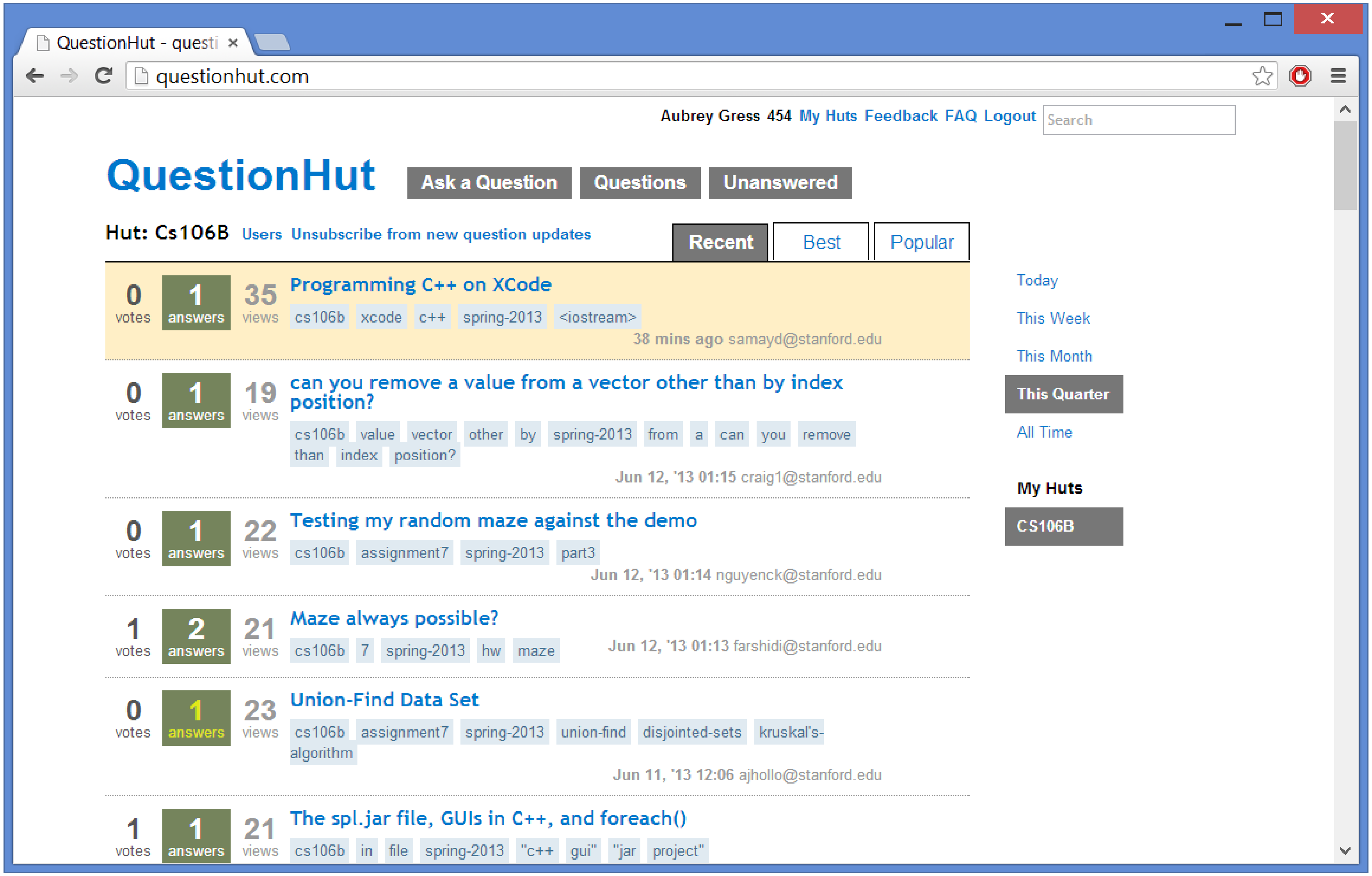
Strings and Streams

Question Hut!



QuestionHut - questi

questionhut.com

Aubrey Gress 454 My Huts Feedback FAQ Logout Search

QuestionHut Ask a Question Questions Unanswered

Hut: Cs106B Users Unsubscribe from new question updates

Recent Best Popular

votes	answers	views			
0	1	35	Programming C++ on XCode	cs106b xcode c++ spring-2013 <iostream>	38 mins ago samayd@stanford.edu
0	1	19	can you remove a value from a vector other than by index position?	cs106b value vector other by spring-2013 from a can you remove than index position?	Jun 12, '13 01:15 craig1@stanford.edu
0	1	22	Testing my random maze against the demo	cs106b assignment7 spring-2013 part3	Jun 12, '13 01:14 nguyenck@stanford.edu
1	2	21	Maze always possible?	cs106b 7 spring-2013 hw maze	Jun 12, '13 01:13 farshidi@stanford.edu
0	1	23	Union-Find Data Set	cs106b assignment7 spring-2013 union-find disjointed-sets kruskal's-algorithm	Jun 11, '13 12:06 ajhollo@stanford.edu
1	1	21	The spl.jar file, GUIs in C++, and foreach()	cs106b in file spring-2013 "c++ gui" "jar project"	

Today
This Week
This Month
This Quarter
All Time

My Huts
CS106B

Question Hut!

- Very useful!
- Great for general questions that other students may be having. For example:
  - “How does this code from lecture work?”
  - “When would I use a HashMap as opposed to a Vector?”
- Please don't post your code.

LAIR Hours Start Today

- Sunday-Wednesday, 7-11PM
- Great place to work on assignments

Bad Style #4

```
const int kSumMax = 10;
int sum;
int main() {
   sum = 0;
   for (int i = 0; i < kSumMax; i++) {
      sum += i;
   }
   cout << “Sum:” << sum;
   return 0;
}
```

Bad Style #4

```
const int kSumMax = 10;
int sum;
int main() {
    sum = 0;
    for (int i = 0; i < kSumMax; i++) {
        sum += i;
    }
    cout << “Sum:” << sum;
    return 0;
}
```

Global Variable!

Good Style #4

```
const int kSumMax = 10;
int main() {
    int sum = 0;
    for (int i = 0; i < kSumMax; i++) {
        sum += i;
    }
    cout << “Sum:” << sum;
    return 0;
}
```

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);
    }
}
```

Today

- C++ Strings
- Recursion with Strings
- Reading Files in C++
- Parameter Passing and Common Mistakes

Today

- C++ Strings
- Recursion with Strings
- Reading Files in C++
- Parameter Passing and Common Mistakes

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:
  - Different names for similar methods.
  - Different behavior for similar methods
- And some really major differences:
  - 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()
```

C++ Strings

- 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.
- Change an individual character:

str [***index***] = ***ch***

- Append more text:

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 search a string for some other string by using `find` (instead of `indexOf`). `find` returns `string::npos` instead of -1 if the string isn't found:

```
if (str1.find(str2) != string::npos) {
    /* found str2 inside str1 */
}
```

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

```
string allButFirstChar = str.substr(1);
string lastFiveChars = str.substr(str.length() - 5, 5);
```

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 = "I like " + integerToString(137);
```

And the Biggest Difference

- In C++, there are two types of strings:
  - C-style strings, inherited from the C programming language
  - C++ `string`s, a library implemented in C++.
- Any *string literal* is a C-style string.
- Most 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.

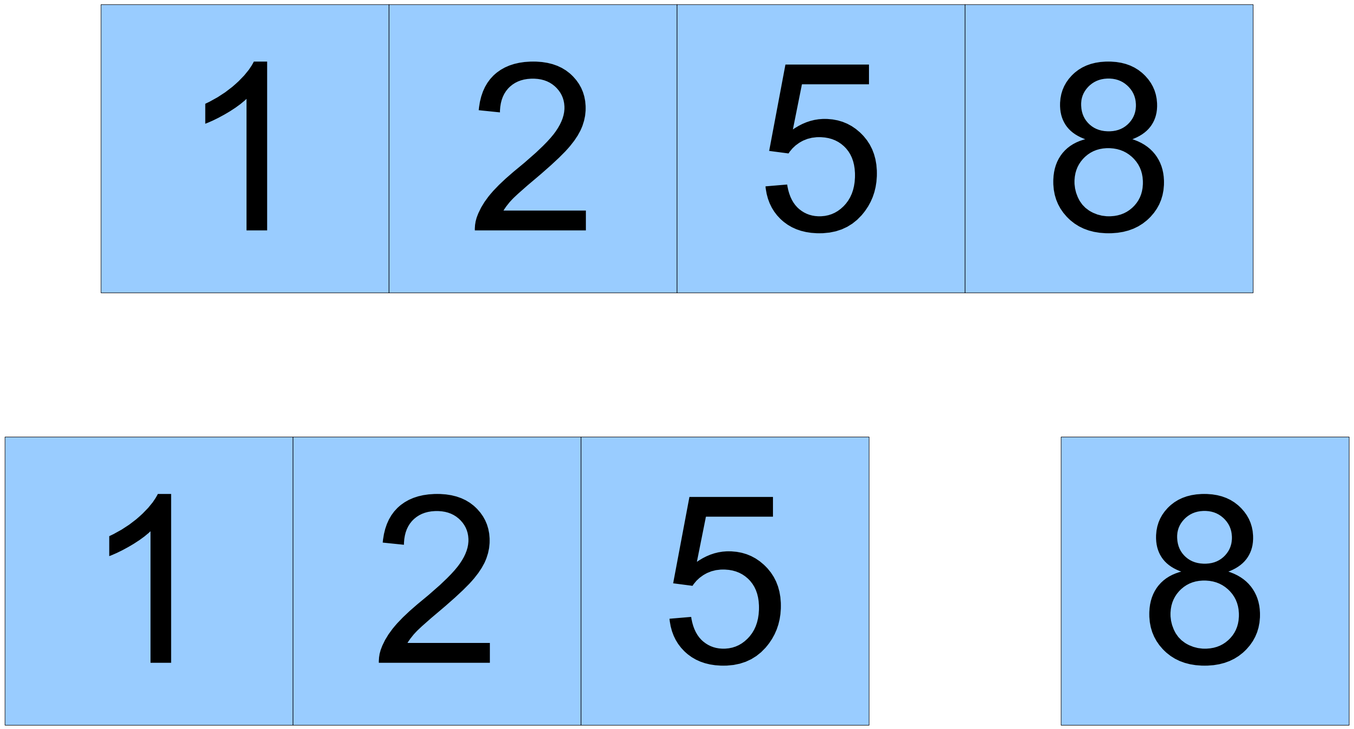
Today

- C++ Strings
- Recursion with Strings
- Reading Files in C++
- Parameter Passing and Common Mistakes

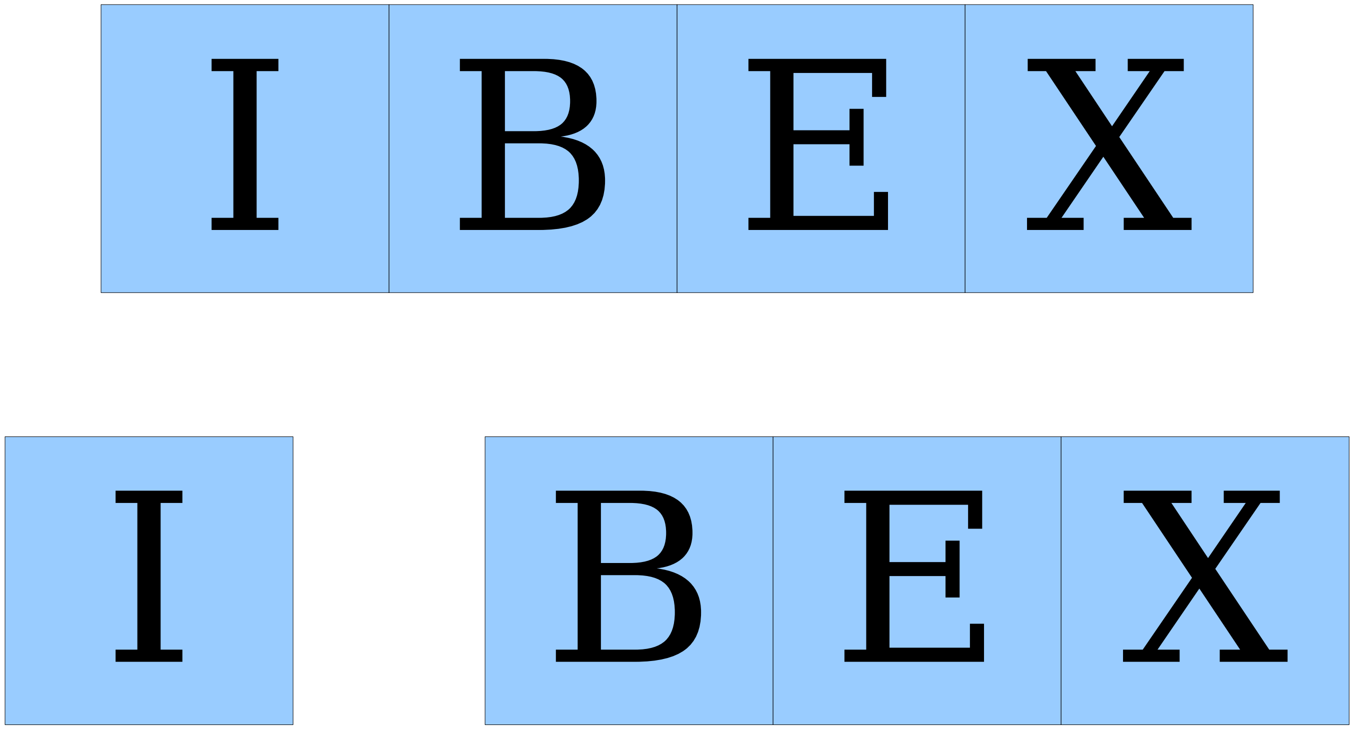
Thinking Recursively

```
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.
}
```

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

N	u	b	i	a	n		I	b	e	x

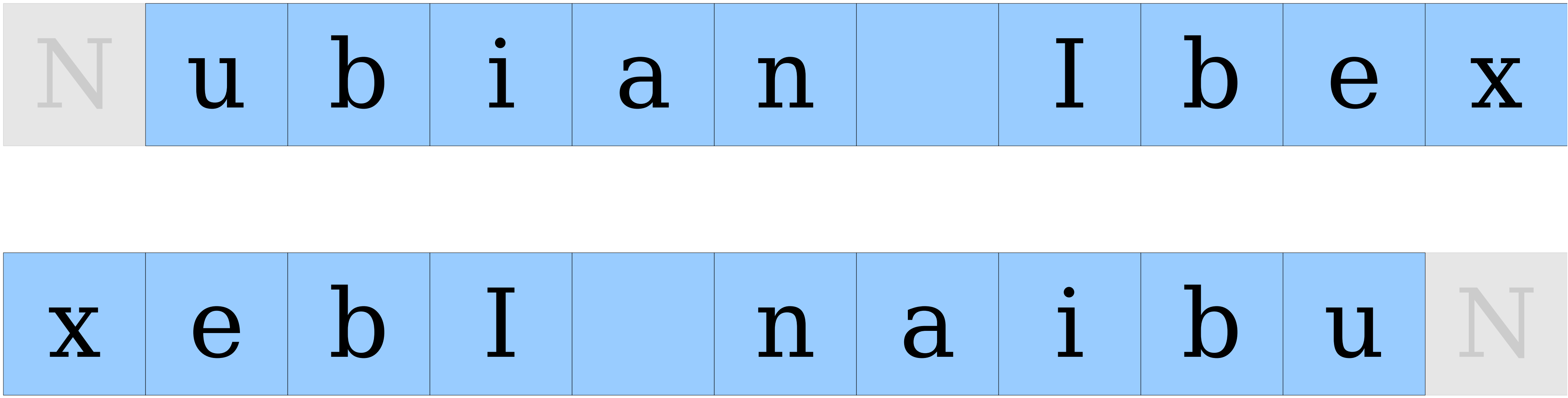
x	e	b	I		n	a	i	b	u	N

Reversing a String

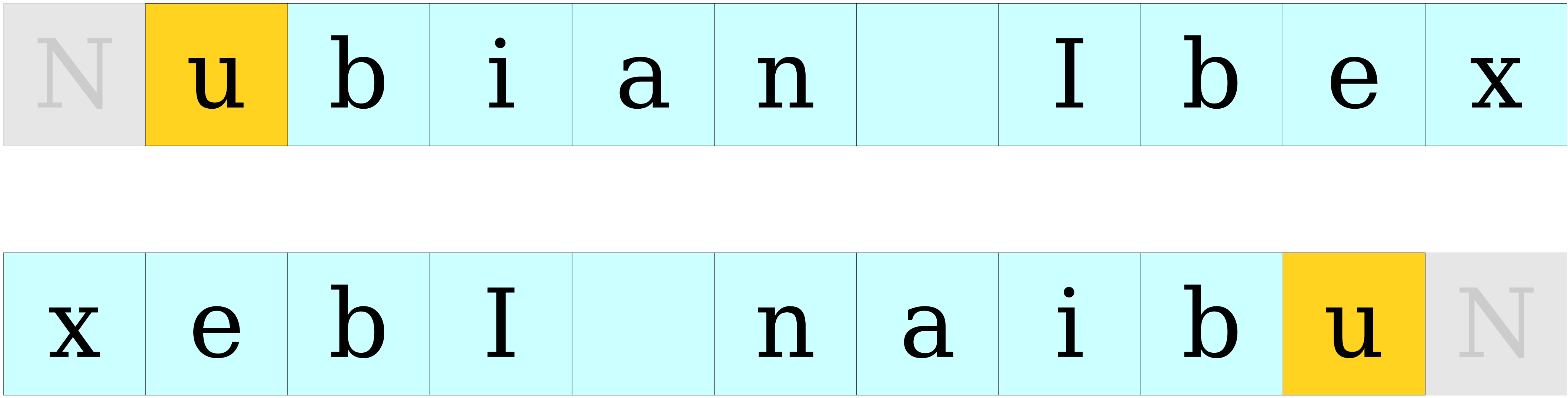
N	u	b	i	a	n		I	b	e	x

x	e	b	I		n	a	i	b	u	N

Reversing a String



Reversing a String



Reverse String: Iterative
reverse.cpp
(On Board)

Reversing a String Recursively

- Remember that every recursive algorithm has two components: the **base case** and the **recursive decomposition**
- What are these for *reverse string*?
  - Base Case: *“When is a string so simple that I already know it's reverse?”*
  - Recursive Decomposition: *“How can I 'shrink' the string to make forward progress?”*

Reversing a String Recursively

- Remember that every recursive algorithm has two components: the **base case** and the **recursive decomposition**
- What are these for *reverse string*?
  - Base Case: *“When is a string so simple that I already know it's reverse?”*
  - Recursive Decomposition: *“How can I 'shrink' the string to make forward progress?”*

Reverse String: Recursive (On Board)

Reversing a String Recursively

```
reverse("TOP")
```

Reversing a String Recursively

```
reverse("TOP") = reverse("OP") + T
```

Reversing a String Recursively

```
reverse("TOP") = reverse("OP") + T

reverse("OP")
```

Reversing a String Recursively

```
reverse("TOP") = reverse("OP") + T

reverse("OP") = reverse("P") + O
```

Reversing a String Recursively

```
reverse("TOP") = reverse("OP") + T

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

reverse("P")
```

Reversing a String Recursively

```
reverse("TOP") = reverse("OP") + T

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

reverse("P") = reverse("") + P
```

Reversing a String Recursively

```
reverse("TOP") = reverse("OP") + T

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

reverse("P") = reverse("") + P

reverse("") = ""
```

Reversing a String Recursively

reverse("$\boxed{T}\boxed{O}\boxed{P}$") = reverse("$\boxed{O}\boxed{P}$") + $\boxed{T}$

reverse("$\boxed{O}\boxed{P}$") = reverse("$\boxed{P}$") + $\boxed{O}$

reverse("$\boxed{P}$") = reverse("") + $\boxed{P}$

reverse("") = ""

Reversing a String Recursively

```
reverse("TOP") = reverse("OP") + T

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

reverse("P") =            "" + P

reverse("") = ""
```

Reversing a String Recursively

```
reverse("TOP") = reverse("OP") + T

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

reverse("P") =                 P

reverse("") = ""
```

Reversing a String Recursively

```
reverse("TOP") = reverse("OP") + T

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

reverse("P") =                 P

reverse("") = ""
```

Reversing a String Recursively

```
reverse("TOP") = reverse("OP") + T

reverse("OP") =             P + O

reverse("P") =              P

reverse("") = ""
```

Reversing a String Recursively

```
reverse("TOP") = reverse("OP") + T

reverse("OP") =              PO

reverse("P") =               P

reverse("") = ""
```

Reversing a String Recursively

```
reverse("TOP") = reverse("OP") + T

reverse("OP") =              PO

reverse("P") =               P

reverse("") = ""
```

Reversing a String Recursively

```
reverse("TOP") =        PO   + T

reverse("OP")  =        PO

reverse("P")   =        P

reverse("") = ""
```

Reversing a String Recursively

`reverse("TOP") =`	POT
`reverse("OP") =`	PO
`reverse("P") =`	P
`reverse("") = ""`	

Palindromes

- A palindrome is a string whose letters are the same forwards and backwards.
- For example:
    - Go hang a salami! I'm a lasagna hog.
    - Mr. Owl ate my metal worm.
    - Anne, I vote more cars race Rome to Vienna.

Recursive Palindromes

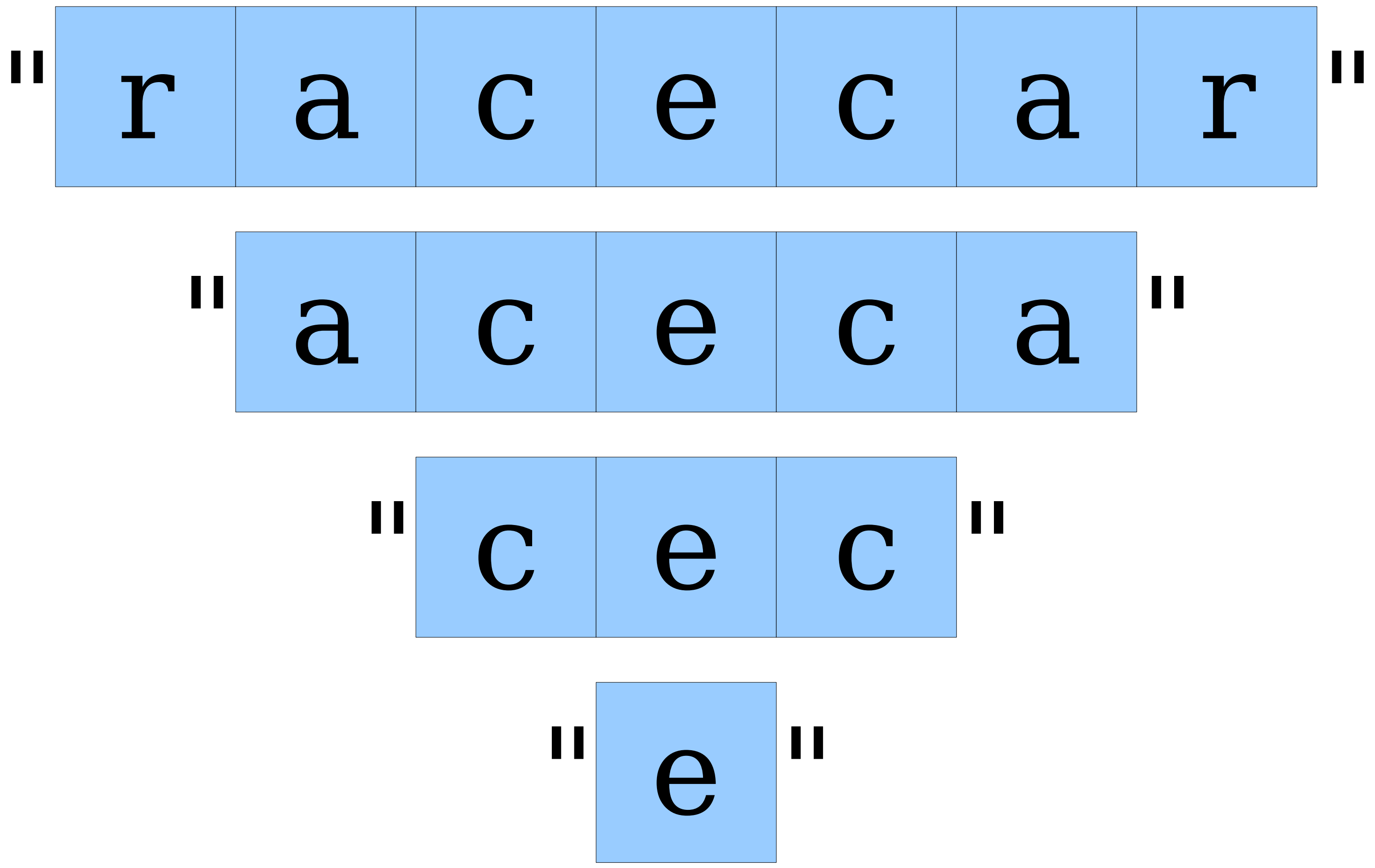
- Base case: *“When is the string so simple that I can immediately tell whether or not it's a palindrome?”*
- Recursive Decomposition: *“How can I simplify the the string?”*

Recursive Palindromes

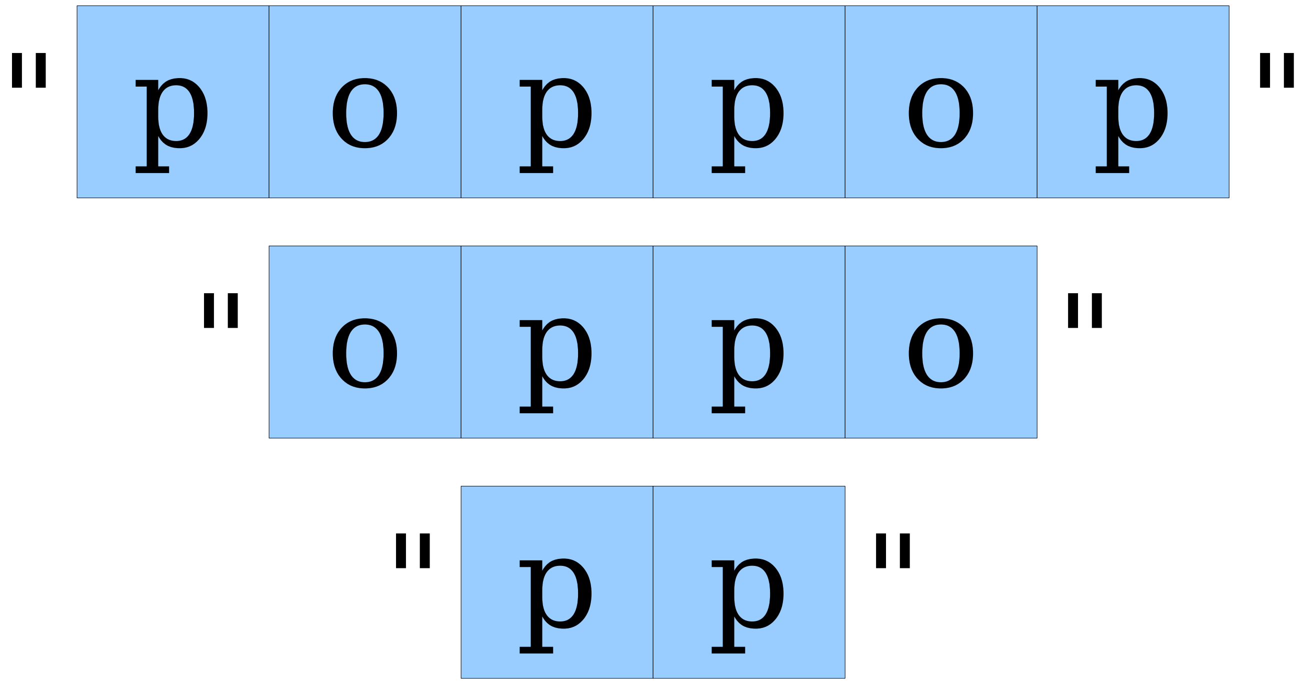
- Base case: *“When is the string so simple that I can immediately tell whether or not it's a palindrome?”*
- Recursive Decomposition: *“How can I simplify the the string?”*

Palindrome: Recursive
palindrome.cpp
(On Computer)

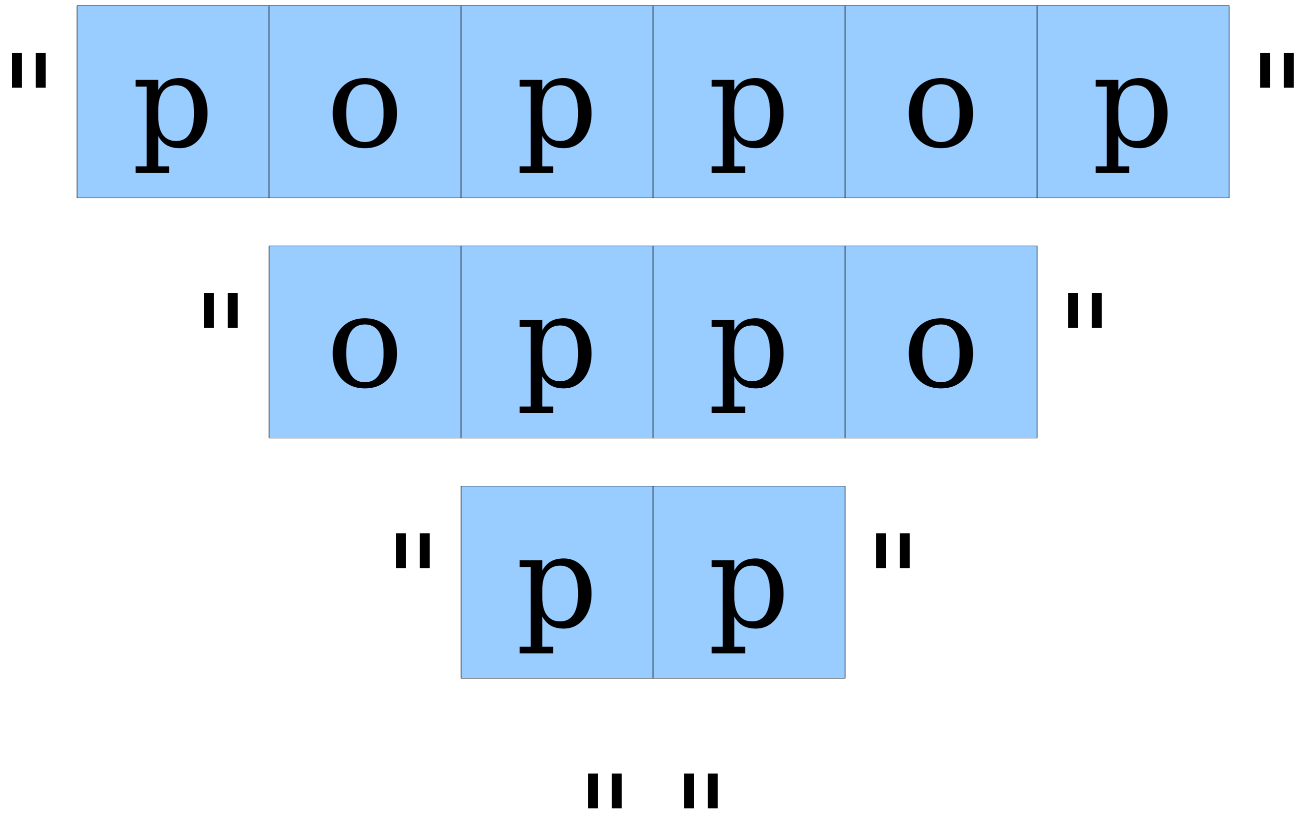
Thinking Recursively



Thinking Recursively



Thinking Recursively



Today

- C++ Strings
- Recursion with Strings
- Reading Files in C++
- Parameter Passing and Common Mistakes

Getting Data from Files

- Now that we have **`string`**s, we can start working with data pulled in from external files.
- File reading in C++ is done using the **`ifstream`** class.
  - Must **`#include`** `<fstream>` to use `ifstream`.

Reading Line by Line

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

  getline(***file***, ***str***)

- 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!**

Reading Files: palindrome.cpp (On Computer)

Reading Formatted Data

- You can read formatted data from a file by using the **stream extraction operator**:

  file >> ***variable***

- Can read any primitive type, plus strings.
- When reading strings, stops at newlines or whitespace.
- Canonical "read formatted data loop:"

```
type val;
while (file >> val) {
    /* … process val … */
}
```

Today

- C++ Strings
- Recursion with Strings
- Reading Files in C++
- Parameter Passing and Common Mistakes

Parameter Passing

- In C++ there are two ways to pass a variable to a function:
  - By **value**: variable passed to function is *copied*

    ```
    void myFunction(int x);
    ```

  - By **reference**: variable passed to function can be modified in the function

    ```
    void myFunction(int &x)
    ```

Parameter Passing

```
int main() {
    int x = 10;
    int y = 20;
    //Here: x = 10, y = 20
    sum(x,y);
    //Here: x = 10, y = 20
    swap(x,y);
    //Here: x = 20, y = 10
    cout << x << " " << y << endl;
    return 0;
}

//Pass by reference
void swap(int &x, int &y) {
    int temp = x;
    x = y;
    y = temp;
}
//Pass by value
void printSum(int x, int y) {
    x += y;
    cout << x << endl;
}
```

Common C++ Mistakes

- If you need to use a function in one of our libraries, remember to **`#include`** the file it is in!

```
#include “simpio.h”
```

- Prototype functions before you use them!

```
bool isPrime(int x);
```

- Variables are not initialized to 0!

```
int x = 0;
```

Next Time

- **`Stack`**
  - A surprisingly useful collection class.
- **`TokenScanner`**
  - A tool for cutting apart strings.
- **The Shunting-Yard Algorithm**
  - How do computers parse expressions?