Containers

Part Two

Outline for Today

- ***`Lexicon`***
  - Storing a collection of words.
- ***`Set`***
  - Storing a group of whatever you'd like.
- ***`Map`***
  - A powerful, fundamental container.

Lexicon

Lexicon

- A **`Lexicon`** is a container that stores a collection of words.
- The `Lexicon` is designed to answer the following question efficiently:

  Given a word, is it contained in the `Lexicon`?

- The `Lexicon` does *not* support access by index. You can't, for example, ask what the 137th English word is.
- However, it *does* support questions of the form "does this word exist?" or "do any words have this as a prefix?"

Tautonyms

- A ***tautonym*** is a word formed by repeating the same string twice.
  - For example: murmur, couscous, papa, etc.
- What English words are tautonyms?

Time-Out for Announcements!

Sections

- Discussion sections start this week!
  - Didn’t sign up for a section? You can sign up for any section that has an open slot by visiting the CS198 website (cs198.stanford.edu).
  - If your section time doesn’t work for you, you can also switch into any section with available space. Visit cs198.stanford.edu to do this.
- ***Reminder:*** Section attendance and participation forms part of your course grade. (Also, if you don’t have a section, none of your work will be graded!)
- ***Reminder:*** We don’t look to Axess enrollments; you need to have a section assigned through our system.

Late Policy

- Everyone has four free “late days” that can be used to extend assignment deadlines.
- Each late day grants an automagic 24-hour extension on an assignment.
- You can use at most two late days per assignment; nothing will be accepted more than 48 hours after the normal deadline.
- Check the syllabus for more information.

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.
- We have a style guide up the course website, as well as a pre-submit checklist to make sure everything is ready to go before you formally submit. Check these out – they’re very useful!

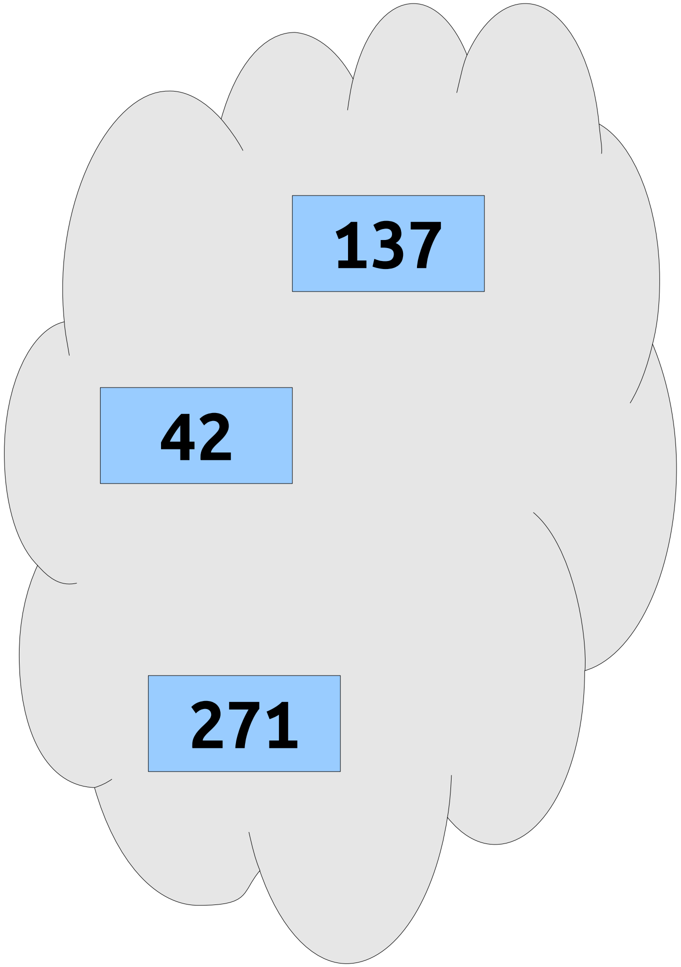
▶

Set

Set

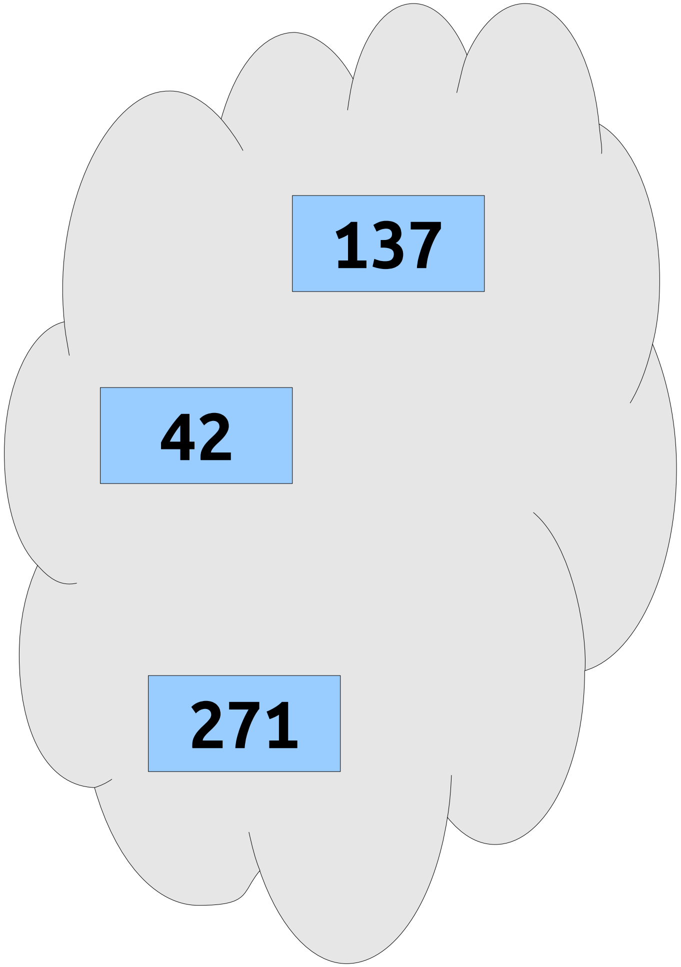
- The **Set** represents an unordered collection of distinct elements.
- Elements can be added and removed. Duplicates aren’t allowed.

```
Set<int> values = {137, 106, 42};

values += 271;
values += 271; // Has no effect

values -= 106;
values -= 103; // Has no effect
```



Set

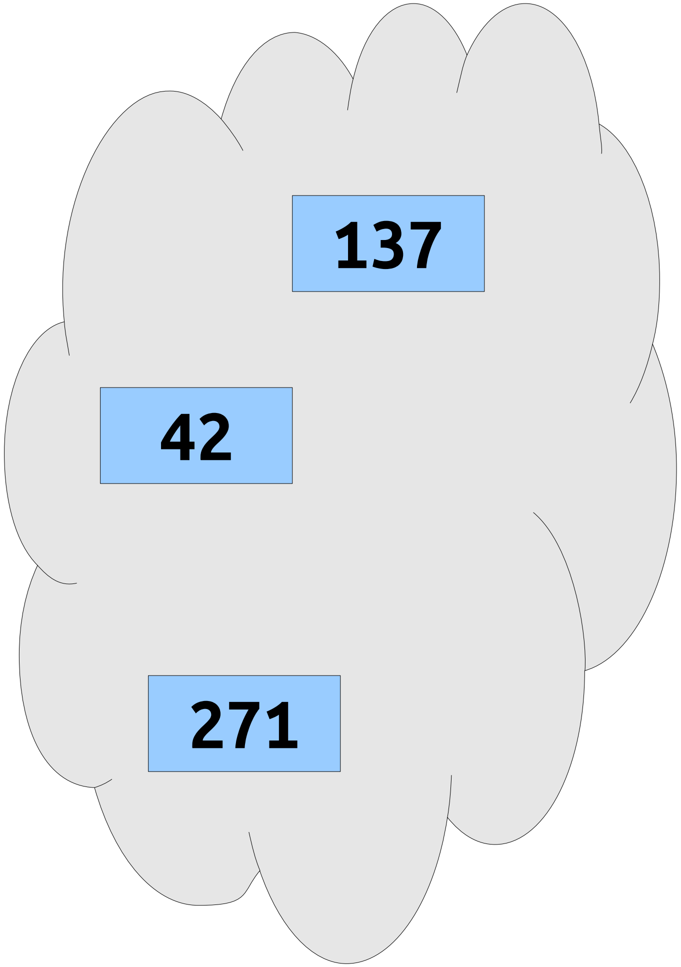
- The `Set` represents an unordered collection of distinct elements.
- Elements can be added and removed. Duplicates aren't allowed.
- You may find it helpful to interpret += as "ensure this item is there" and -= as "ensure this item isn't there."



Set

- Sets make it easy to check if you’ve seen something before.
- You can loop over the contents of a set with a range-based **for** loop.

```
if (values.contains(137)) {
    cout << "<(^_^)>" << endl;
}

for (int value: values) {
    cout << value << endl;
}
```



Operations on Sets

- You can add a value to a `Set` by writing

```
set += value;
```

- You can remove a value from a `Set` by writing

```
set -= value;
```

- You can check if a value exists in a `Set` by writing

```
set.contains(value)
```

- Many more operations are available (union, intersection, difference, subset, etc.). Check the Stanford C++ Library Reference guide for details!

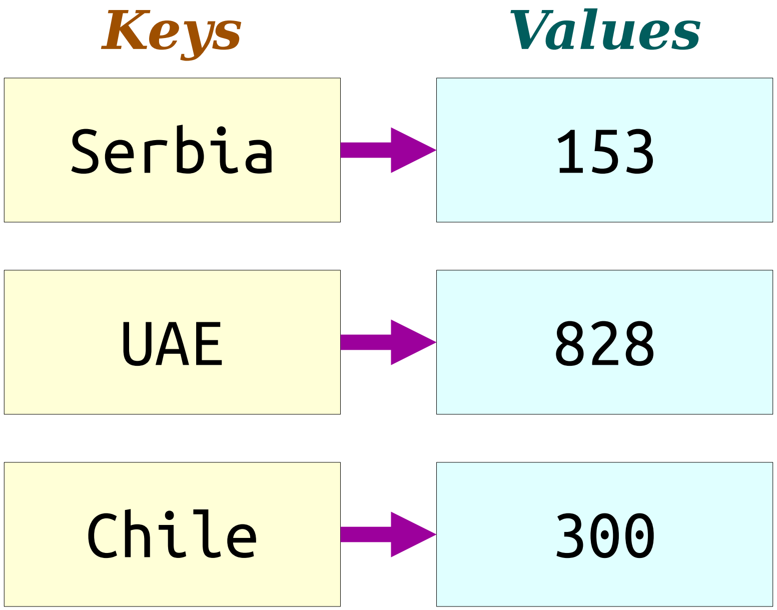
Application: Word Economy

- Some long words are use few distinct letters.
  - “caracara” has length eight, but only uses the letters c, r, and a.
- The ***character efficiency*** of a word is the ratio of its length to the number of different letters it contains.
  - “caracara” has efficiency $^8/_3 \approx 2.67$.
- What is the highest-efficiency English word?

Map

Map

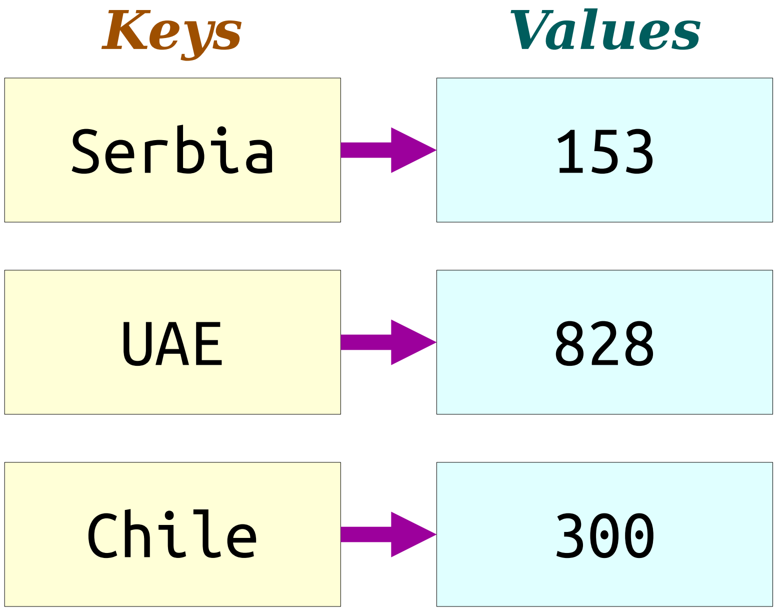
- The **Map** class represents a set of key/value pairs.
  - It's analogous to dict in Python, to Map in Java, and to objects (used as key/value stores) in JavaScript.
- Each key is associated with a value.
- Given a key, we can look up the associated value.



```
Map<string, int> heights;

heights["Serbia"] = 153;
heights["UAE"] = 360;
heights["Chile"] = 300;
heights["UAE"] = 828;

cout << heights["Chile"] << endl;
```

Map

- We can loop over the keys in a map with a range-based for loop.
- We can check whether a key is present in the map.



Keys	*Values*
Serbia	153
UAE	828
Chile	300

```
for (string key: heights) {
    cout << heights[key] << endl;
}

if (heights.containsKey("Mali") {
    cout << "BCEAO" << endl;
}
```

What’d I Say?

- Our program will prompt the user to repeatedly type in text.
- Each time, we’ll report how many previous times the user has typed in that text.
- We’ll use a `Map` to track frequencies!

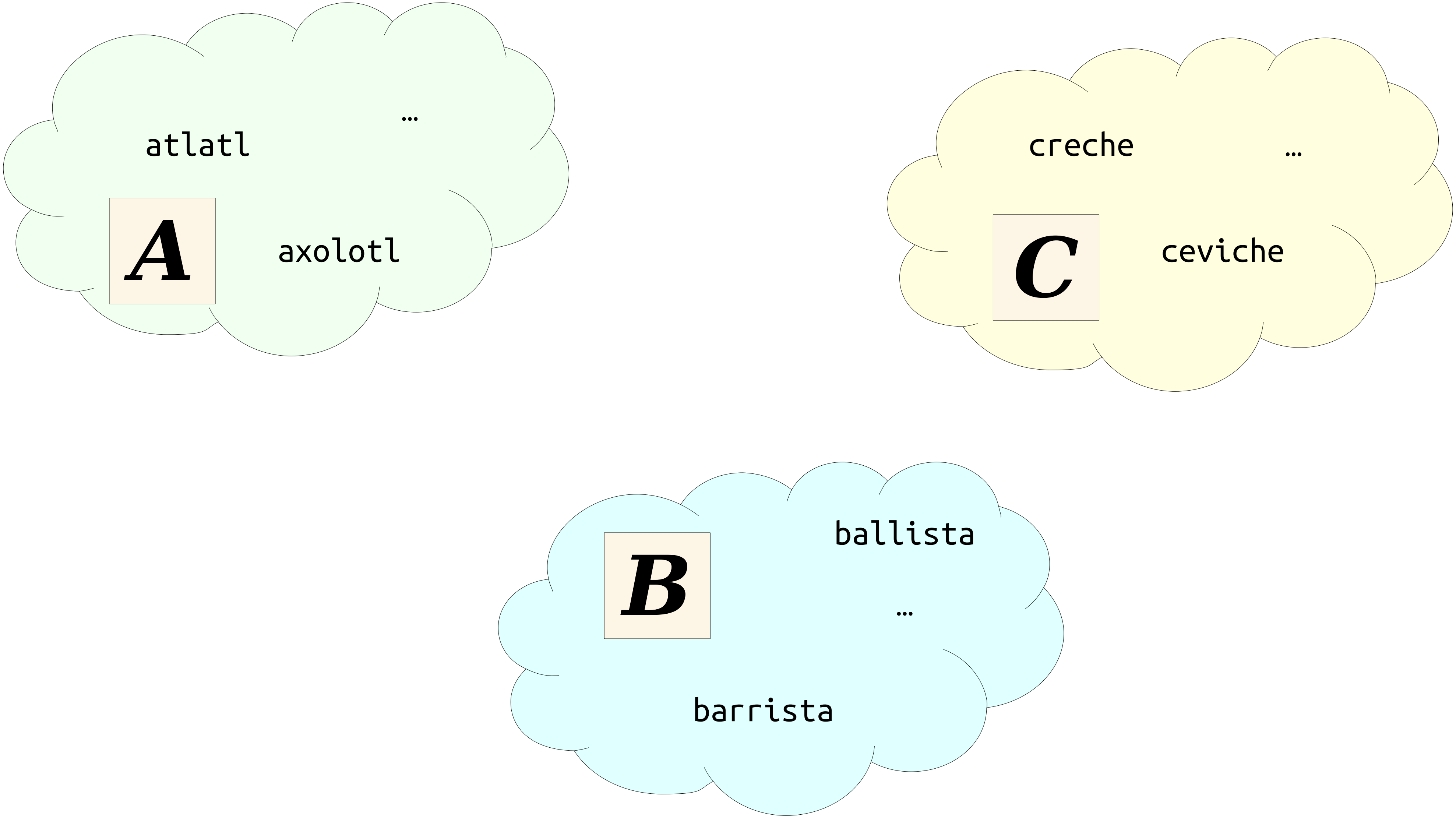
Map Autoinsertion

- If you look up something in a `Map` using square brackets,
  - if the key already exists, its associated value is returned; and
  - if the key doesn't exist, it's added in with a "sensible default" value, and that value is then returned.
- This can take some getting used to, but it's surprisingly convenient.

Type	***Default***
`int`	`0`
`double`	`0.0`
`bool`	`false`
`string`	`""`
Any Container	Empty container of that type
`char`	*`(it's complicated)`*

Grouping by First Letters

Grouping by First Letters



atlatl

…

A

axolotl

creche

…

C

ceviche

B

ballista

…

barrista

Your Action Items

- ***Read Chapter 5.***
  - It's all about container types, and it'll fill in any remaining gaps from this week.
- ***Read the Style Guide***
  - Coding style is important! We want to be clear with our expectations.
- ***Keep Working on Assignment 1.***
  - If you're following our recommended timetable, you'll have finished Debugger Warmups and Fire at this point and will be working on Only Connect.

Next Time

- ***Stacks and Queues***
  - Specialized containers for specialized sequences.
  - Applications to text analysis and music.