Strings and Streams

Recap from Last Time

Recursion on Numbers

- Here's a recursive function that computes $n!$:

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

- Here's a recursive implementation of a function to compute the sum of the digits of a number:

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

New Stuff!

Digital Roots Revisited

- Here's some code to compute the digital root of a number:

```
int digitalRootOf(int n) {
    while (n >= 10) {
        n = sumOfDigitsOf(n);
    }
    return n;
}
```

- How might we write this recursively?

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 (problem is sufficiently simple) {
   Directly solve the problem.
   Return the solution.
 } else {
   Split the problem up 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.
}
```

Strings in C++

Strings

- A ***string*** is a (possibly empty) sequence of characters.
- Strings in C++ are conceptually similar to strings in Java.
- There are several minor differences, like
  - different names for similar methods and
  - different behavior for similar methods
- And some really major differences.
  - There are two types of strings in C++.

C++ Strings

- C++ strings are represented with the `string` type.
- To use `string`, you must

```
#include <string>
```

  at the top of your program.
- You can get the number of characters in a string by calling

```
str.length()
```

- You can read a single character in a string by writing

```
str[index]
```

- Despite the above syntax, C++ strings are not arrays; it's just a convenient syntactic shortcut.

Operations on Characters

- In C++, the header `<cctype>` contains a variety of useful functions that you can apply to characters.
- The following functions check whether a character is of a given type:

```
isalpha  isdigit
isalnum  islower  isupper
isspace  ispunct
```

Strings are Mutable

- Unlike Java strings, C++ strings are mutable and can be modified.
- To change an individual character of a string, write

```
str[index] = ch;
```

- To append more text, you can write

```
str += text;
```

- These operations directly change the string itself, rather than making a copy of the string.

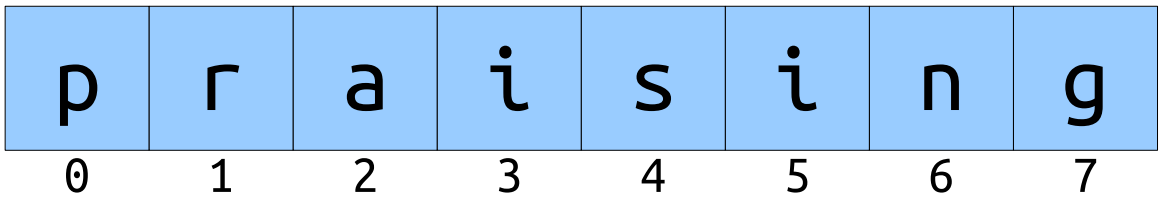
Other Important Differences

- In C++, the == operator can directly be used to compare strings:

```
if (str1 == str2) {
    /* strings match */
}
```

- You can get a substring of a string by calling the `substr` method. `substr` takes in a start position and optional *length* (not an end position!)

```
string allButFirstChar    = str.substr(1);
```



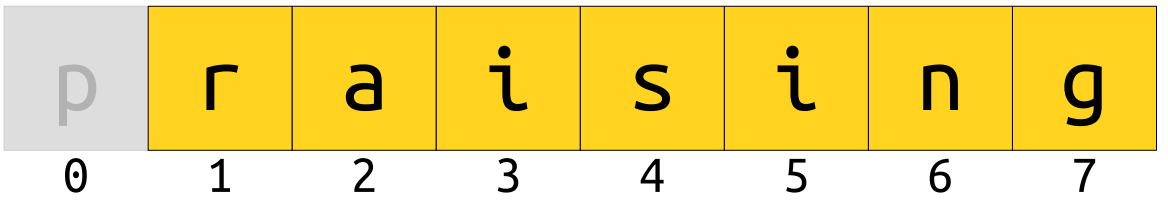
Other Important Differences

- In C++, the == operator can directly be used to compare strings:

```
if (str1 == str2) {
    /* strings match */
}
```

- You can get a substring of a string by calling the `substr` method. `substr` takes in a start position and optional *length* (not an end position!)

```
string allButFirstChar    = str.substr(1);
```



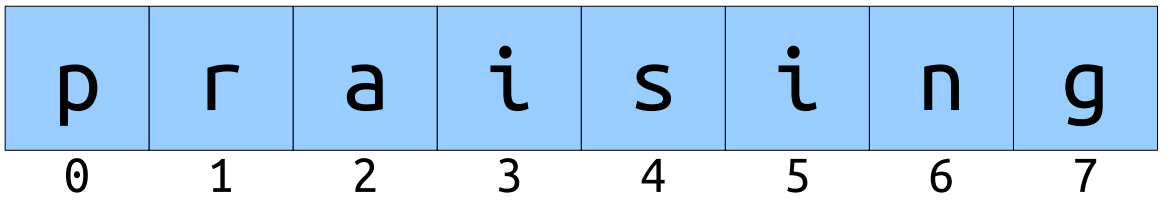
Other Important Differences

- In C++, the == operator can directly be used to compare strings:

```
if (str1 == str2) {
    /* strings match */
}
```

- You can get a substring of a string by calling the `substr` method. `substr` takes in a start position and optional *length* (not an end position!)

```
string allButFirstChar    = str.substr(1);
string allButFirstAndLast = str.substr(1, str.length() - 2);
```



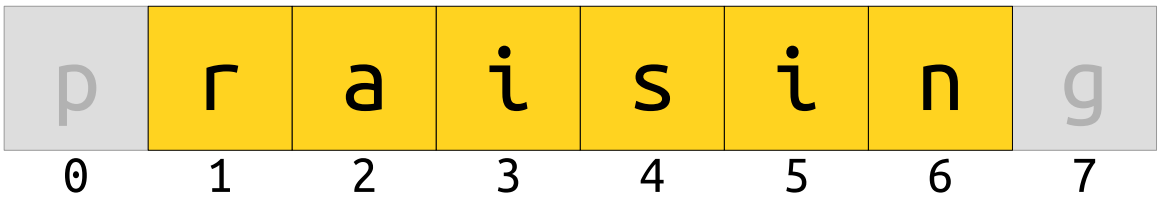
Other Important Differences

- In C++, the == operator can directly be used to compare strings:

```
if (str1 == str2) {
    /* strings match */
}
```

- You can get a substring of a string by calling the `substr` method. `substr` takes in a start position and optional *length* (not an end position!)

```
string allButFirstChar    = str.substr(1);
string allButFirstAndLast = str.substr(1, str.length() - 2);
```



Even More Differences

- In Java, you can concatenate just about anything with a string.
- In C++, you can only concatenate strings and characters onto other strings.
- We provide a library "`strlib.h`" to make this easier.

```
string s = "He really likes " + integerToString(137);
```

And the Biggest Difference

- In C++, there are two types of strings:
  - C-style strings, inherited from the C programming language, and
  - C++ `strings`, a library implemented in C++.
- Any string literal is a C-style string.
- Almost none of the operations we've just described work on C-style strings.
- Takeaway point: Be careful with string literals in C++.
  - Use the `string` type whenever possible.

```
string s = "Nubian " + "ibex";
```

```
string s = "Nubian " + "ibex";
```

Each of these strings is a C-style string, and C-style strings cannot be added with +. This code doesn't compile.

```
string s = "Nubian " + "ibex";
```

```
string s = string("Nubian ") + "ibex";
```

```
string s = string("Nubian ") + "ibex";
```

Now that we explicitly add a cast from a C-style string to a C++-style string, this code is legal. If you need to perform concatenations like this ones, make sure to cast at least one of the string literals to a C++ string.

Time-Out for Announcements!

Assignment 1

- ***Assignment 1: Welcome to C++*** goes out today. It's due on Monday, January 23rd at the start of class.
  - Play around with C++ and the Stanford libraries!
  - Get some practice with recursion.
  - Explore the debugger!
  - Teach the computer to read, sorta. ☺
- LaIR hours begin on *Monday* because of the national holiday.

The CS106B Grading Scale

++

+

✓+

✓

✓-

-

--

0

Assignment Grading

- You will receive two scores: a functionality score and a style score.
- The ***functionality score*** is based on correctness.
  - Do your programs produce the correct output?
  - Do they work on all legal 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.

Late Days

- Everyone has ***two*** free “late days” to use as needed.
- A “late day” is an automatic extension for one *class period* (Monday to Wednesday, Wednesday to Friday, or Friday to Monday).
- If you need an extension beyond late days, please talk to Anton. Your section leader cannot grant extensions.

Section Signups

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

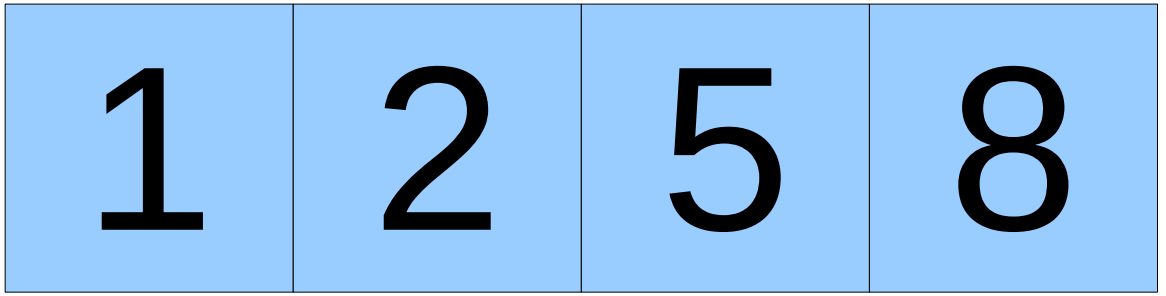
  http://cs198.stanford.edu/section

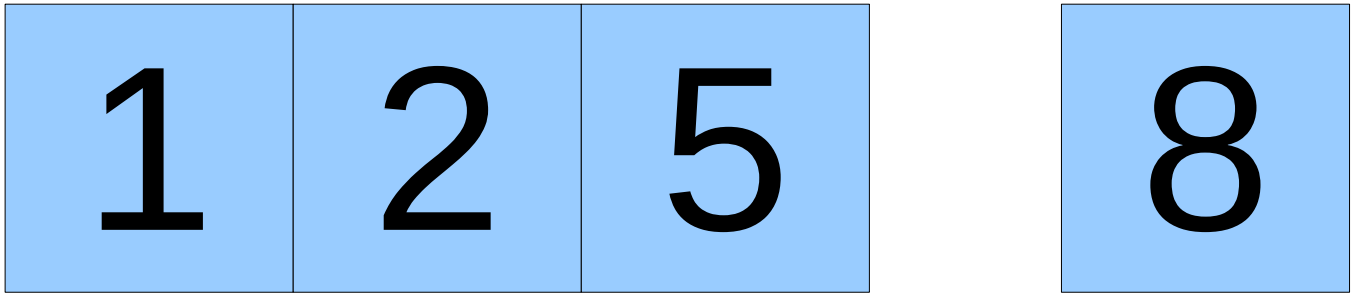
- Link available on the CS106B course website.

One More Unto the Breach!

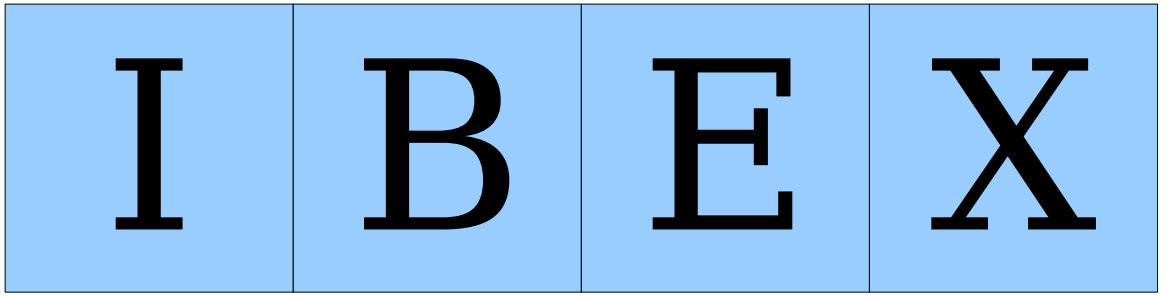
Recursion and Strings

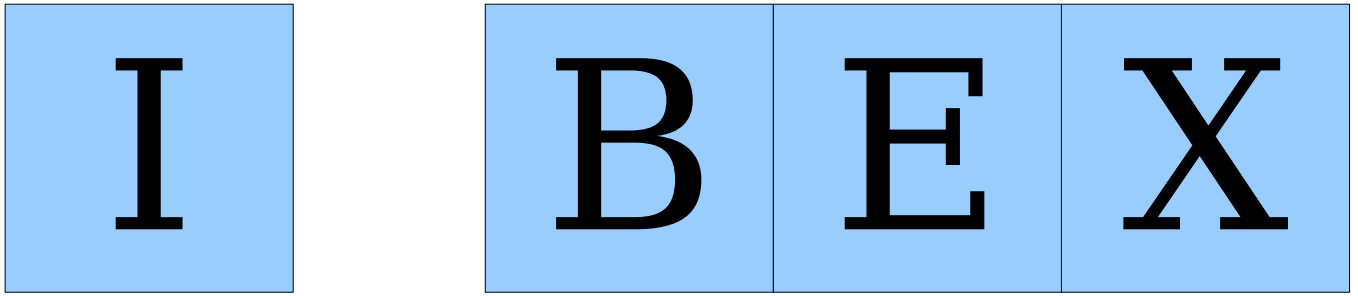
Thinking Recursively



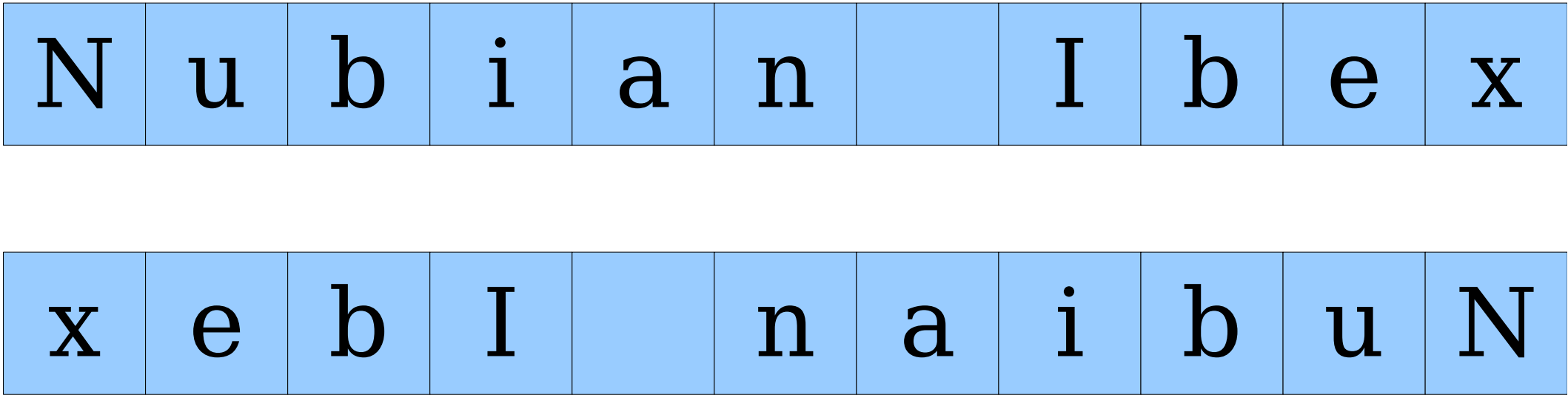


Thinking Recursively

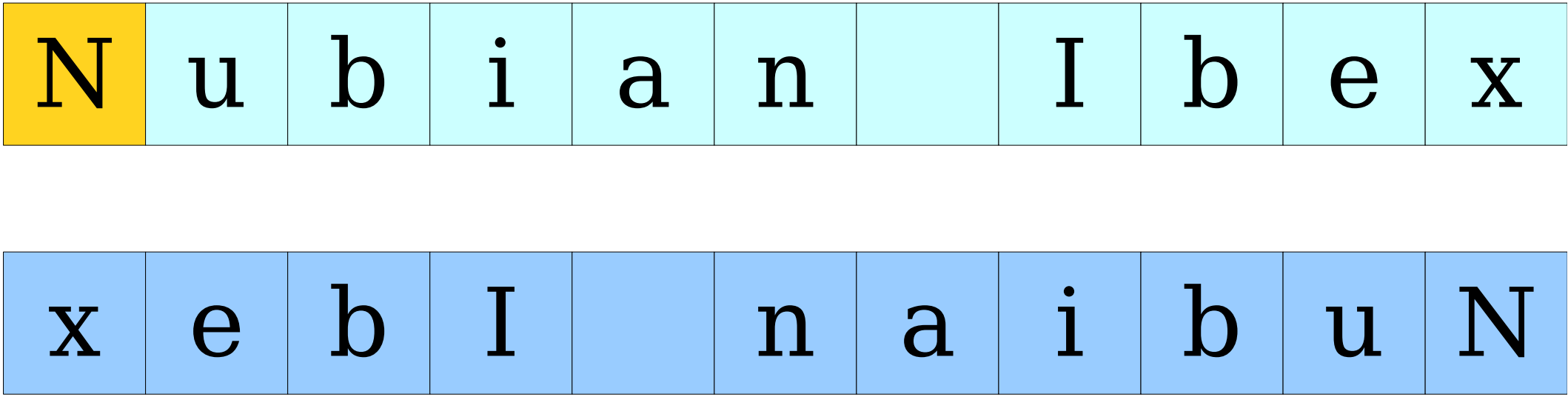




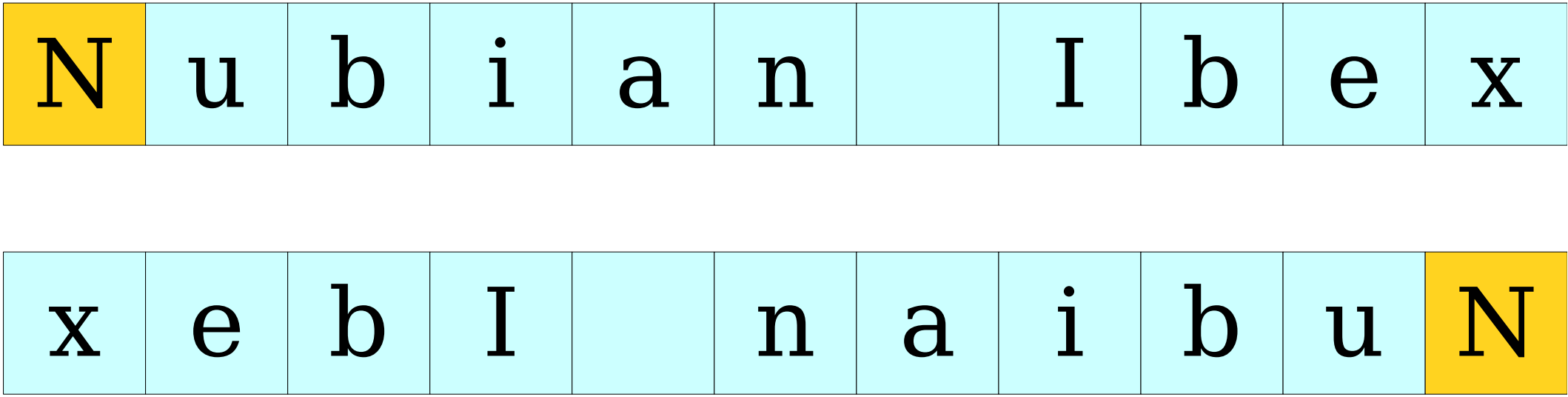
Reversing a String



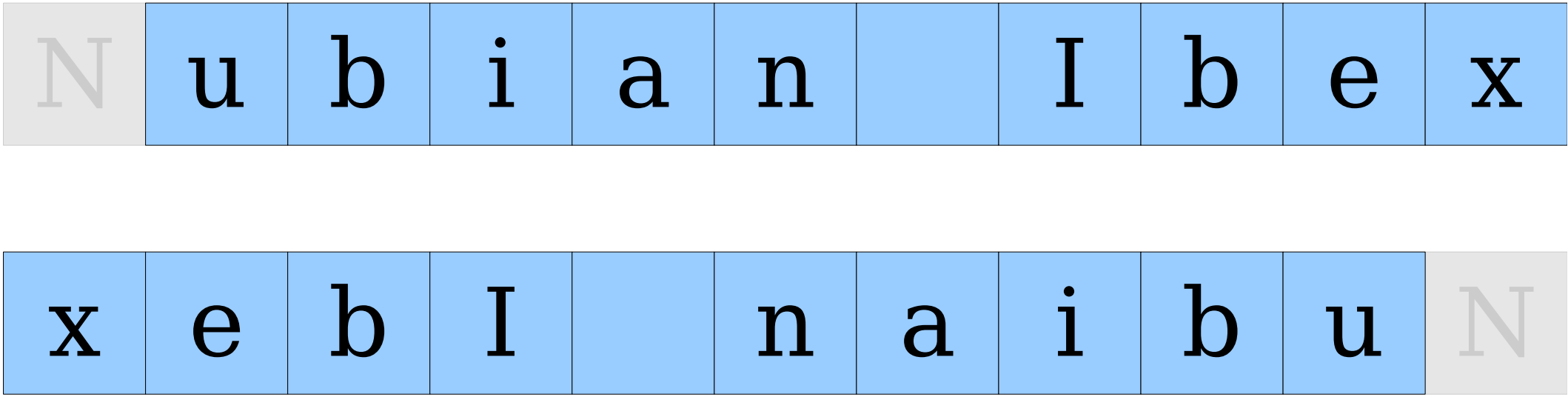
Reversing a String



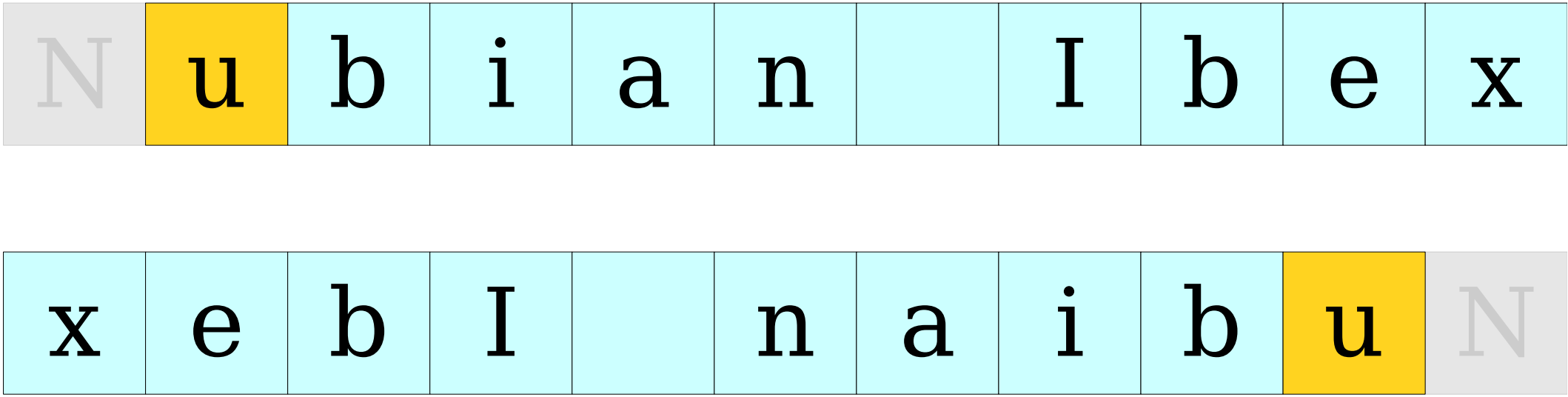
Reversing a String



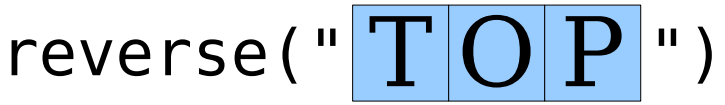
Reversing a String



Reversing a String

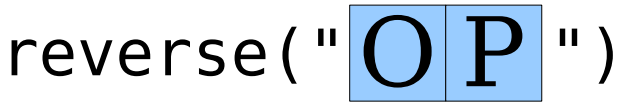


Reversing a String Recursively



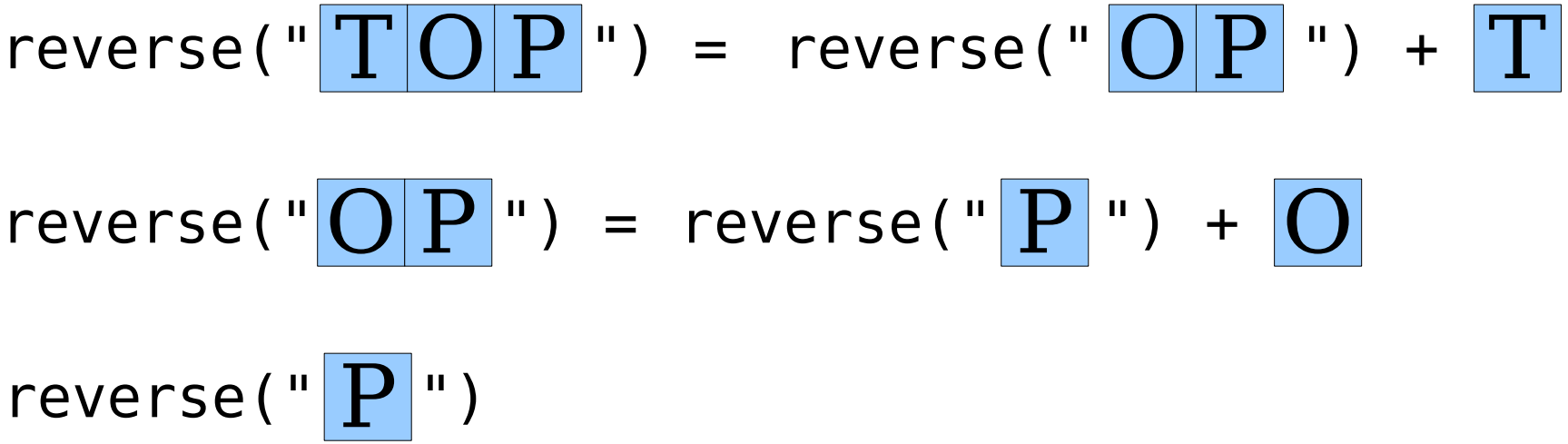
Reversing a String Recursively

Reversing a String Recursively

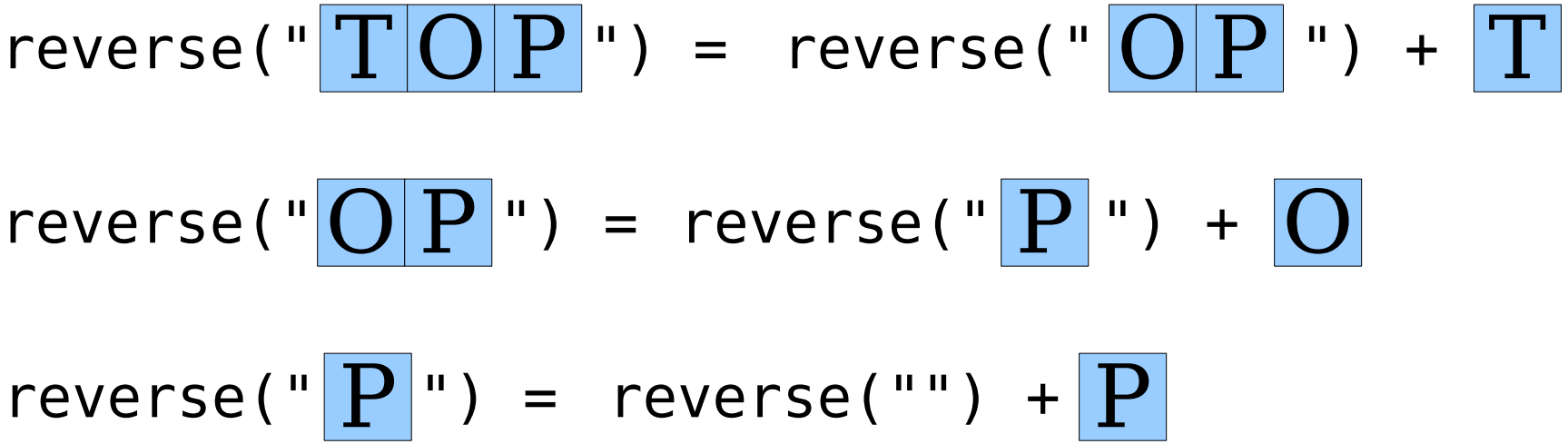


Reversing a String Recursively

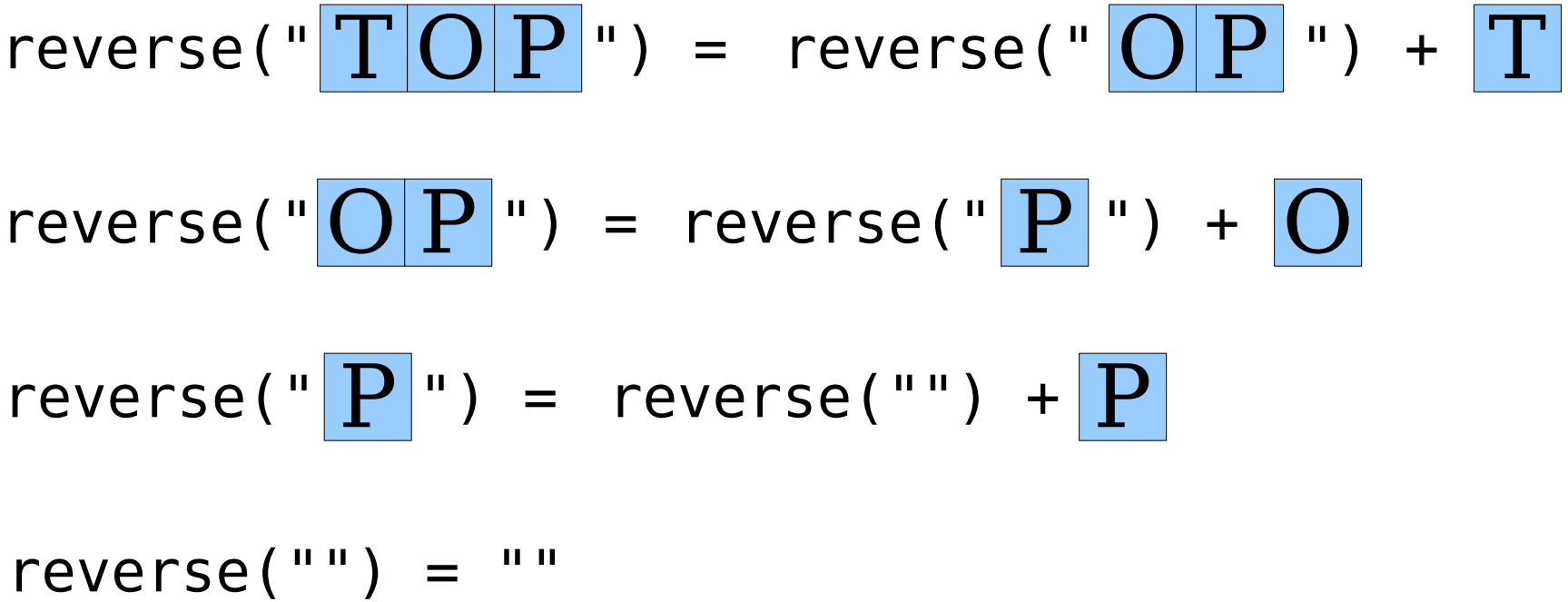
Reversing a String Recursively



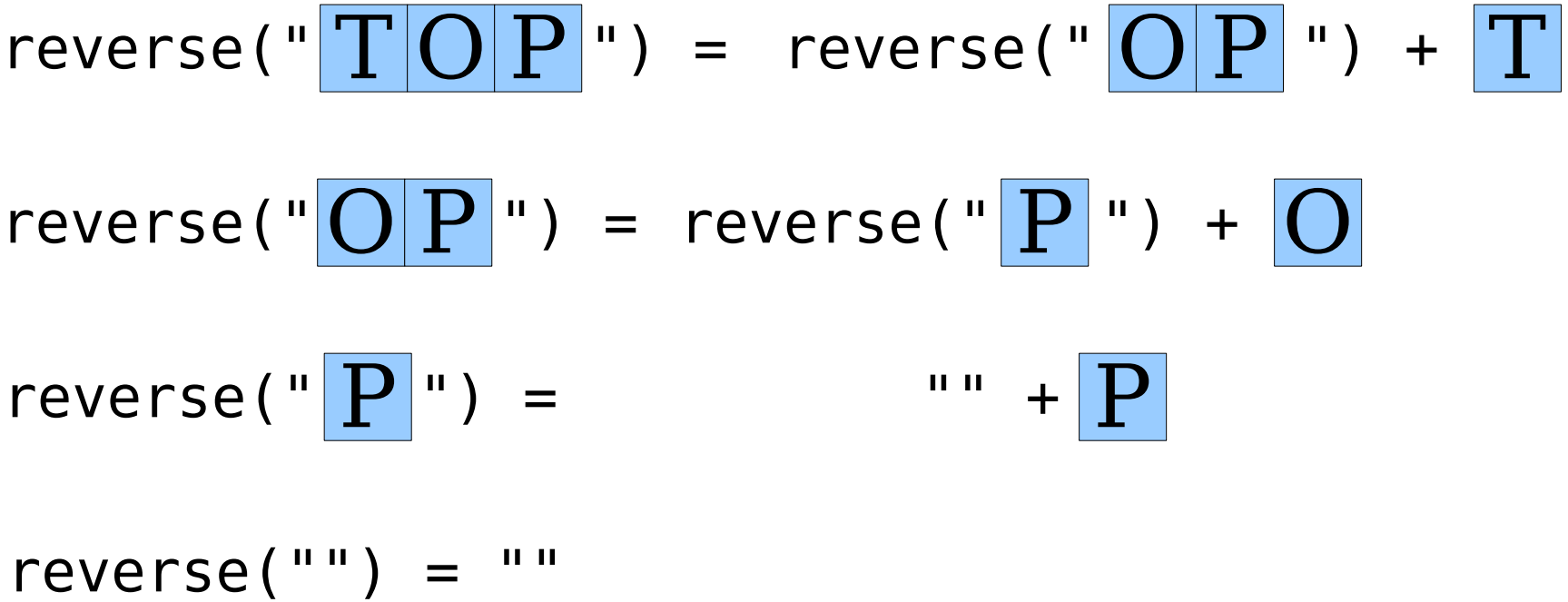
Reversing a String Recursively



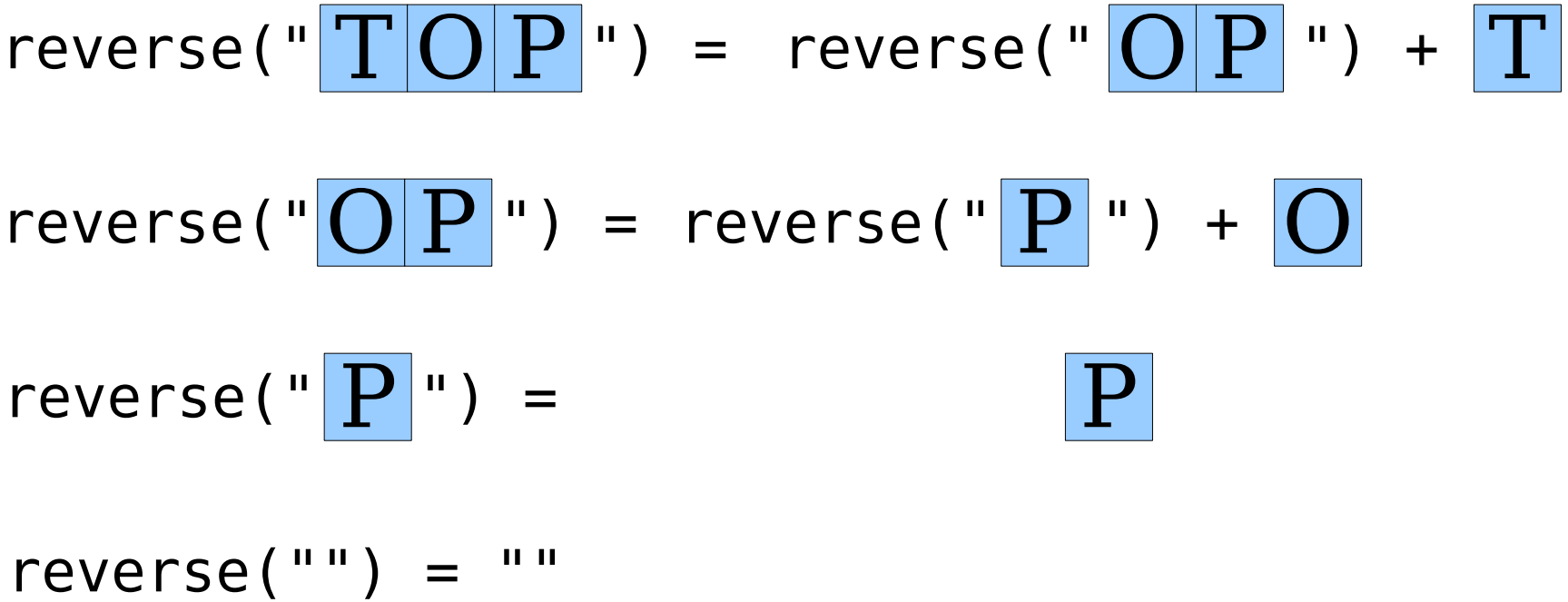
Reversing a String Recursively



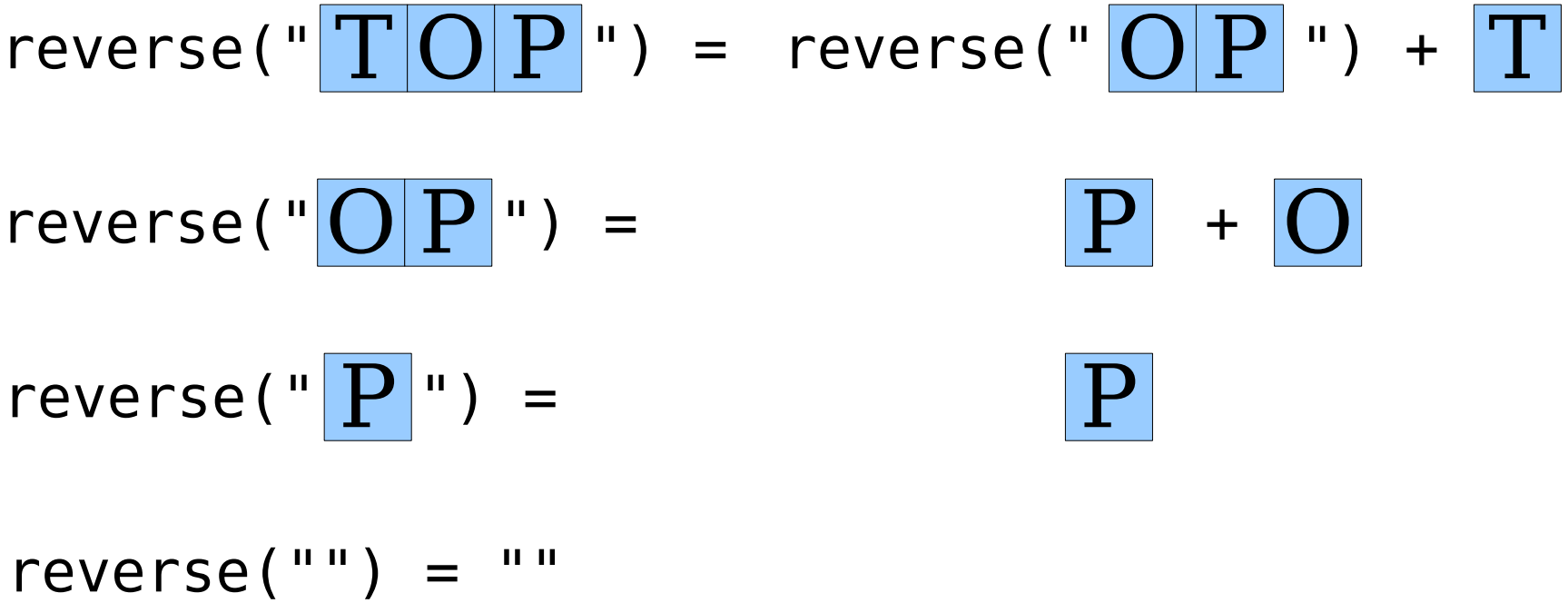
Reversing a String Recursively



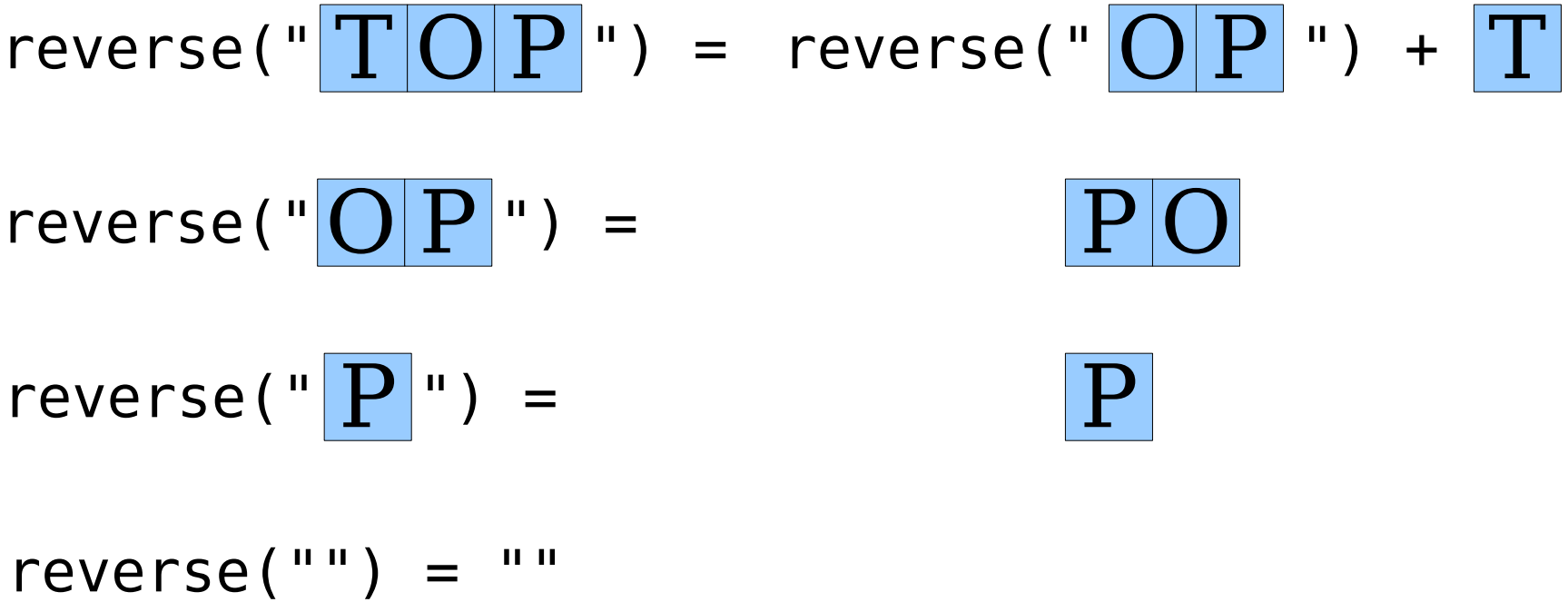
Reversing a String Recursively



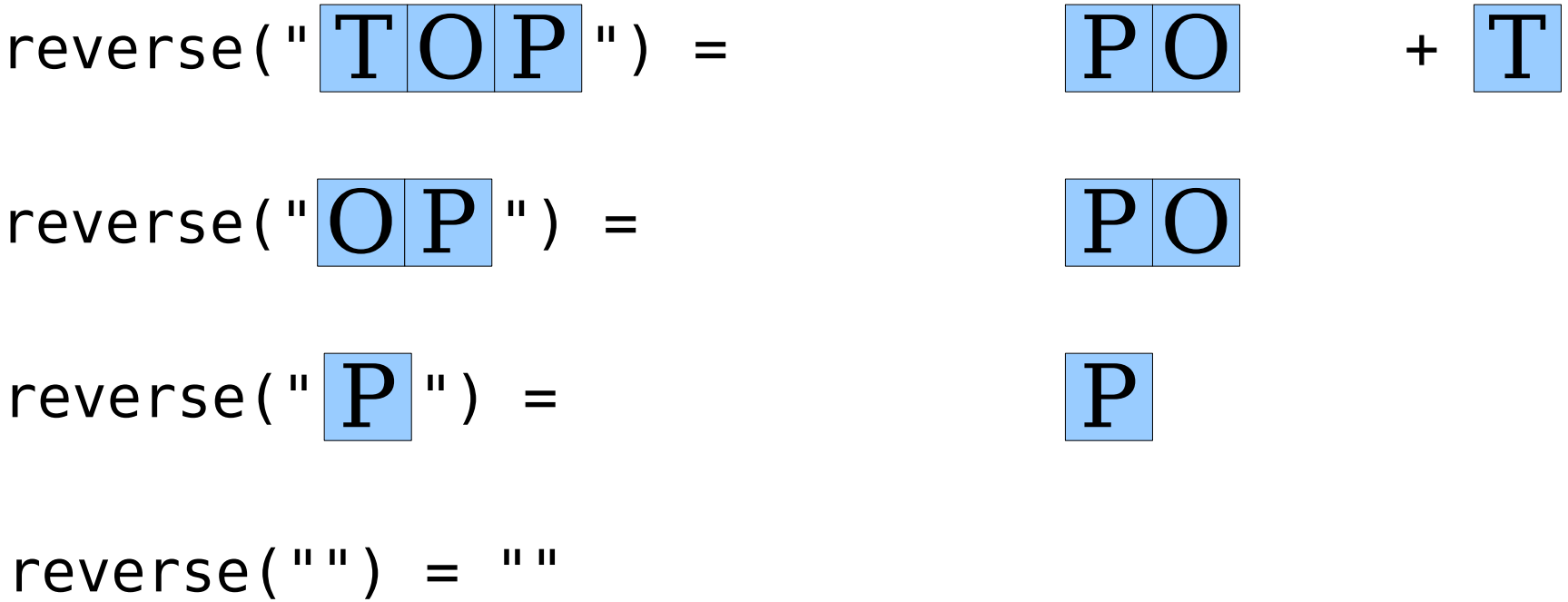
Reversing a String Recursively



Reversing a String Recursively



Reversing a String Recursively



Reversing a String Recursively

reverse("TOP") = POT

reverse("OP") = PO

reverse("P") = P

reverse("") = ""

C++ Streams

Getting Data from Files

- File reading in C++ is a lot easier than file reading in Java.
- To open and read from a file in C++, you use the `ifstream` (`i`nput `f`ile `stream`) class.
- It's exported by the `<fstream>` header.

Reading Line by Line

- You can read a line out of an `ifstream` by using the **`getline`** function:

```
getline(file, str)
```

- (Notice this is all lower case, in contrast to the `getLine` function from `simpio.h`).
- The canonical "read each line of a file loop" is shown here:

```
string line;
while (getline(file, line)) {
    /* … process line … */
}
```

- Chapter 4 of the course reader has more details about file I/O in C++; highly recommended!

Recap from Today

- Recursion is everywhere!
- C++ strings are mutable, look like arrays, and have a slightly different syntax than Java Strings.
- There are two kinds of C++ strings, string objects and C-style strings. C-style strings are Hairy and Scary.
- Recursion applies to strings just as it does everything else!
- File reading in C++ isn’t that bad!

Your Action Items

- Read Chapter 3 and Chapter 4 of the textbook to learn more about strings and to get an intro to file processing.
- Start working on Assignment 1. Try to complete some of the earlier problems by our next class meeting.

Next Time

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