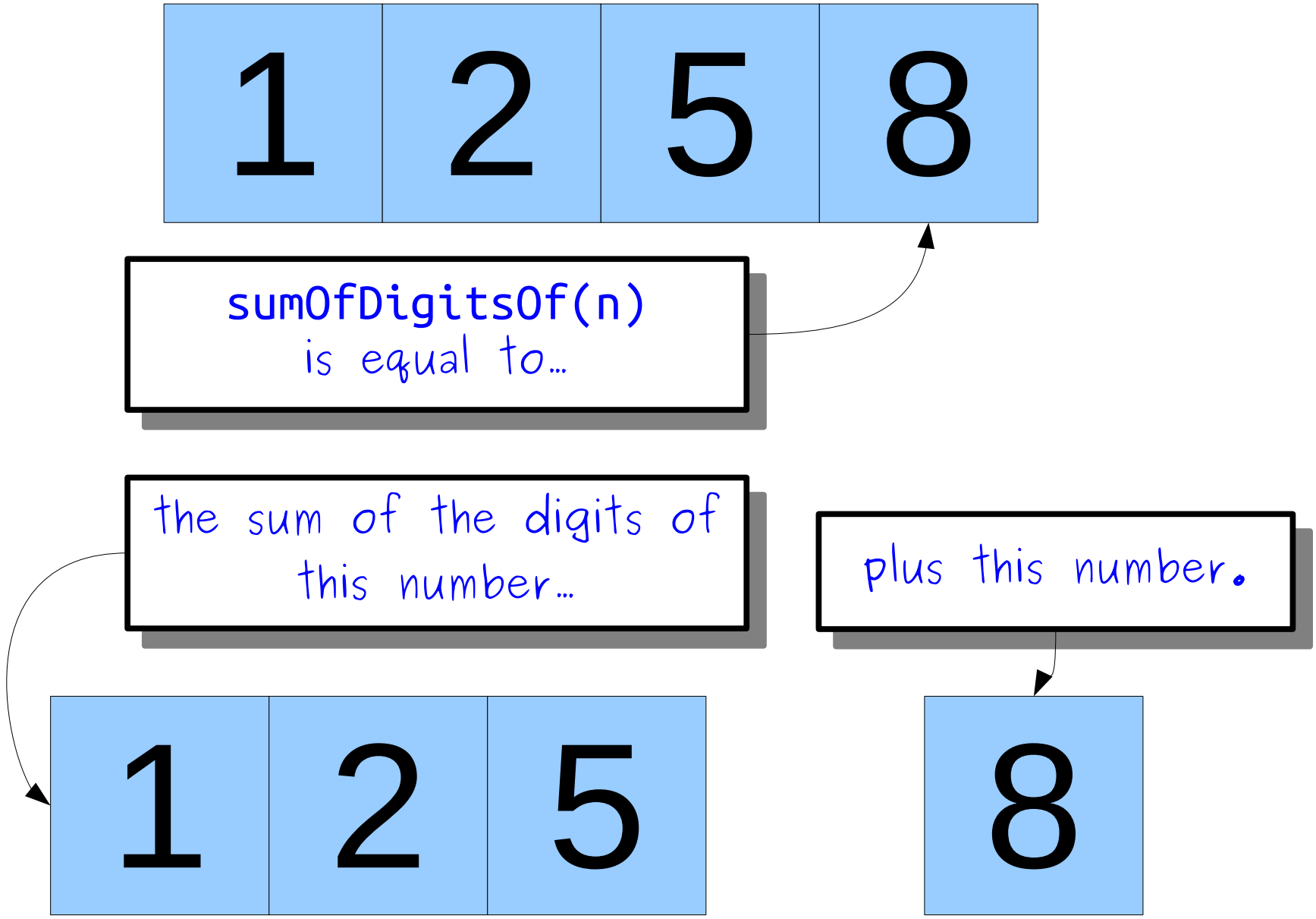
```
int sumOfDigitsOf(int n) {
    int result = 0;

    while (n > 0) {
        result += (n % 10);
        n /= 10;
    }

    return result;

}
```

- Let’s rewrite this function recursively!

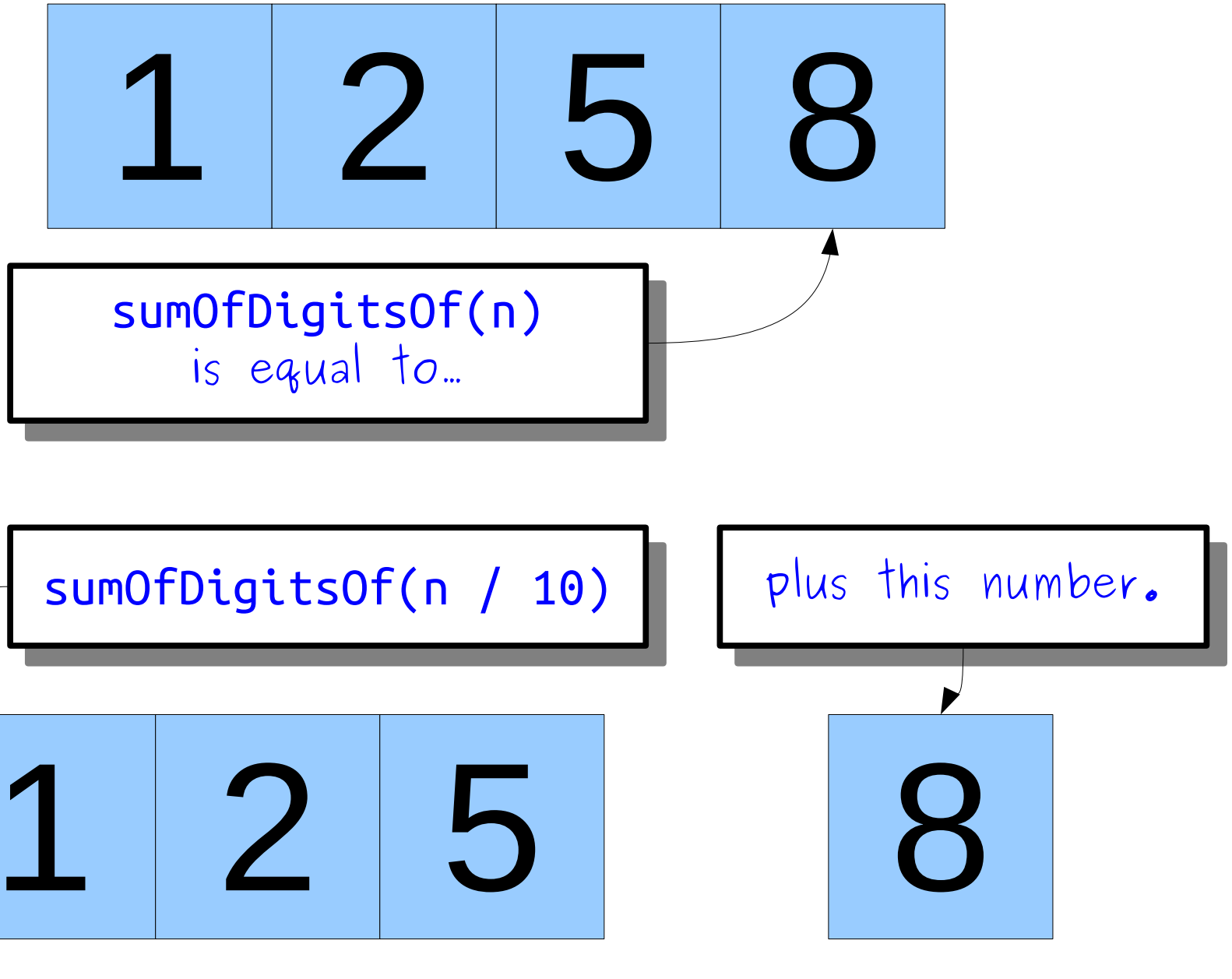
Summing Up Digits



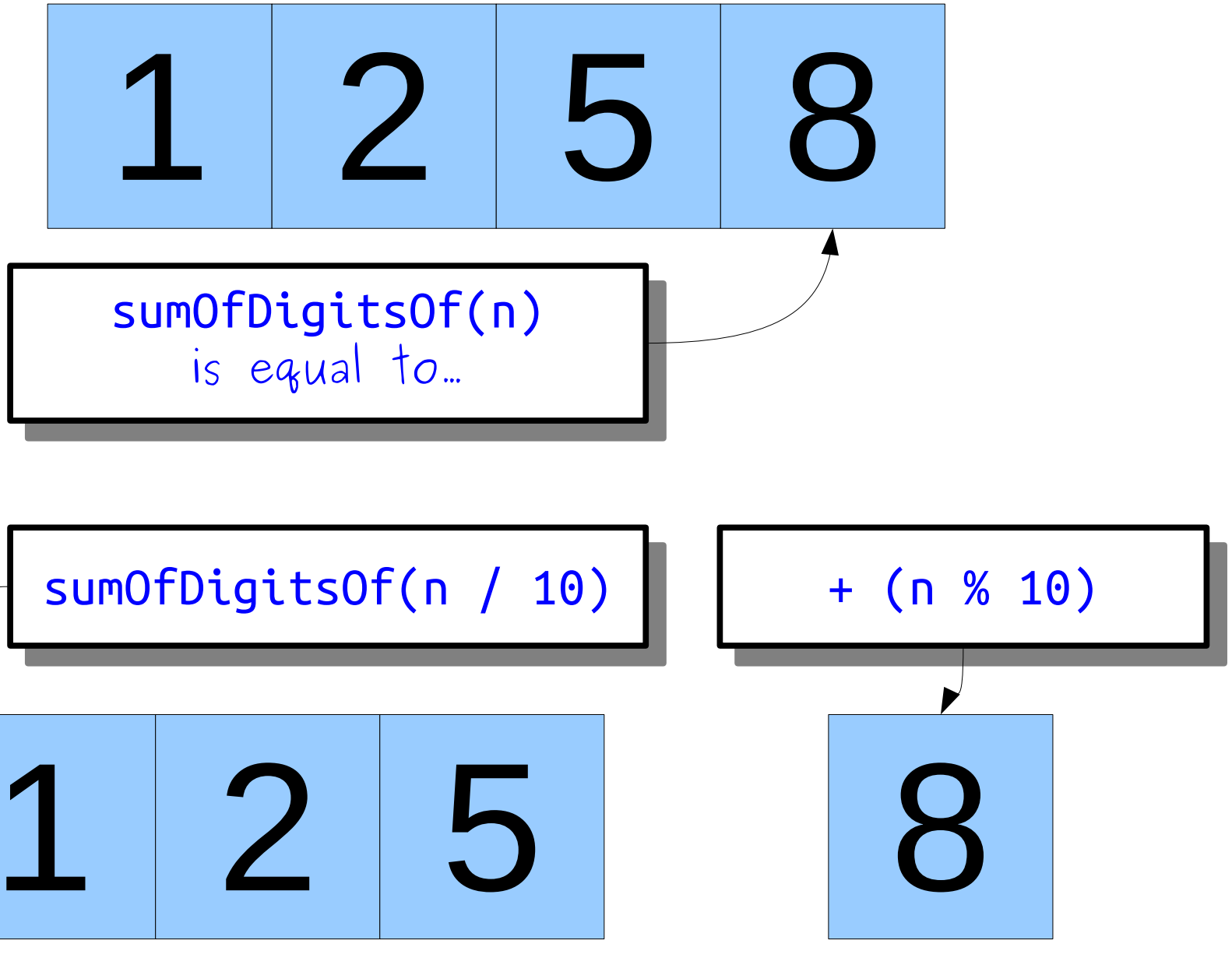
Summing Up Digits



Summing Up Digits



Summing Up Digits



New Stuff!

Tracing the Recursion

```
int main() {
    int sum = sumOfDigitsOf(137);
    cout << "Sum is " << sum << endl;
}
```

Tracing the Recursion

```
int main() {
    int sum = sumOfDigitsOf(137);
    cout << "Sum is " << sum << endl;
}
```

Tracing the Recursion

```
int main() {
}
```

```
int sumOfDigitsOf(int n) {
    if (n < 10) {
        return n;
    } else {
        return sumOfDigitsOf(n / 10) + (n % 10);
    }
}
```

int n **137**

Tracing the Recursion

```
int main() {
}
```

```
int sumOfDigitsOf(int n) {
    if (n < 10) {
        return n;
    } else {
        return sumOfDigitsOf(n / 10) + (n % 10);
    }
}
```

int n **137**

Tracing the Recursion

```
int main() {
}
```

```
int sumOfDigitsOf(int n) {
    if (n < 10) {
        return n;
    } else {
        return sumOfDigitsOf(n / 10) + (n % 10);
    }
}
```

int n **137**

Tracing the Recursion

```
int main() {
}
```

```
int sumOfDigitsOf(int n) {
    if (n < 10) {
        return n;
    } else {
        return sumOfDigitsOf(n / 10) + (n % 10);
    }
}
```

int n **137**

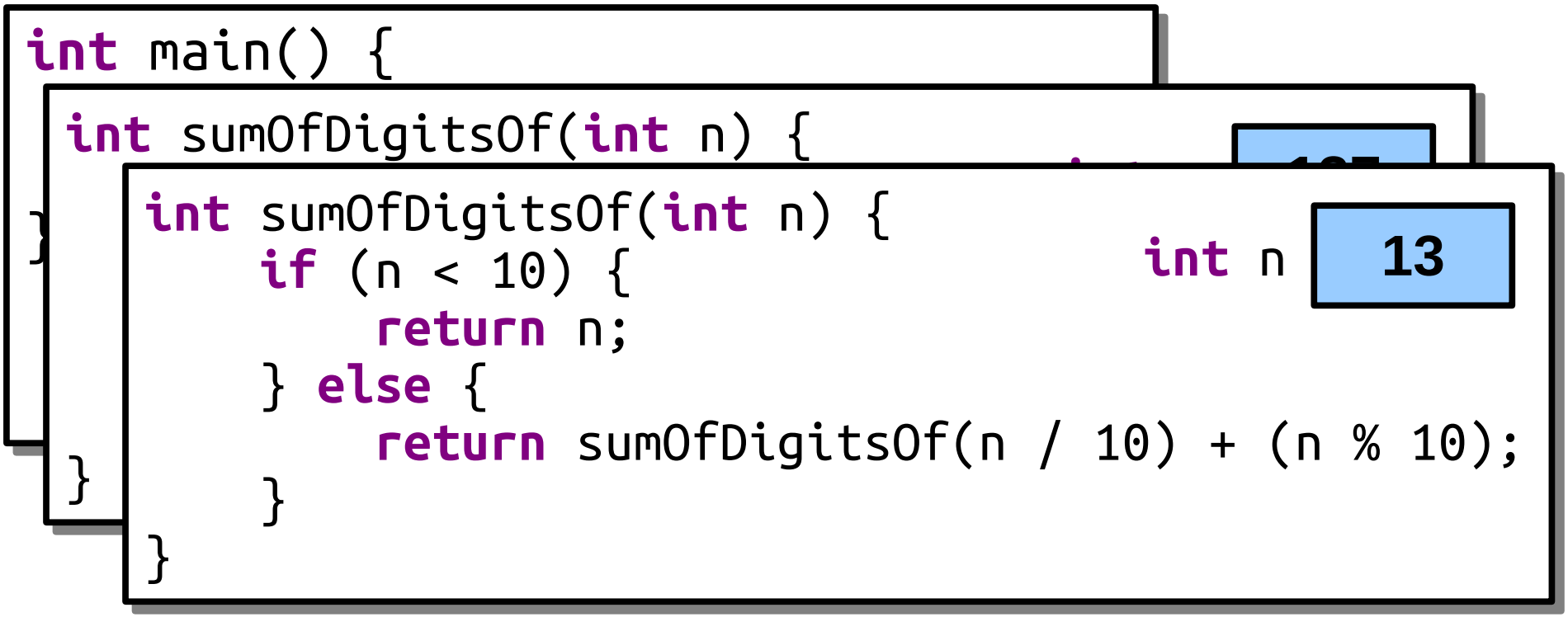
Tracing the Recursion

```
int main() {
}
```

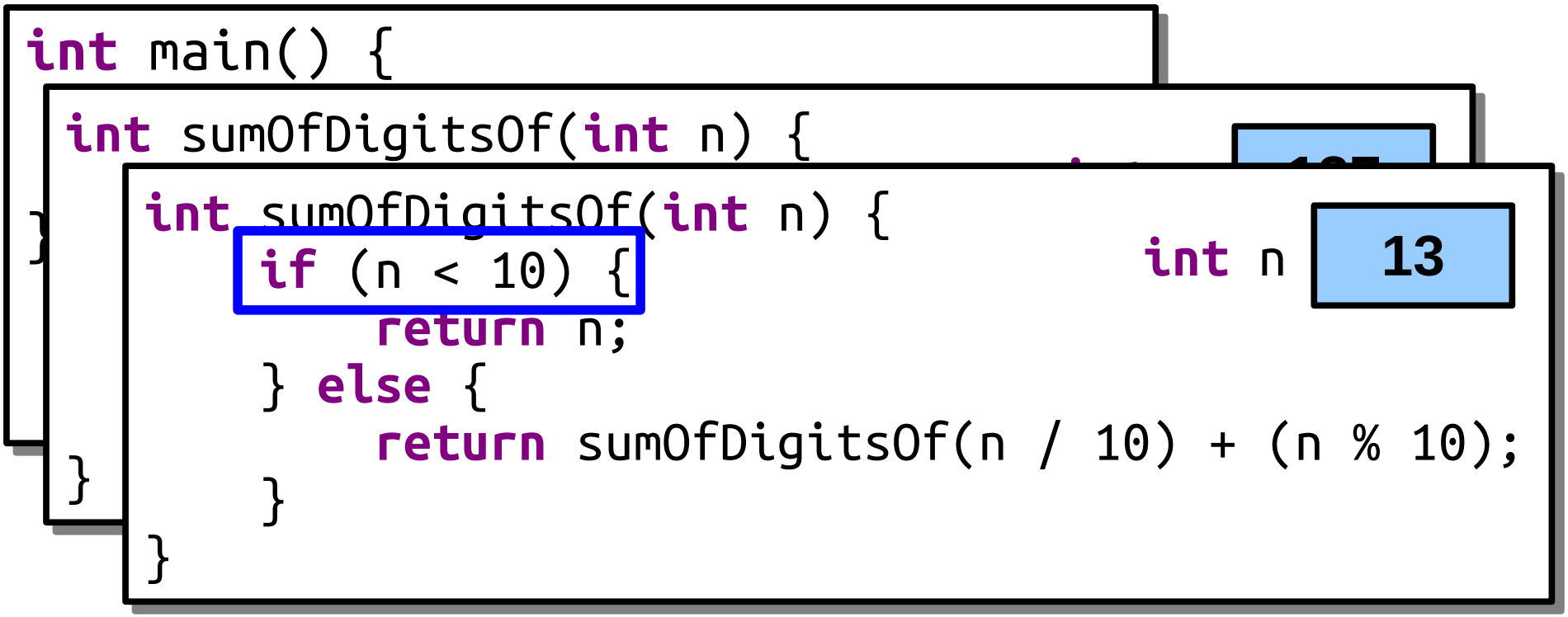
```
int sumOfDigitsOf(int n) {
    if (n < 10) {
        return n;
    } else {
        return sumOfDigitsOf(n / 10) + (n % 10);
    }
}
```

int n **137**

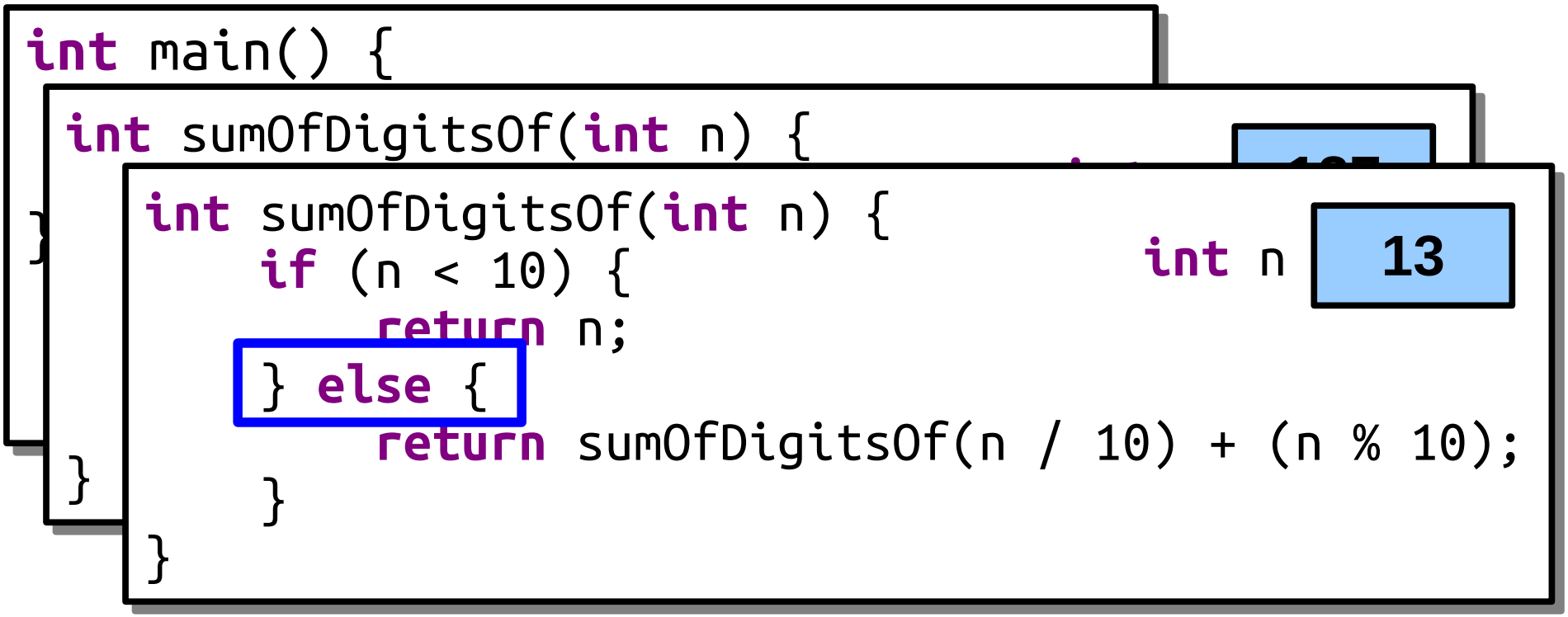
Tracing the Recursion



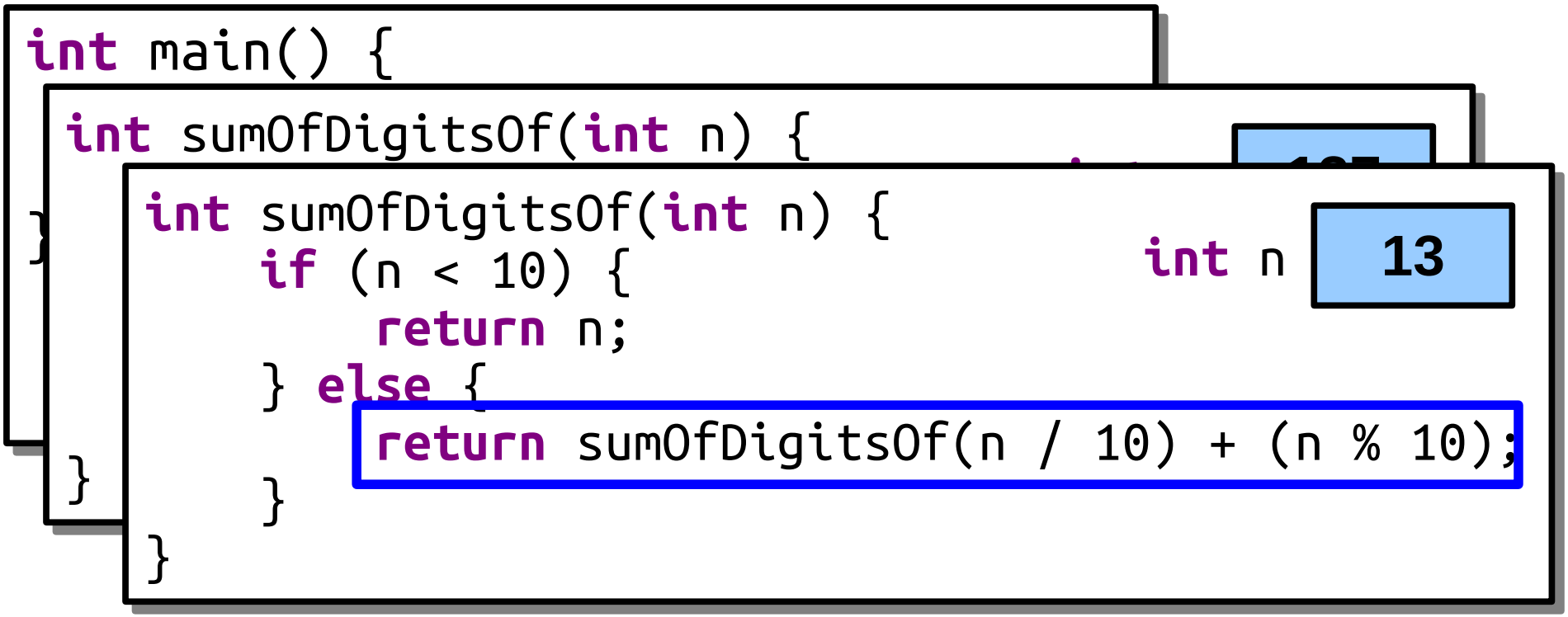
Tracing the Recursion



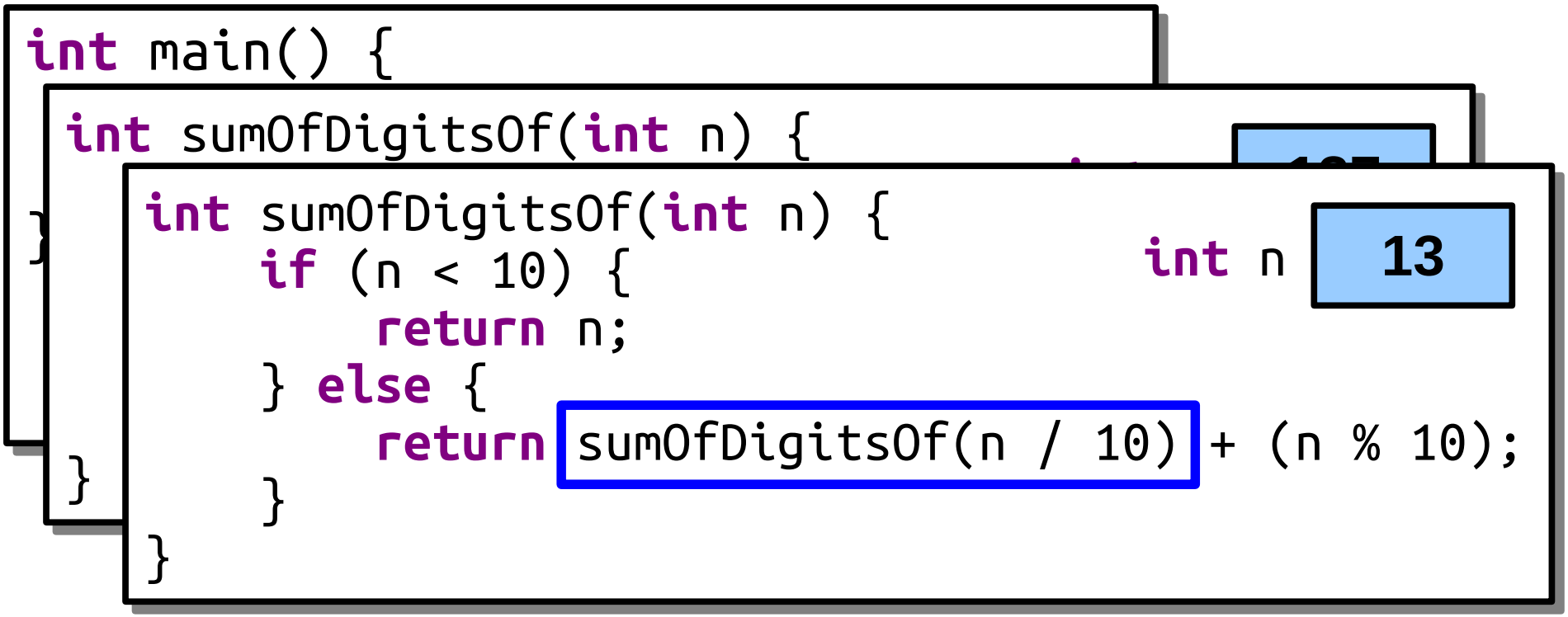
Tracing the Recursion



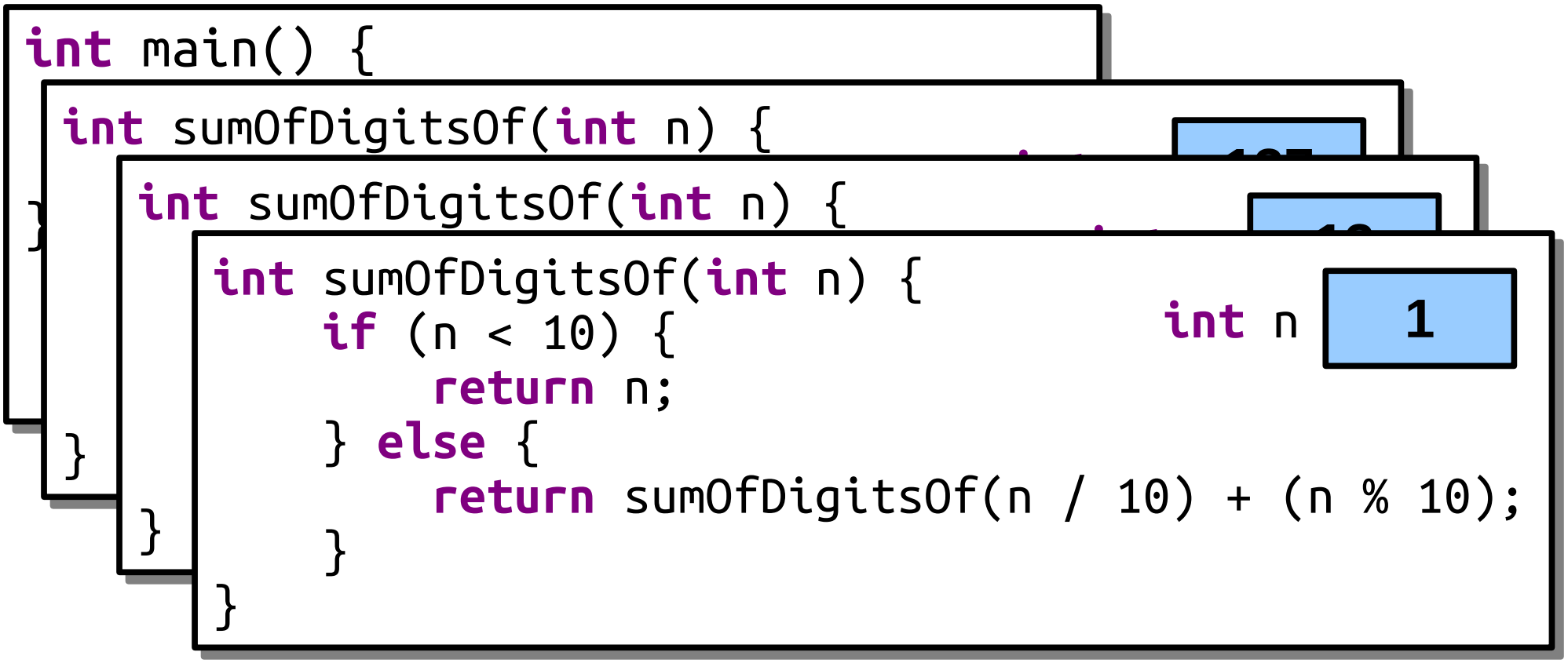
Tracing the Recursion



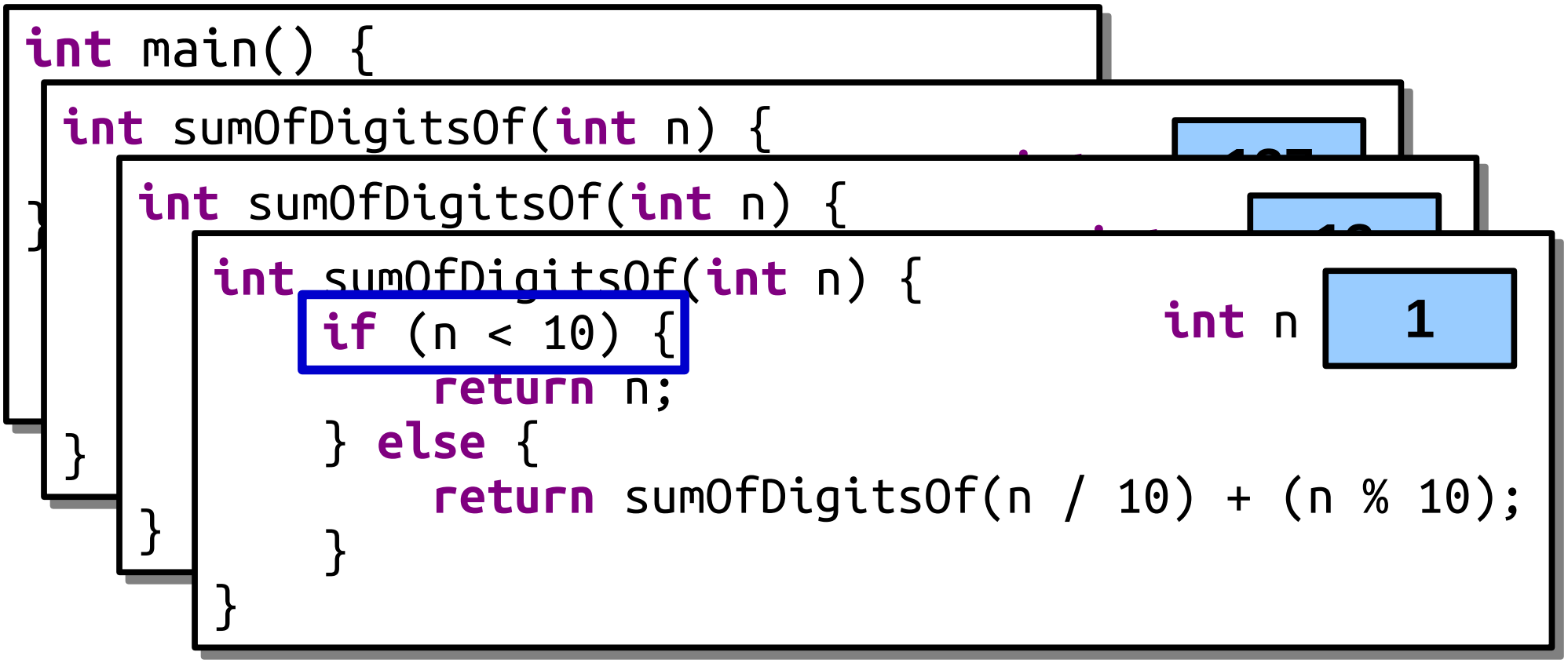
Tracing the Recursion



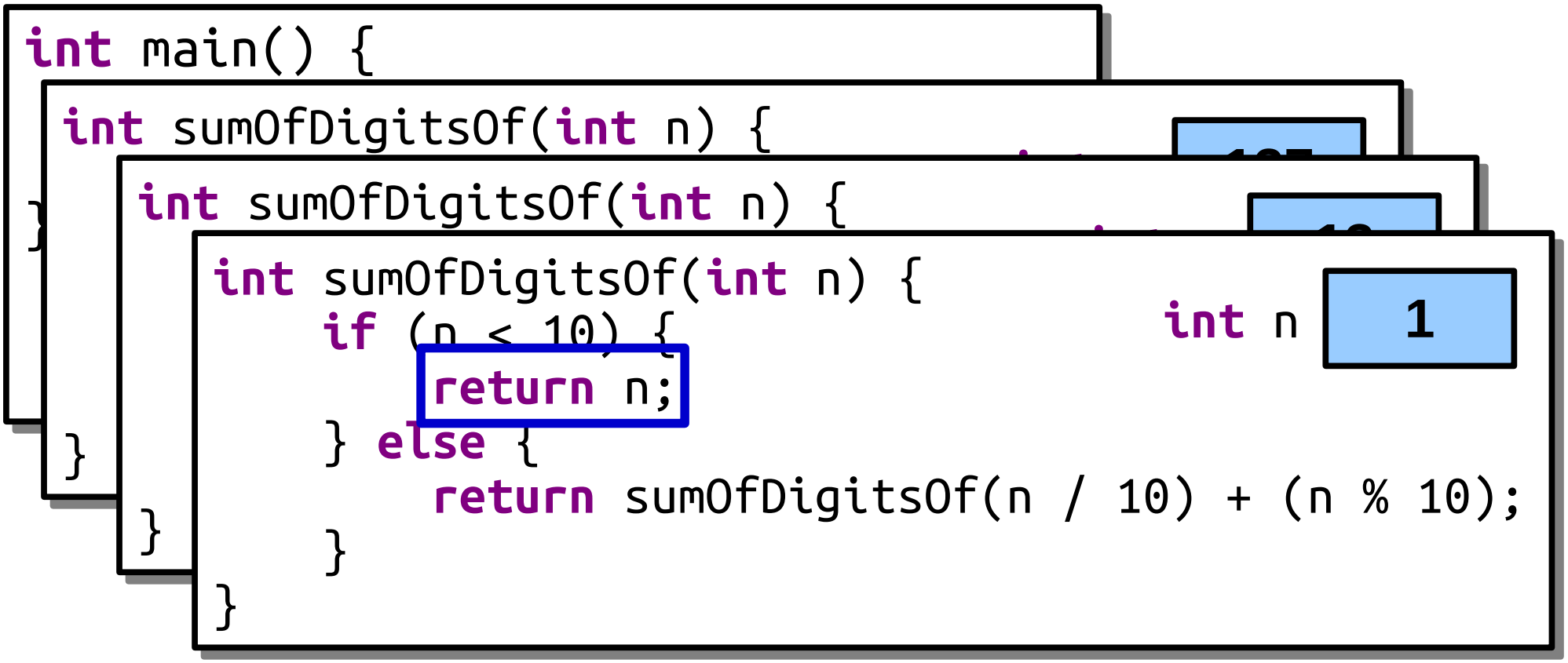
Tracing the Recursion



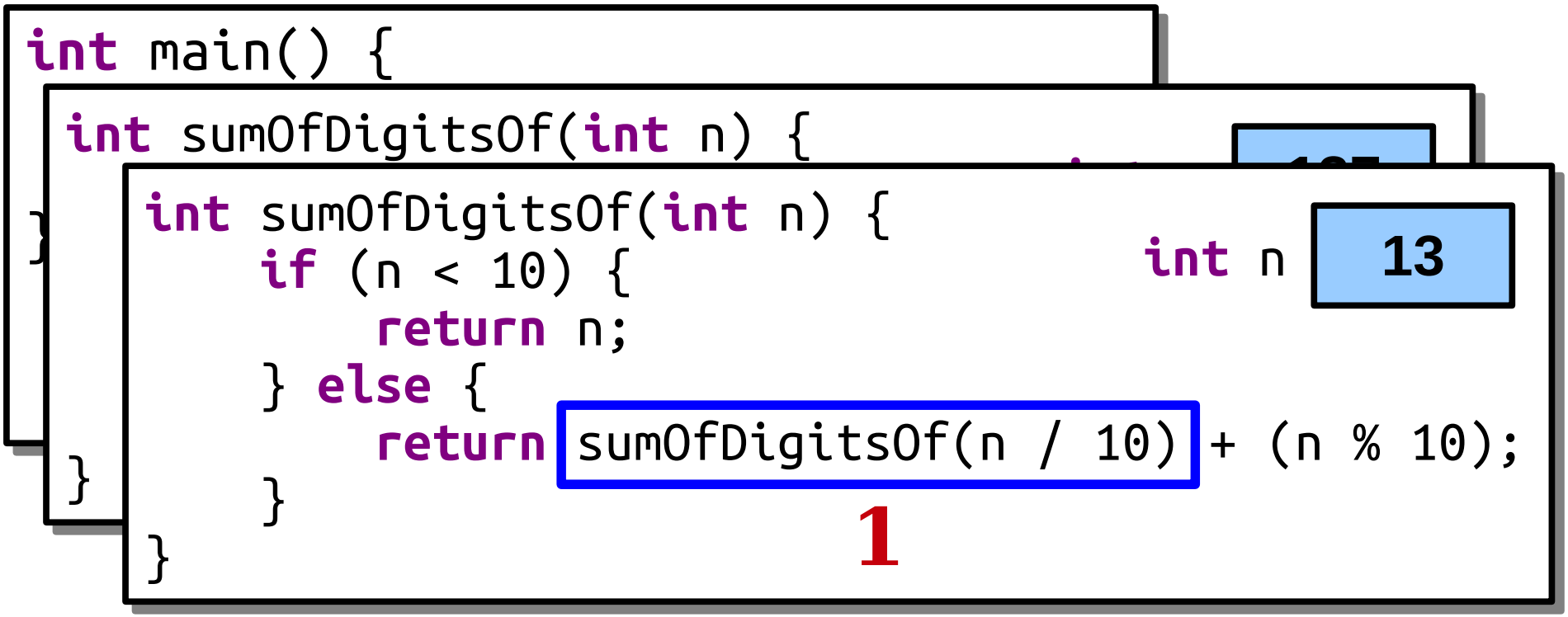
Tracing the Recursion



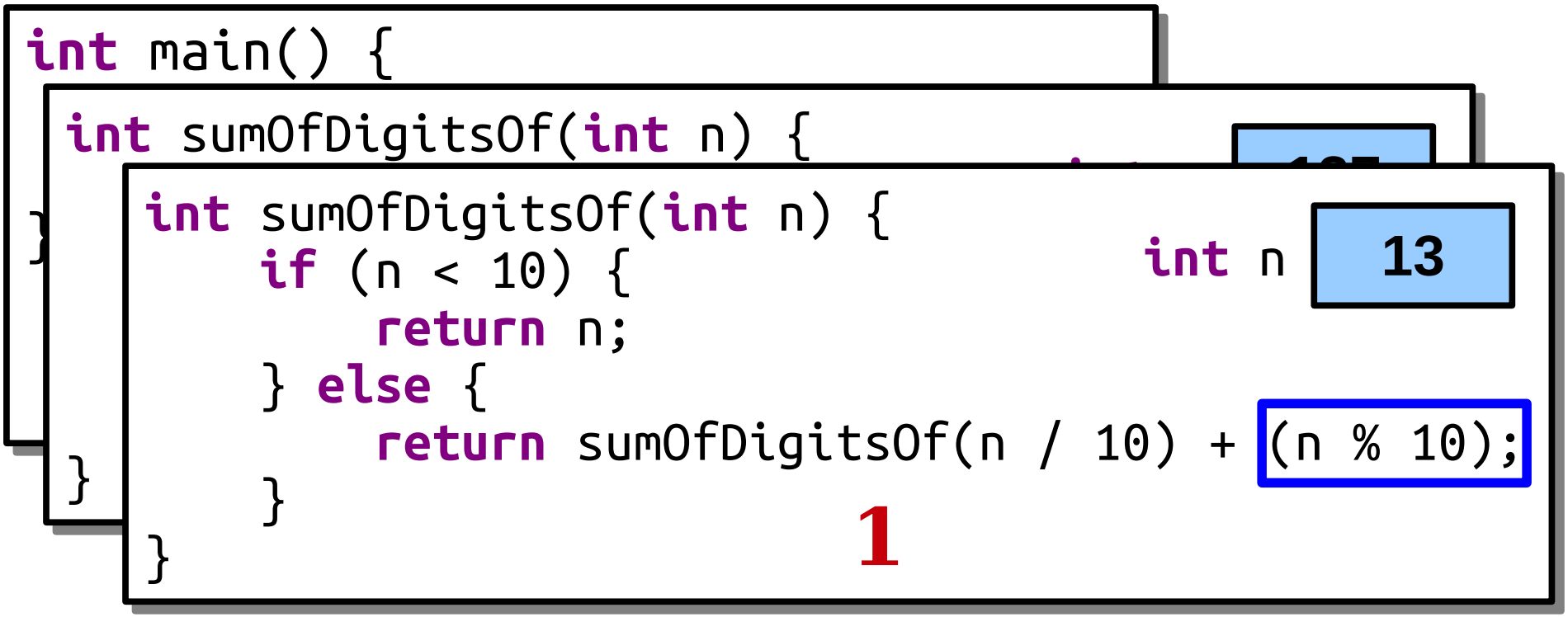
Tracing the Recursion



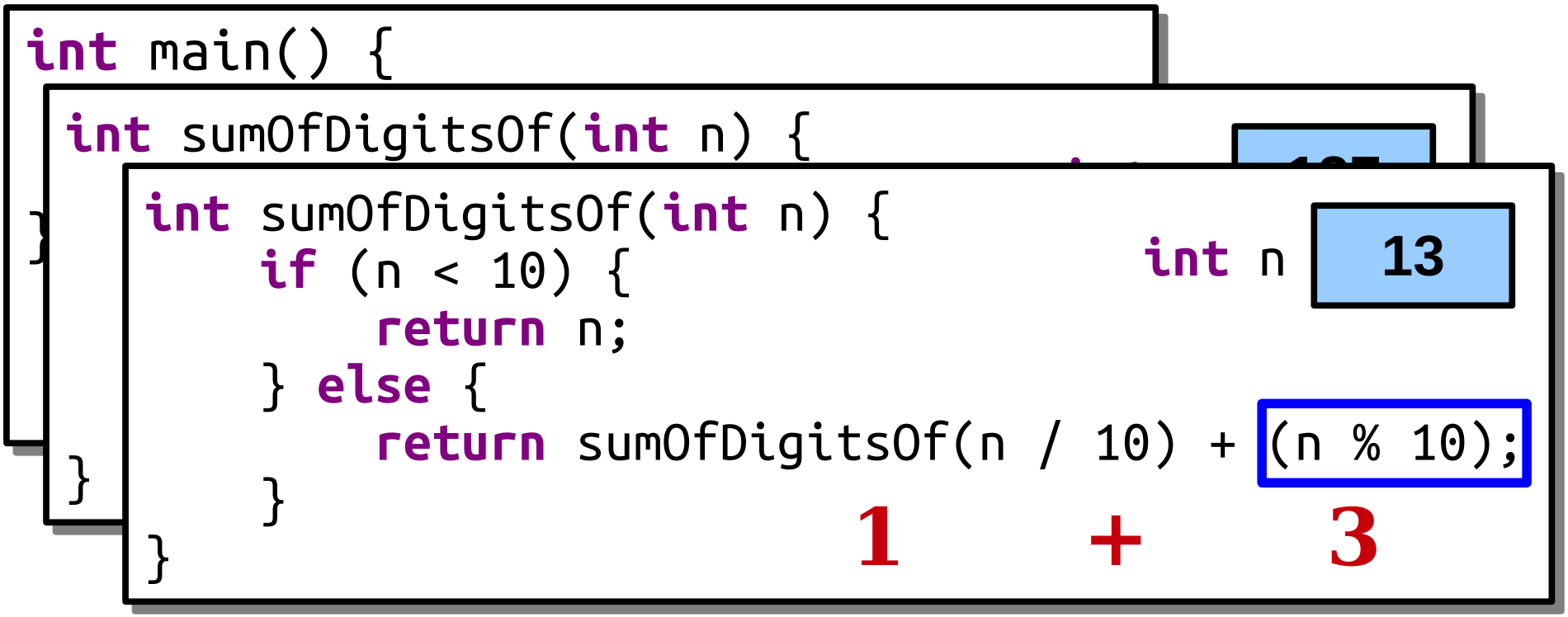
Tracing the Recursion



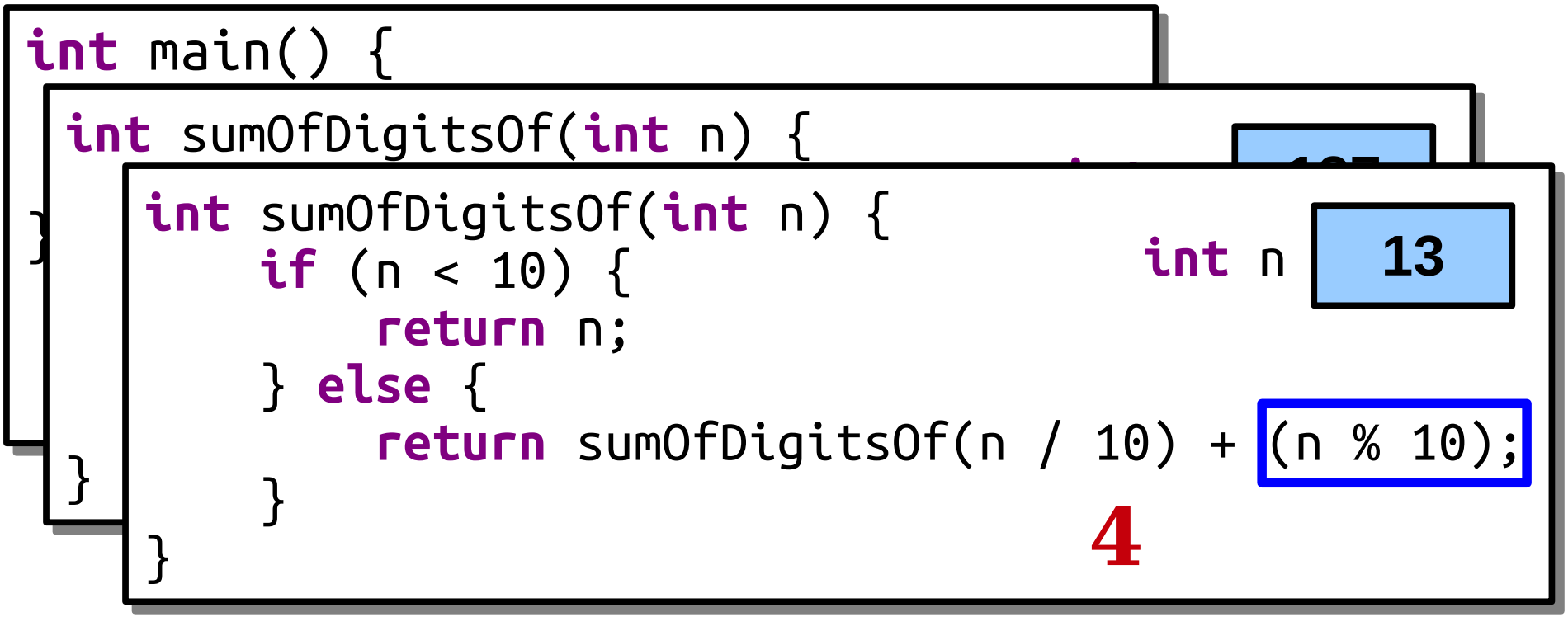
Tracing the Recursion



Tracing the Recursion



Tracing the Recursion



Tracing the Recursion

```
int main() {
}
```

```
int sumOfDigitsOf(int n) {
    if (n < 10) {
        return n;
    } else {
        return sumOfDigitsOf(n / 10) + (n % 10);
    }
}
```

int n **137**

4

Tracing the Recursion

```
int main() {
}
```

```
int sumOfDigitsOf(int n) {
    if (n < 10) {
        return n;
    } else {
        return sumOfDigitsOf(n / 10) + (n % 10);
    }
}
```

int n **137**

4

Tracing the Recursion

```
int main() {
}
```

```
int sumOfDigitsOf(int n) {
    if (n < 10) {
        return n;
    } else {
        return sumOfDigitsOf(n / 10) + (n % 10);
    }
}
```

int n **137**

4 + 7

Tracing the Recursion

```
int main() {
}
```

```
int sumOfDigitsOf(int n) {
    if (n < 10) {
        return n;
    } else {
        return sumOfDigitsOf(n / 10) + (n % 10);
    }
}
```

int n **137**

11

Tracing the Recursion

```
int main() {
    int sum = sumOfDigitsOf(137);
    cout << "Sum is " << sum << endl;
}
```

11

Thinking Recursively

```
if (The problem is very simple) {
   Directly solve the problem.
   Return the solution.
 } else {
   Split the problem into one or more
   smaller problems with the same
   structure as the original.
   Solve each of those smaller problems.
   Combine the results to get the overall
   solution.
   Return the overall solution.
}
```

These simple cases are called *base cases*.

These are the *recursive cases*.

Example: ***Digital Roots***

Digital Roots

- The ***digital root*** is the number you get by repeatedly summing the digits of a number until you're down to a single digit.
- What is the digital root of 5?
- What is the digital root of 27?
- What is the digital root of 137?

Digital Roots

- The ***digital root*** is the number you get by repeatedly summing the digits of a number until you're down to a single digit.
- What is the digital root of 5?
  - 5 is a single digit, so the answer is 5.
- What is the digital root of 27?

- What is the digital root of 137?

Digital Roots

- The ***digital root*** is the number you get by repeatedly summing the digits of a number until you're down to a single digit.
- What is the digital root of 5?
  - 5 is a single digit, so the answer is 5.
- What is the digital root of 27?
  - 2 + 7 = 9.

- What is the digital root of 137?

Digital Roots

- The ***digital root*** is the number you get by repeatedly summing the digits of a number until you're down to a single digit.
- What is the digital root of 5?
  - 5 is a single digit, so the answer is 5.
- What is the digital root of 27?
  - 2 + 7 = 9.
  - The answer is 9.
- What is the digital root of 137?

Digital Roots

- The ***digital root*** is the number you get by repeatedly summing the digits of a number until you're down to a single digit.
- What is the digital root of 5?
  - 5 is a single digit, so the answer is 5.
- What is the digital root of 27?
  - 2 + 7 = 9.
  - The answer is 9.
- What is the digital root of 137?
  - 1 + 3 + 7 = 11.

Digital Roots

- The ***digital root*** is the number you get by repeatedly summing the digits of a number until you're down to a single digit.
- What is the digital root of 5?
  - 5 is a single digit, so the answer is 5.
- What is the digital root of 27?
  - 2 + 7 = 9.
  - The answer is 9.
- What is the digital root of 137?
  - 1 + 3 + 7 = 11.
  - 1 + 1 = 2.

Digital Roots

- The ***digital root*** is the number you get by repeatedly summing the digits of a number until you're down to a single digit.
- What is the digital root of 5?
  - 5 is a single digit, so the answer is 5.
- What is the digital root of 27?
  - 2 + 7 = 9.
  - The answer is 9.
- What is the digital root of 137?
  - 1 + 3 + 7 = 11.
  - 1 + 1 = 2.
  - The answer is 2.

Digital Roots

Digital Roots

The digital root of 9 2 5 8

Digital Roots

The digital root of 9 2 5 8 is the same as

Digital Roots

The digital root of 9 2 5 8 is the same as

The digital root of 9+2+5+8

Digital Roots

The digital root of 9 2 5 8 is the same as

The digital root of 2 4

Digital Roots

The digital root of 9 2 5 8 is the same as

The digital root of 2 4 which is the same as

Digital Roots

The digital root of $9\ 2\ 5\ 8$ is the same as

The digital root of $2\ 4$ which is the same as

The digital root of $2 + 4$

Digital Roots

The digital root of 9 2 5 8 is the same as

The digital root of 2 4 which is the same as

The digital root of 6

Thinking Recursively

```
if (The problem is very simple) {
   Directly solve the problem.
   Return the solution.
 } else {
   Split the problem into one or more
   smaller problems with the same
   structure as the original.
   Solve each of those smaller problems.
   Combine the results to get the overall
   solution.
   Return the overall solution.
}
```

These simple cases are called *base cases*.

These are the *recursive cases*.

Time-Out for Announcements!

Section Signups

- Section signups are open right now. They close Sunday at 5PM.
- Sign up for section at

  https://cs198.stanford.edu/

- Click on “CS106 Sections Login,” then choose “Section Signup.”

Assignment 1

- Assignment 0 was due today at the start of class.
- ***Assignment 1: Welcome to C++*** goes out today. It's due on Friday, January 22nd at the start of class.
  - Play around with C++ and the Stanford libraries!
  - Get some practice with recursion!
  - Explore the debugger!
  - See some pretty pictures!
- We recommend making slow and steady progress on this assignment throughout the course of the week. There's a recommended timetable on the front page of the handout.
- We've posted three handouts online. We strongly recommend reading over them before starting.
  - Handout 06: Debugging Your Code
  - Handout 07: Assignment Submission Checklist
  - Handout 08: Python-to-C++ Guide

Submission Policy

- If you submit your assignment before the deadline, we'll give you a small "on-time" bonus at the end of the quarter.
- There's a 48-hour grace period after the stated deadline. If you submit during this period, we'll still accept your work for credit, but you won't get the "on-time" bonus.
- No work submitted more than 48 hours late will be accepted (unless you've contacted Chase and arranged for an extension).

Assignment Grading

- Your coding assignments are graded on both functionality and on coding style.
- The ***functionality score*** is based on correctness.
  - Do your programs produce the correct output?
  - Do they work on all inputs?
  - etc.
- The ***style score*** is based on how well your program is written.
  - Are your programs well-structured?
  - Do you decompose problems into smaller pieces?
  - Do you use variable naming conventions consistently?
  - etc.

Getting Help

Getting Help

- ***LaIR Hours***
  - Sunday – Thursday, 5PM – 9PM Pacific.
  - Also Tuesday and Thursday 9AM – 11AM Pacific. These those hours are reserved for folks who can’t make the normal time slot.
  - Visit this site to request help: **https://cs198.stanford.edu/lair/student**
- ***Chase’s and Keith’s Office Hours***
  - Check Canvas for Zoom links and times.

One More Unto the Breach!

Strings in C++

C++ Strings

- To use strings, you need to add the line

```
#include <string>
```

- to the top of your program to import the strings library. You'll get Cruel and Unusual Error Messages if you forget to do this.
- Then, you can do whatever stringy things you want! Here's some examples…

```cpp
/*          C++ Version          */
string s = "Elena Kagan";
s += ", joined " + to_string(2010);

char first = s[0];
char last  = s[s.length() - 1]

if (s.find("e") != string::npos) {
  string first = s.substr(0, 5);
  string last  = s.substr(7);
}

if (s == "Sonia Sotomayor") {
  cout << "John Roberts" << endl;
}
```

```python
"""      Python Version      """
s = "Elena Kagan"
s += ", joined " + str(2010)

first = s[0]
last  = s[-1]

if 'e' in s:
    first = s[0:5]
    last  = s[7:]


if s == "Sonia Sotomayor":
    print("John Roberts")
```

```java
/*          Java Version          */
String s = "Elena Kagan";
s += ", joined " + 2010;

char first = s.charAt(0);
char last  = s.charAt(s.length());

if (s.indexOf("e") != -1) {
  String first = s.substring(0, 5);
  String last  = s.substring(7);
}

if (s.equals("Sonia Sotomayor")) {
  System.out.println("John Roberts");
}
```

```javascript
//    JavaScript Version
let s = "Elena Kagan";
s += ", joined " + 2010;

let first = s[0];
let last  = s[s.length - 1];

if (s.indexOf("e") != -1) {
  let first = s.substring(0, 5);
  let last  = s.substring(7);
}

if (s === "Sonia Sotomayor") {
  console.log("John Roberts");
}
```

```cpp
/*          C++ Version          */
string s = "Elena Kagan";
s += ", joined " + to_string(2010);

char first = s[0];
char last  = s[s.length() - 1]

if (s.find("e") != string::npos) {
  string first = s.substr(0, 5);
  string last  = s.substr(7);
}

if (s == "Sonia Sotomayor") {
  cout << "John Roberts" << endl;
}
```

```python
"""      Python Version      """
s = "Elena Kagan"
s += ", joined " + str(2010)

first = s[0]
last  = s[-1]

if 'e' in s:
    first = s[0:5]
    last  = s[7:]


if s == "Sonia Sotomayor":
    print("John Roberts")
```

```java
/*          Java Version          */
String s = "Elena Kagan";
s += ", joined " + 2010;

char first = s.charAt(0);
char last  = s.charAt(s.length());

if (s.indexOf("e") != -1) {
  String first = s.substring(0, 5);
  String last  = s.substring(7);
}

if (s.equals("Sonia Sotomayor")) {
  System.out.println("John Roberts");
}
```

```javascript
//    JavaScript Version
let s = "Elena Kagan";
s += ", joined " + 2010;

let first = s[0];
let last  = s[s.length - 1];

if (s.indexOf("e") != -1) {
  let first = s.substring(0, 5);
  let last  = s.substring(7);
}

if (s === "Sonia Sotomayor") {
  console.log("John Roberts");
}
```

```cpp
/*          C++ Version          */
string s = "Elena Kagan";
s += ", joined " + to_string(2010);

char first = s[0];
char last  = s[s.length() - 1]

if (s.find("e") != string::npos) {
  string first = s.substr(0, 5);
  string last  = s.substr(7);
}

if (s == "Sonia Sotomayor") {
  cout << "John Roberts" << endl;
}
```

```python
"""      Python Version      """
s = "Elena Kagan"
s += ", joined " + str(2010)

first = s[0]
la
if

if
```

C++ strings must be declared using double quotes rather than single quotes.

```java
/*          Java Version          */
String s = "Elena Kagan";
s += ", joined " + 2010;

char first = s.charAt(0);
char last  = s.charAt(s.length());

if (s.indexOf("e") != -1) {
  String first = s.substring(0, 5);
  String last  = s.substring(7);
}

if (s.equals("Sonia Sotomayor")) {
  System.out.println("John Roberts");
}
```

```javascript
//    JavaScript Version
let s = "Elena Kagan";
s += ", joined " + 2010;

let first = s[0];
let last  = s[s.length - 1];

if (s.indexOf("e") != -1) {
  let first = s.substring(0, 5);
  let last  = s.substring(7);
}

if (s === "Sonia Sotomayor") {
  console.log("John Roberts");
}
```

```cpp
/*          C++ Version          */
string s = "Elena Kagan";
s += ", joined " + to_string(2010);

char first = s[0];
char last  = s[s.length() - 1]

if (s.find("e") != string::npos) {
  string first = s.substr(0, 5);
  string last  = s.substr(7);
}

if (s == "Sonia Sotomayor") {
  cout << "John Roberts" << endl;
}
```

```python
"""      Python Version      """
s = "Elena Kagan"
s += ", joined " + str(2010)

first = s[0]
last  = s[-1]

if 'e' in s:
```

You can use + and += to append to a string. You can only append other strings or characters. Use the to_string function to convert data to strings.

```java
/*          Java Version          */
String s = "Elena Kagan";
s += ", joined " + 2010;

char first = s.charAt(0);
char last  = s.charAt(s.length());

if (s.indexOf("e") != -1) {
  String first = s.substring(0, 5);
  String last  = s.substring(7);
}

if (s.equals("Sonia Sotomayor")) {
  System.out.println("John Roberts");
}
```

```javascript
s += ", joined " + 2010;

let first = s[0];
let last  = s[s.length - 1];

if (s.indexOf("e") != -1) {
  let first = s.substring(0, 5);
  let last  = s.substring(7);
}

if (s === "Sonia Sotomayor") {
  console.log("John Roberts");
}
```

```cpp
/*          C++ Version          */
string s = "Elena Kagan";
s += ", joined " + to_string(2010);

char first = s[0];
char last  = s[s.length() - 1]

if (s.find("e") != string::npos) {
  string first = s.substr(0, 5);
  string last  = s.substr(7);
}

if (s == "Sonia Sotomayor") {
  cout << "John Roberts" << endl;
}
```

```python
"""       Python Version       """
s = "Elena Kagan"
s += ", joined " + str(2010)

first = s[0]
last  = s[-1]

if 'e' in s:
    first = s[0:5]
    last  = s[7:]

if s == "Sonia Sotomayor":
```

```java
/*          Java Version          */
String s = "Elena Kagan";
s += ", joined " + 2010;

char first = s.charAt(0);
char last  = s.charAt(s.length());

if (s.indexOf("e") != -1) {
  String first = s.substring(0, 5);
  String last  = s.substring(7);
}

if (s.equals("Sonia Sotomayor")) {
  System.out.println("John Roberts");
}
```

You can select an individual character out of a string by using square brackets. Indices start at zero.

C++ has different types for individual characters (char) and for strings of zero or more characters (string). Check Chapter 1.5 of the textbook for details.

```cpp
/*          C++ Version          */
string s = "Elena Kagan";
s += ", joined " + to_string(2010);

char first = s[0];
char last  = s[s.length() - 1]

if (s.find("e") != string::npos) {
  string first = s.substr(0, 5);
  string last  = s.substr(7);
}

if (s == "Sonia Sotomayor") {
  cout << "John Roberts" << endl;
}
```

```python
"""       Python Version       """
s = "Elena Kagan"
s += ", joined " + str(2010)

first = s[0]
last  = s[-1]

if 'e' in s:
    first = s[0:5]
    last  = s[7:]

if s == "Sonia Sotomayor":
    print("John Roberts")
```

C++ doesn't support negative array indices the way that Python does. You can pick the last character of the string by getting its length and subtracting one.

```java
/*          Java Version          */
String s = "Elena Kagan";
s += ", joined " + 2010;

char first = s.charAt(0);
char last  = s.charAt(s.length());

if (s.indexOf("e") != -1) {
  String first = s.substring(0, 5);
  String last  = s.substring(7);
}

if (s.equals("Sonia Sotomayor")) {
  System.out.println("John Roberts");
}
```

```javascript
  let last  = s.substring(7);
}

if (s === "Sonia Sotomayor") {
  console.log("John Roberts");
}
```

```cpp
/*          C++ Version          */
string s = "Elena Kagan";
s += ", joined " + to_string(2010);

char first = s[0];
char last  = s[s.length() - 1]

if (s.find("e") != string::npos) {
  string first = s.substr(0, 5);
  string last  = s.substr(7);
}

if (s == "Sonia Sotomayor") {
  cout << "John Roberts" << endl;
}
```

```python
"""       Python Version        """
s = "Elena Kagan"
s += ", joined " + str(2010)

first = s[0]
last  = s[-1]

if 'e' in s:
    first = s[0:5]
    last  = s[7:]

if s == "Sonia Sotomayor":
    print("John Roberts")
```

```java
/*          Java Version          */
String s = "Elena Kagan";
s += ", joined " + 2010;

char first = s.charAt(0);
char last  = s.charAt(s.length());

if (s.indexOf("e") != -1) {
  String first = s.substring(0, 5);
  String last  = s.substring(7);
}

if (s.equals("Sonia Sotomayor")) {
  System.out.println("John Roberts");
}
```

```javascript
  console.log("John Roberts");
}
```

The find member function returns the index of the given pattern if it exists, and the verbosely-named constant string::npos otherwise. This pattern kinda sorta is like the "in" keyword from Python.

```cpp
/*          C++ Version          */
string s = "Elena Kagan";
s += ", joined " + to_string(2010);

char first = s[0];
char last  = s[s.length() - 1]

if (s.find("e") != string::npos) {
  string first = s.substr(0, 5);
  string last  = s.substr(7);
}

if (s == "Sonia Sotomayor") {
  cout << "John Roberts" << endl;
}
```

```python
"""      Python Version      """
s = "Elena Kagan"
s += ", joined " + str(2010)

first = s[0]
last  = s[-1]

if 'e' in s:
    first = s[0:5]
    last  = s[7:]


if s == "Sonia Sotomayor":
    print("John Roberts")
```

```java
/*          Java Version          */
String s = "Elena Kagan";
s += ", joined " + 2010;

char first = s.charAt(0);
char last  = s.charAt(s.length());

if (s.indexOf("e") != -1) {
  String first = s.substring(0, 5);
  String last  = s.substring(7);
}

if (s.equals("Sonia Sotomayor")) {
  System.out.println("John Roberts");
}
```

```javascript
//    JavaScript Version
  console.log("John Roberts");
}
```

You can get substrings by using the .substr member function. If you give two parameters, the first is a start index, and the second is a **length**, not an end index.

```cpp
/*          C++ Version          */
string s = "Elena Kagan";
s += ", joined " + to_string(2010);

char first = s[0];
char last  = s[s.length() - 1]

if (s.find("e") != string::npos) {
  string first = s.substr(0, 5);
  string last  = s.substr(7);
}

if (s == "Sonia Sotomayor") {
  cout << "John Roberts" << endl;
}
```

```python
"""       Python Version       """
s = "Elena Kagan"
s += ", joined " + str(2010)

first = s[0]
last  = s[-1]

if 'e' in s:
    first = s[0:5]
    last  = s[7:]


if s == "Sonia Sotomayor":
    print("John Roberts")
```

```java
/*          Java Version          */
String s = "Elena Kagan";
s += ", joined " + 2010;

char first = s.charAt(0);
char last  = s.charAt(s.length());

if (s.indexOf("e") != -1) {
  String first = s.substring(0, 5);
  String last  = s.substring(7);
}

if (s.equals("Sonia Sotomayor")) {
  System.out.println("John Roberts");
}
```

```javascript
//    JavaScript Version
```

You can compare strings for equality using ==. If you're coming from Python, great! This will feel normal. If you're coming from Java, hopefully this will be a welcome relief.

```cpp
/*            C++ Version            */
string s = "Elena Kagan";
s += ", joined " + to_string(2010);

char first = s[0];
char last  = s[s.length() - 1]

if (s.find("e") != string::npos) {
  string first = s.substr(0, 5);
  string last  = s.substr(7);
}

if (s == "Sonia Sotomayor") {
  cout << "John Roberts" << endl;
}
```

```python
"""          Python Version          """
s = "Elena Kagan"
s += ", joined " + str(2010)

first = s[0]
last  = s[-1]

if 'e' in s:
    first = s[0:5]
    last  = s[7:]


if s == "Sonia Sotomayor":
    print("John Roberts")
```

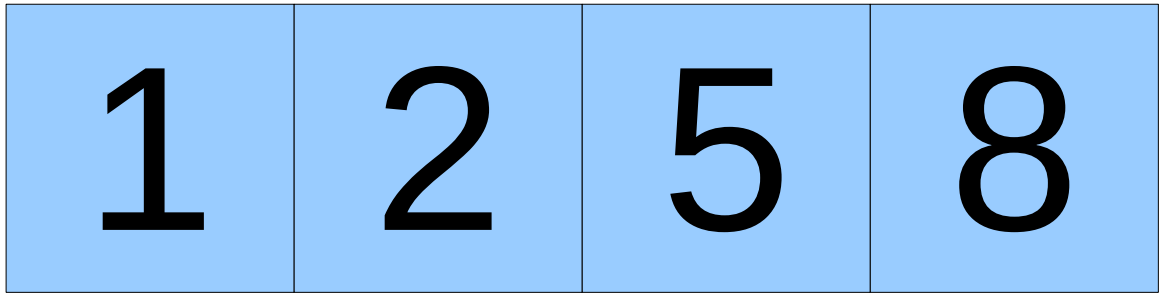
```java
/*           Java Version            */
String s = "Elena Kagan";
s += ", joined " + 2010;

char first = s.charAt(0);
char last  = s.charAt(s.length());

if (s.indexOf("e") != -1) {
  String first = s.substring(0, 5);
  String last  = s.substring(7);
}

if (s.equals("Sonia Sotomayor")) {
  System.out.println("John Roberts");
}
```

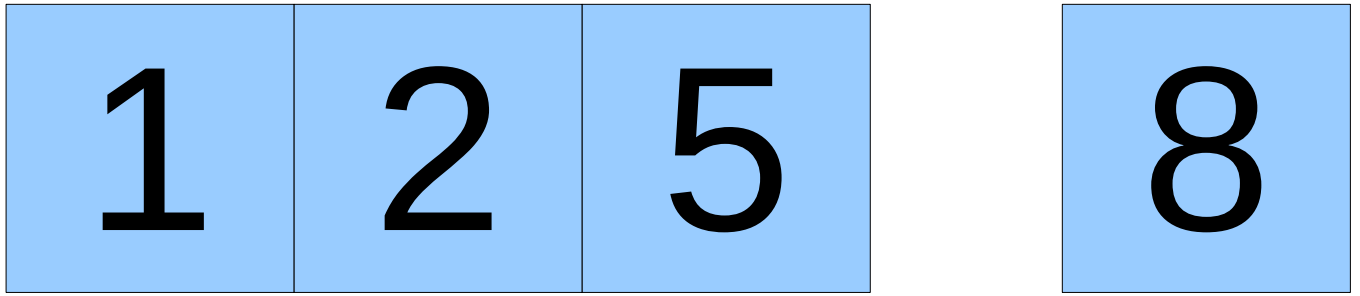
```javascript
                             ion
                          0;

                          - 1];

                          -1) {
                          ing(0, 5);
  let last  = s.substring(7);
}

if (s === "Sonia Sotomayor") {
  console.log("John Roberts");
}
```

You can print strings the same way you print anything else.

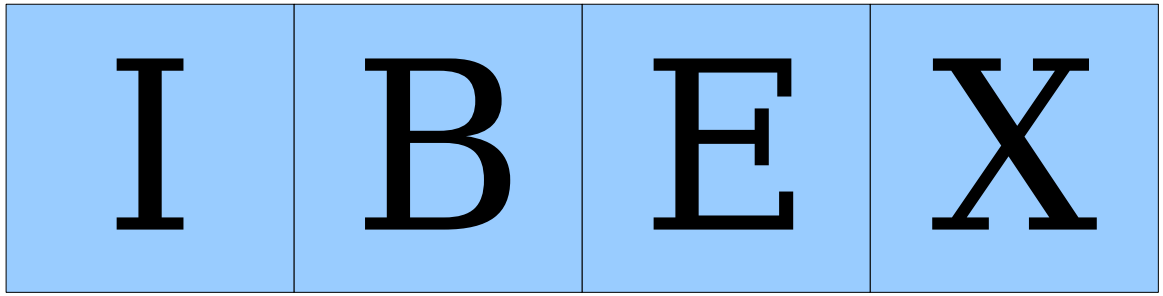
Recursion and Strings

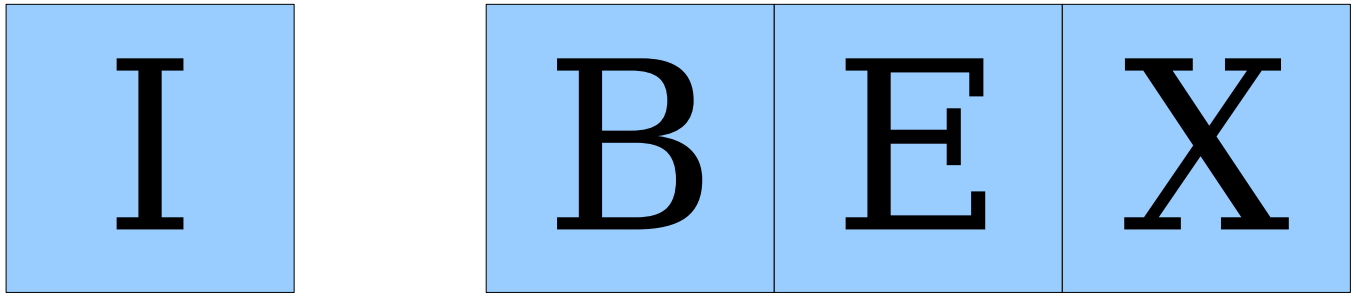
Thinking Recursively





Thinking Recursively



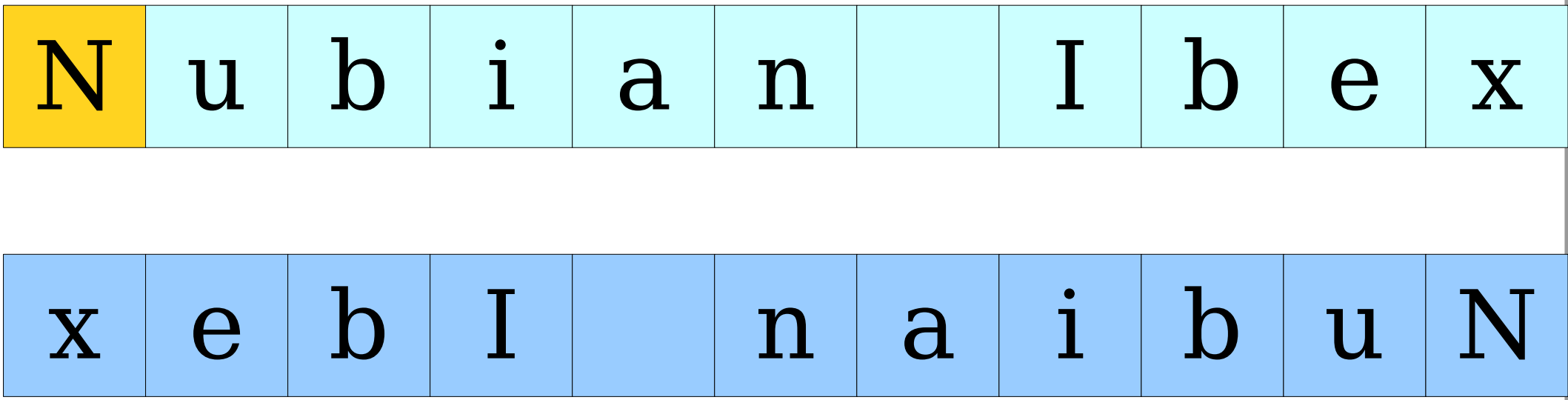


Reversing a String

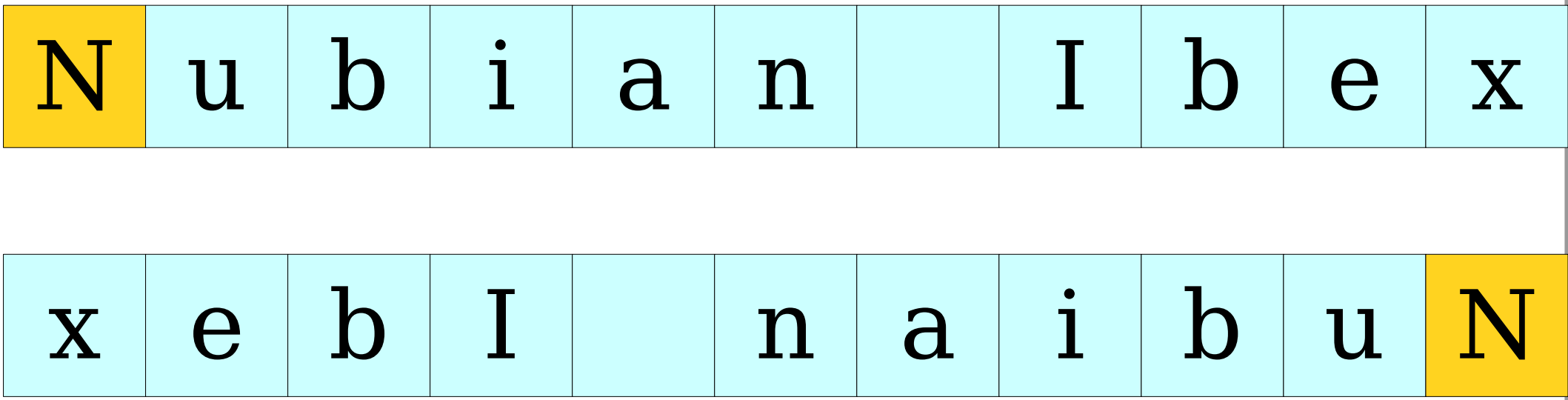
N	u	b	i	a	n		I	b	e	x

x	e	b	I		n	a	i	b	u	N

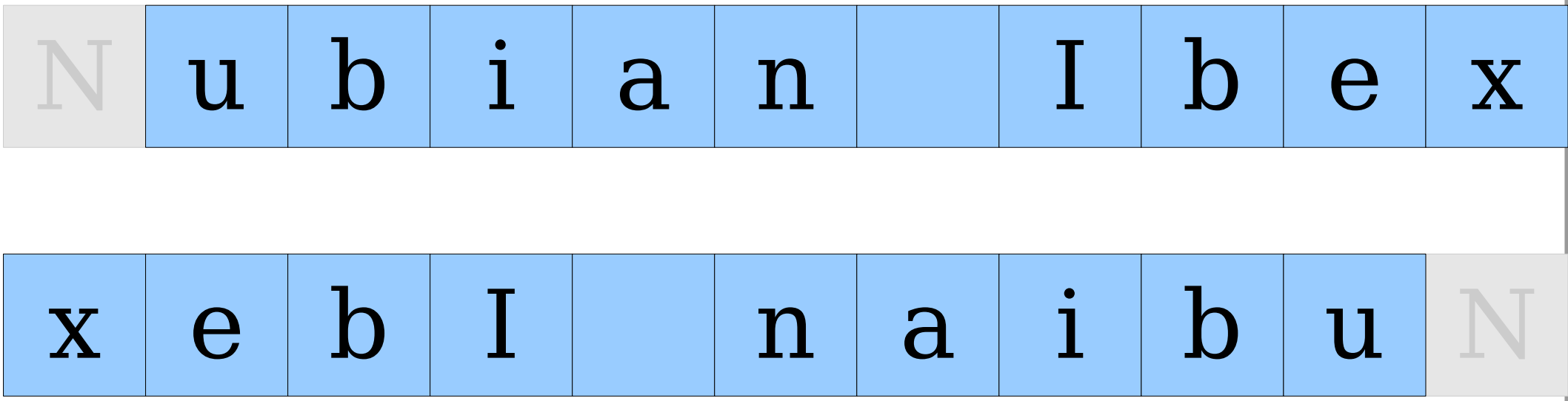
Reversing a String



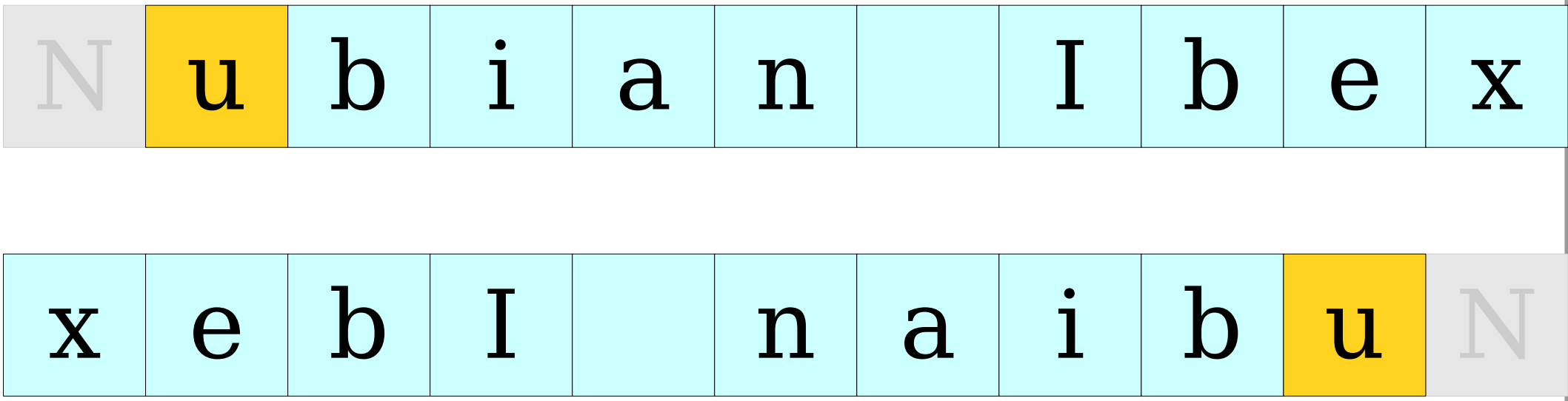
Reversing a String



Reversing a String



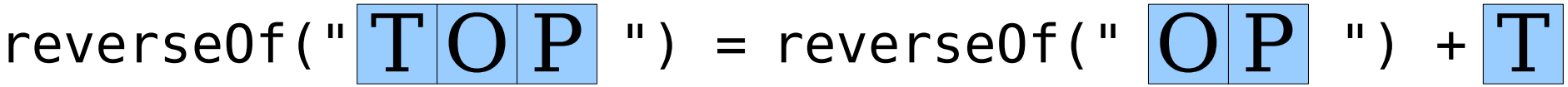
Reversing a String



Reversing a String Recursively

reverseOf("TOP ")

Reversing a String Recursively

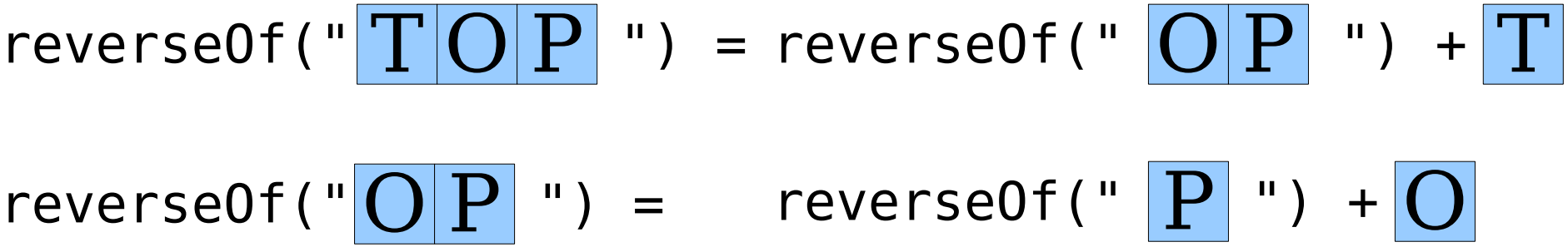


Reversing a String Recursively

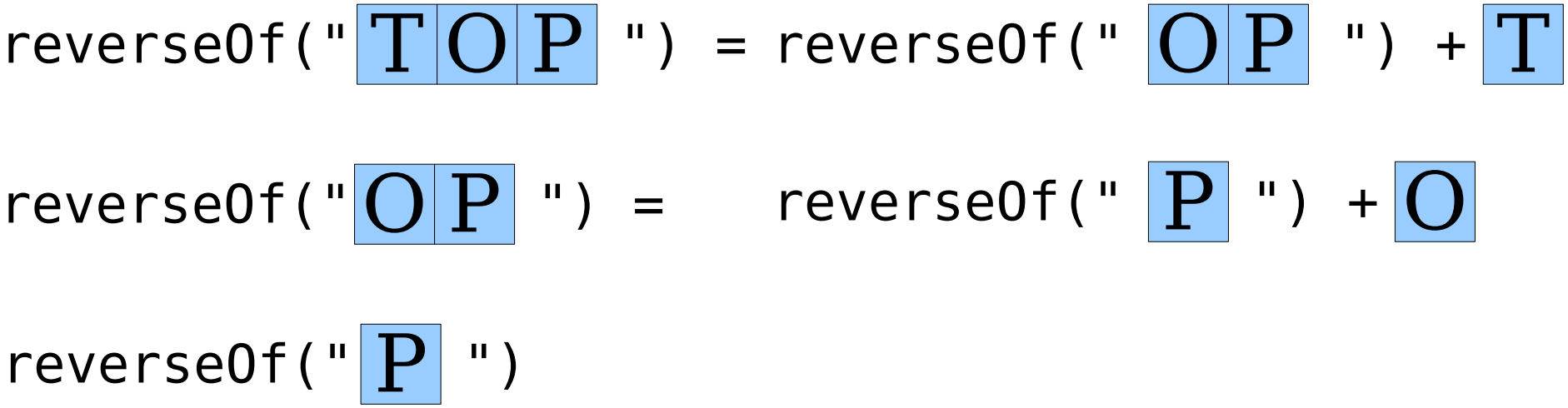
reverseOf("TOP") = reverseOf("OP") + T

reverseOf("OP")

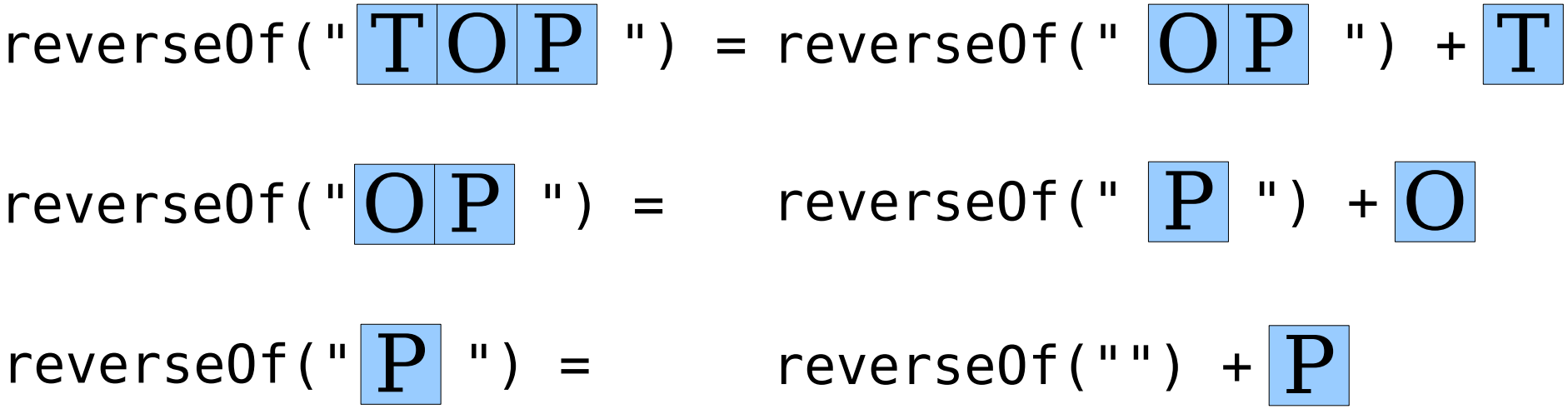
Reversing a String Recursively



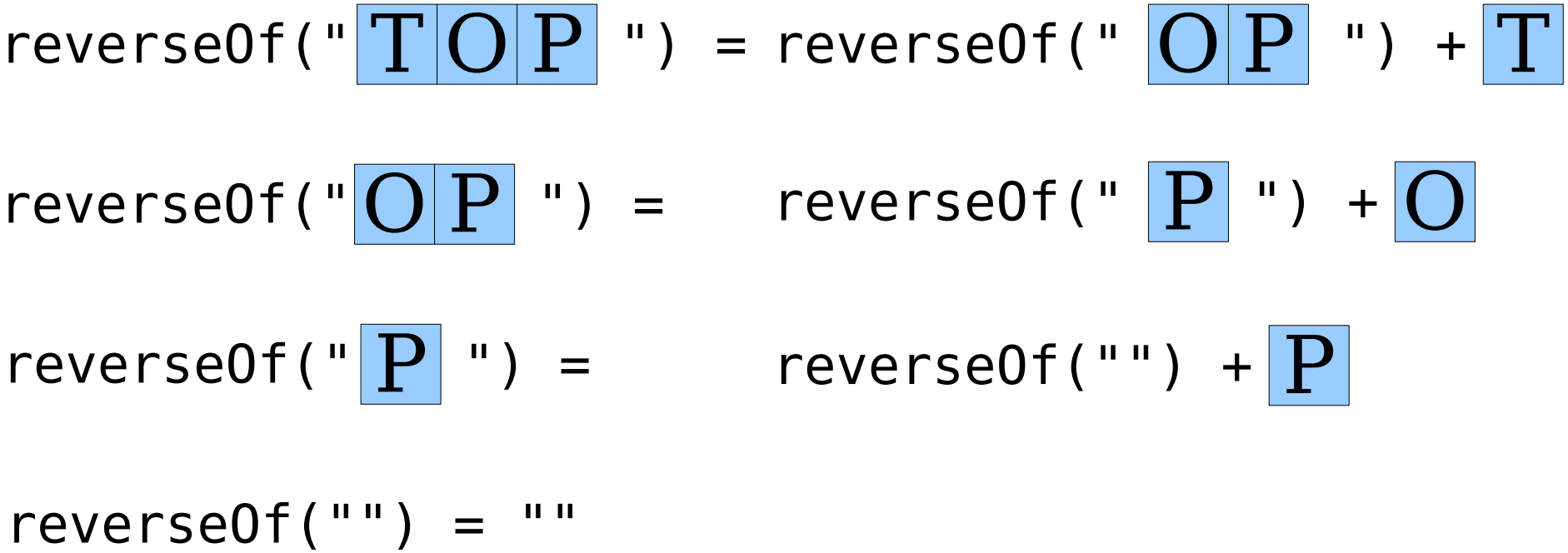
Reversing a String Recursively



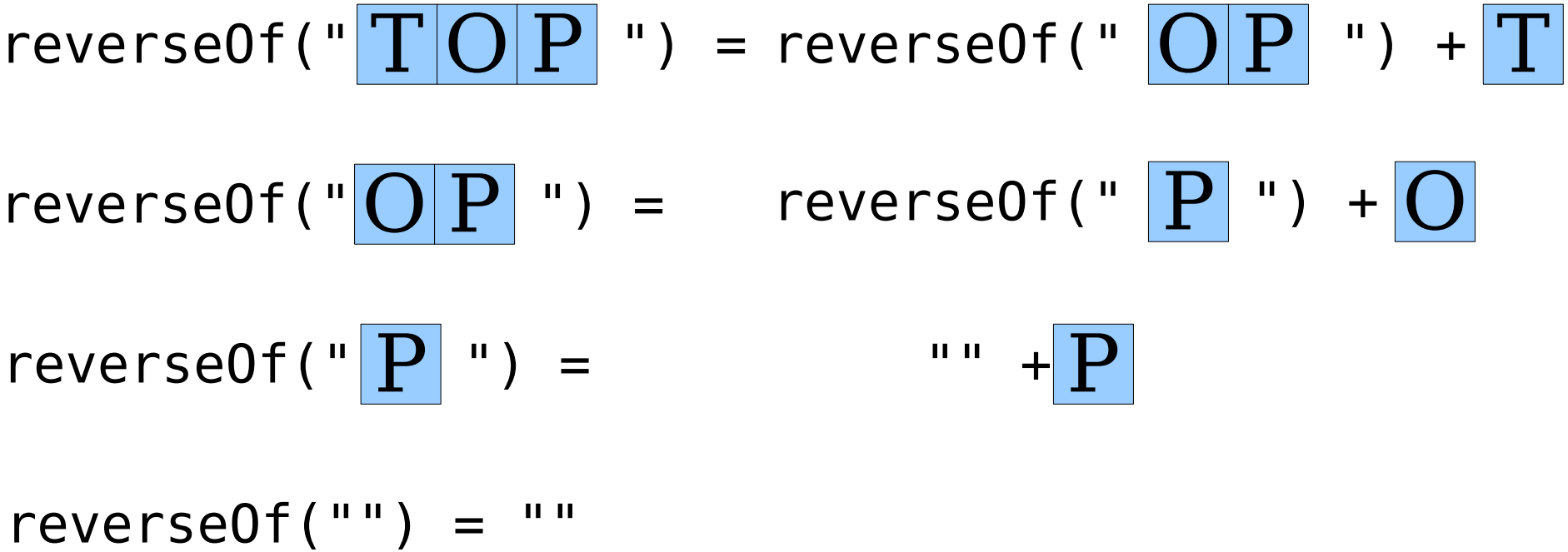
Reversing a String Recursively



Reversing a String Recursively



Reversing a String Recursively



Reversing a String Recursively

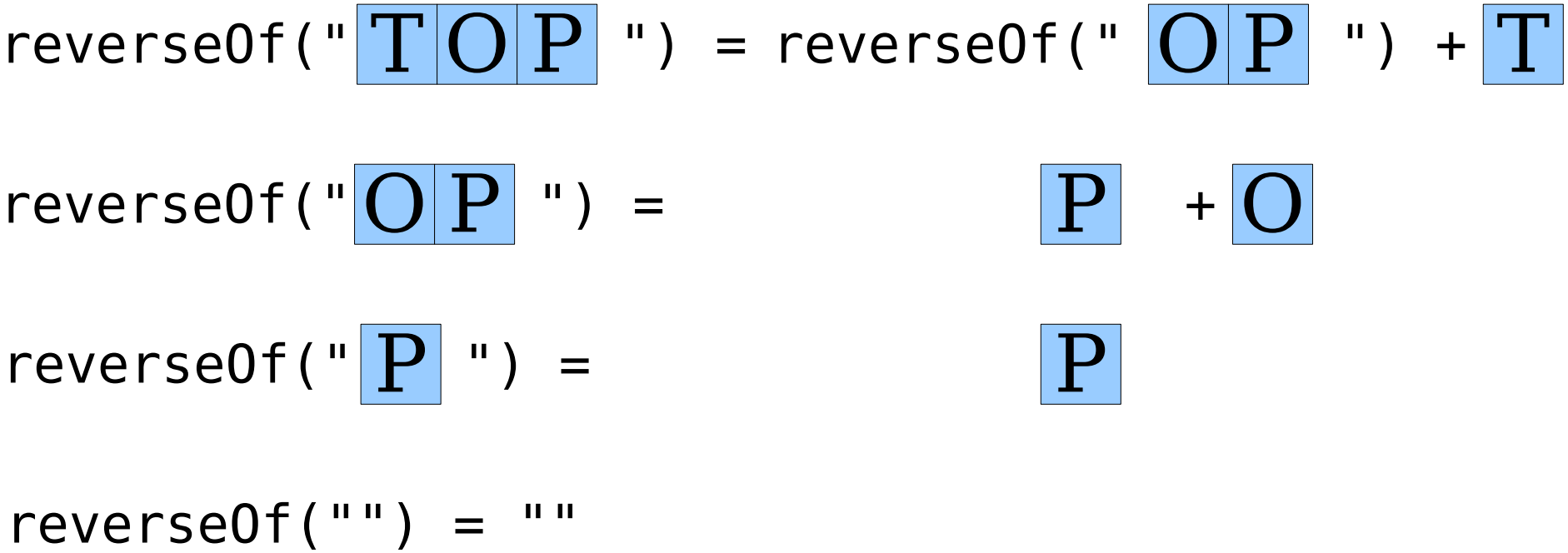
reverseOf("TOP") = reverseOf("OP") + T

reverseOf("OP") = reverseOf("P") + O

reverseOf("P") = P

reverseOf("") = ""

Reversing a String Recursively



Reversing a String Recursively

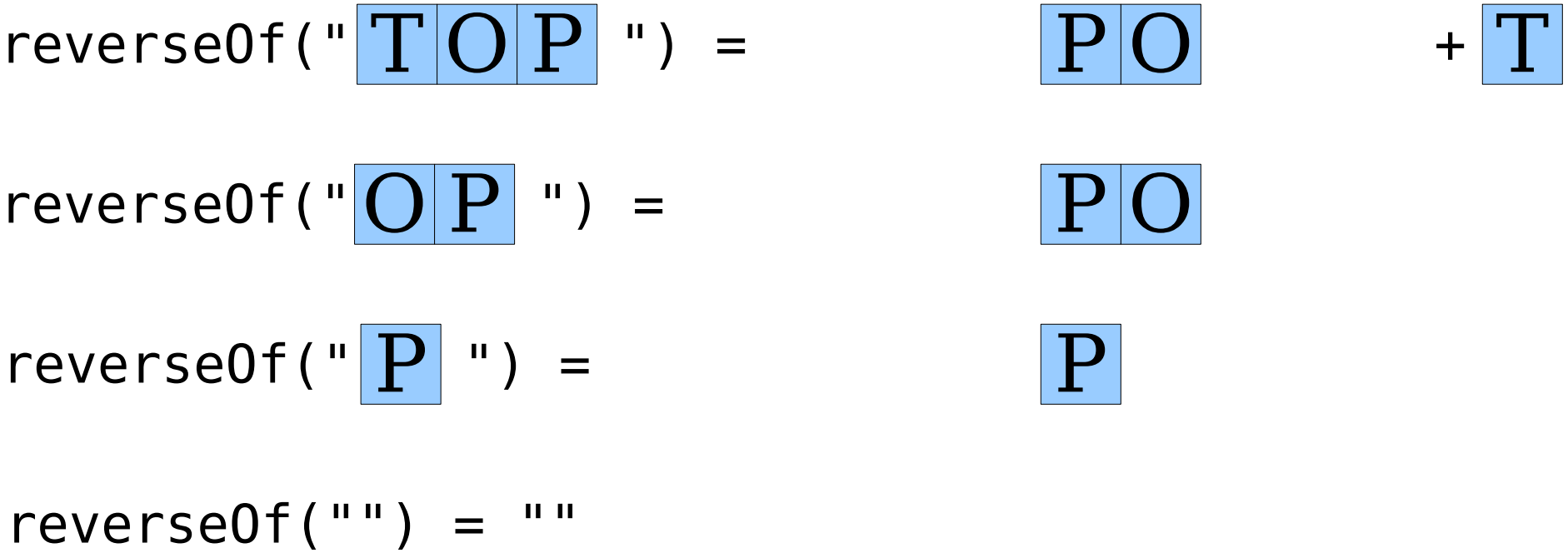
reverseOf("TOP") = reverseOf("OP") + T

reverseOf("OP") = PO

reverseOf("P") = P

reverseOf("") = ""

Reversing a String Recursively



Reversing a String Recursively

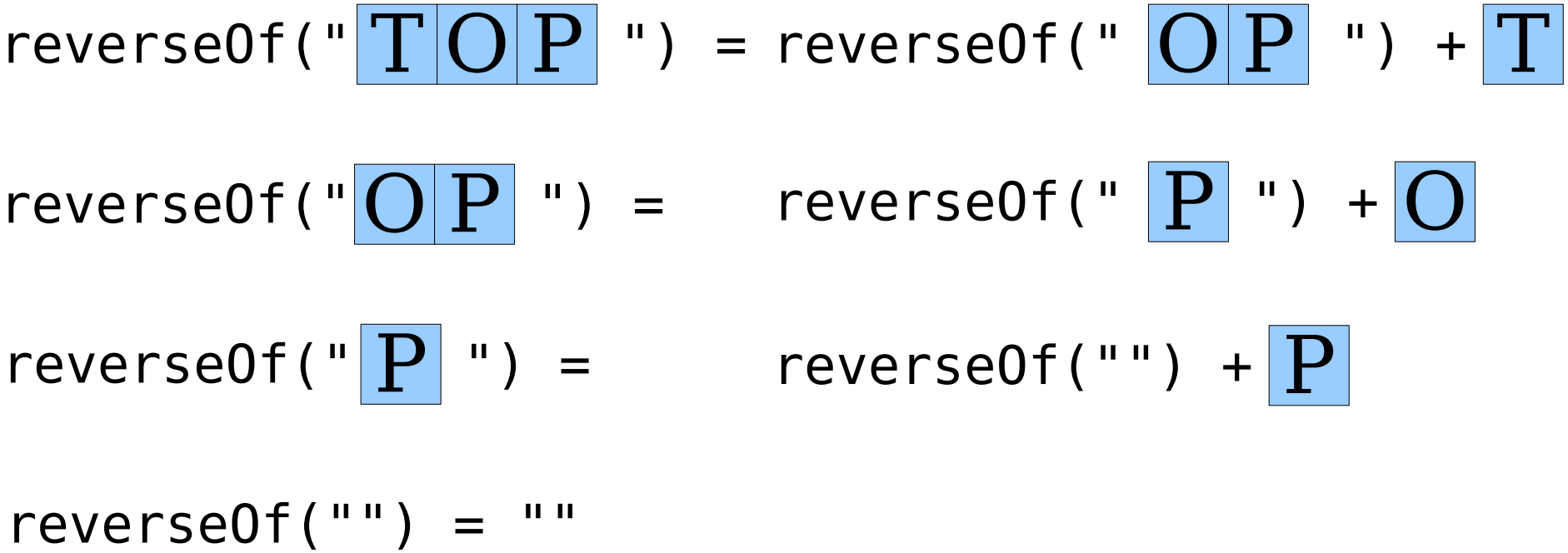
reverseOf("TOP") = POT

reverseOf("OP") = PO

reverseOf("P") = P

reverseOf("") = ""

Reversing a String Recursively



Thinking Recursively

```
if (The problem is very simple) {
   Directly solve the problem.
   Return the solution.
 } else {
   Split the problem into one or more
   smaller problems with the same
   structure as the original.
   Solve each of those smaller problems.
   Combine the results to get the overall
   solution.
   Return the overall solution.
}
```

These simple cases are called *base cases*.

These are the *recursive cases*.

Recap from Today

- Recursion works by identifying
  - one or more ***base cases***, simple cases that can be solved directly, and
  - one or more ***recursive cases***, where a larger problem is turned into a smaller one.
- C++ strings have some endearing quirks compared to other languages. Importantly, they're mutable.
- Recursion is everywhere! And you can use it on strings.

Your Action Items

- ***Read Chapter 3.***
  - This chapter is all about strings and string processing, and it has some real winners.
- ***Start working on Assignment 1.***
  - Aim to complete Stack Overflows and one or two of the recursion problems by Monday.

Next Time

- ***The `Vector` Type***
  - Storing sequences in C++!
- ***Recursion on `Vectors`.***
  - Of course. ☺