Introduction; PA1

ITP 435
Week 1, Lecture 1

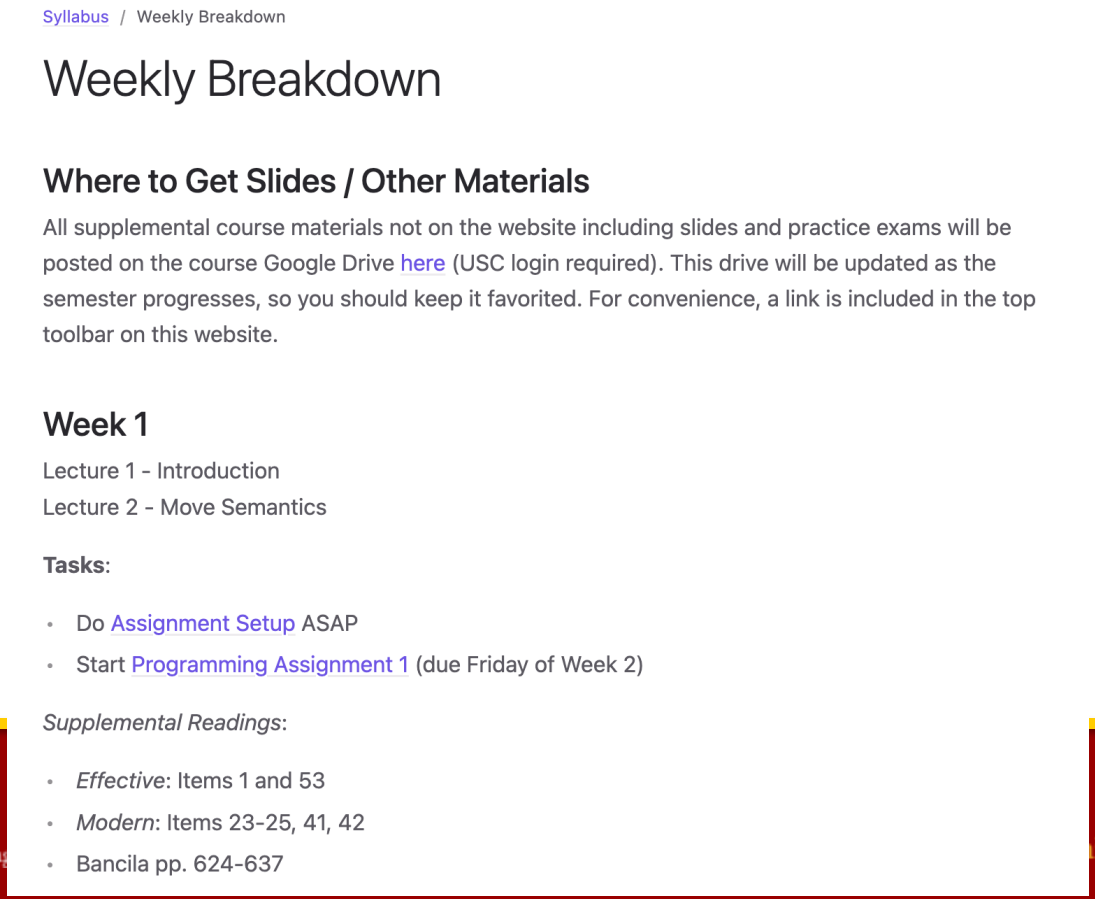
Logistics

- All course materials are on the course website: https://itp435-20243.github.io/
- No discussion or lab section – ask questions during lecture!
- Many class meetings will have short in-person group activities to reinforce lecture content, are graded credit/no credit
- There's a midterm and final exam (cumulative)

Helpful Links

- Syllabus: https://itp435-20243.github.io/Syllabus.html

- Weekly Breakdown (including link to Google Drive w/ slides, etc): https://itp435-20243.github.io/Weekly.html



Syllabus / Weekly Breakdown

Weekly Breakdown

Where to Get Slides / Other Materials

All supplemental course materials not on the website including slides and practice exams will be posted on the course Google Drive here (USC login required). This drive will be updated as the semester progresses, so you should keep it favorited. For convenience, a link is included in the top toolbar on this website.

Week 1

Lecture 1 - Introduction
Lecture 2 - Move Semantics

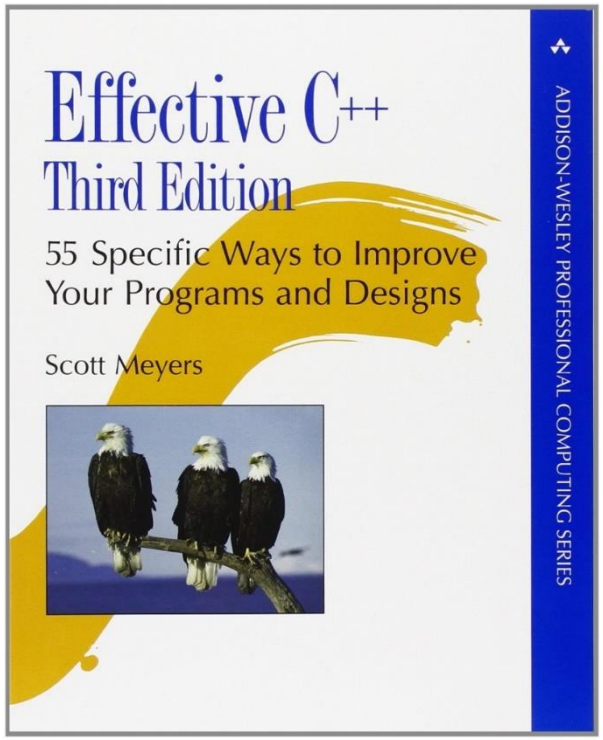
Tasks:

- Do Assignment Setup ASAP
- Start Programming Assignment 1 (due Friday of Week 2)

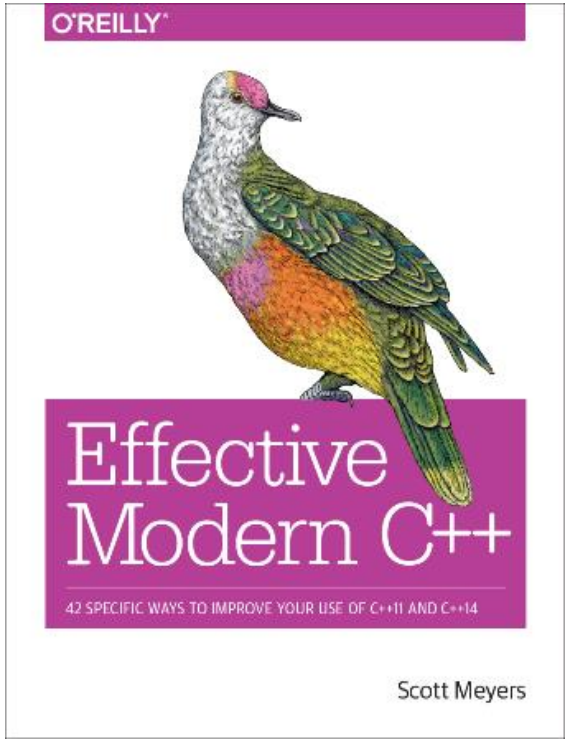
Supplemental Readings:

- *Effective*: Items 1 and 53
- *Modern*: Items 23-25, 41, 42
- Bancila pp. 624-637

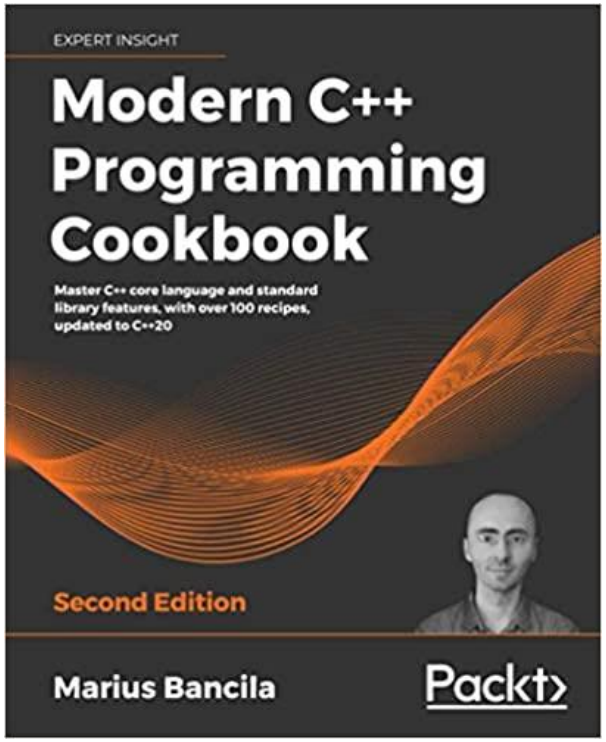
Books



Important



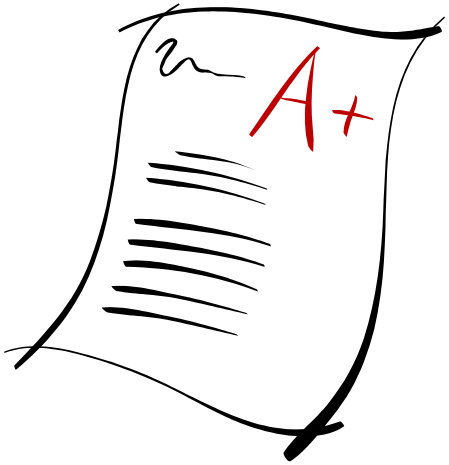
Recommended



Supplemental

You can read these all for free through the USC Library! (Links in the syllabus)

Grading Rubric



Category	Grade %
Programming Assignments (7% Each)	49%
Midterm Exam	20%
Final Exam	20%
In-class activities	11%

Programming Assignments

- 7 assignments, ~2 weeks per
- Instructions on the course site

- Expect each PA to take 8-12 hours – ***don't procrastinate!***

Programming Assignments (Slip Days)

- You get four slip days for the entire semester
- After you've exhausted all your slip days, you lose -25 points per day late after
- ***No further extensions*** beyond the 4 slip days everyone gets

Exams

- Midterm/Final Exams will be in-person, timed, written exams
- Final exam is cumulative
- Check the syllabus for dates!

In-class Activity Grading

- Most class meetings will have small ~10 minute group activities
- They're graded credit/no credit (one submission per group)
- We understand that sometimes things come up and you must miss class. There are a total of 25* of these and you can miss up to 4 without any penalty on your grade

* 25 is the goal, but we may end up with slightly less

Development Platforms

- We recommend developing with an IDE, though you can use the command line if you want to:

Platform	IDE?	Command Line?
Windows	Visual Studio **2022**	Yes, in WSL
Mac	Visual Studio Code	Yes
Linux	Visual Studio Code	Yes

- While other IDEs can work, we only officially support Visual Studio 2022 on Windows and Visual Studio Code on Mac

Academic Integrity – What not to do

- ***Do not*** share your code files or part of your code files with your classmates (current or future students).
- ***Do not*** post your code on a publicly-accessible website (GitHub, course hero, etc).
- ***Do not*** step through anyone else's code. If you need help debugging ask an instructor or TA or Piazza.
- ***Do not*** have your tutor write your code or take the exam for you (true story).
- ***Do not*** have ChatGPT, GitHub Copilot or another AI write your code
- If you're not sure if something is allowed: ask an instructor
- Instructors/LAs are always happy to help!

AI Use Policy

- We do not allow you to use ChatGPT, GitHub Copilot, or other AIs to write your code in this class.

- Why not?
  - These duplicate code without attribution or accounting for license, which is a legal problem if you work in industry
  - When you interview with companies, they will expect you to write code and solve problems without the use of AI
  - In their current state, they are woefully unreliable for more difficult tasks (like later assignments or things you're expected to do in industry)
  - You will fail the exams

Why C++?

Which industries still use lots of C++?

- Some teams at big tech companies like Google/Apple/Microsoft
- Investment banks, financial trading, hedge funds
- High performance computing
- Video games
- Defense contractors

Is C++ the best language for everything?

- No
  - Python is great for a quick script where you don't care too much about performance (there's a built-in library for everything)
  - Programming a website in C++ is possible, but totally not worth it
  - For an operating system, you probably want to use C

That Being Said

- This class is about ***writing code in an environment where the performance matters***. If we care about performance, why not maximize it?

Goals of this Class

1. Quickly refresh your memory on the C++ you may have forgotten.

2. Refine your ability to design and write high-quality C++ code.

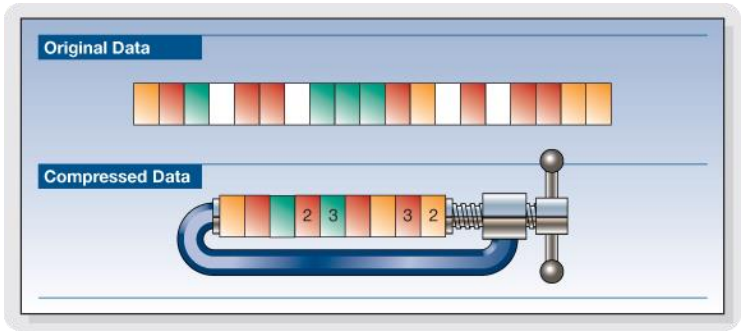
3. Learn new ways to use things you already know about (eg., templates).

4. Learn the ways the language is has evolved with the newer C++ language updates.

PA1 Topic: LZW Compression

What's data compression?

- Take data and try to make it smaller



- You use it all the time…
  - ZIP
  - PNG
  - MP4
  - MOV

LZW

- Lempel-Ziv-Welch
- Invented in 1984
- Is a “dictionary” compression algorithm
- Used in GIF
- Not particularly great compared to current state of the art, but not nearly as complex as the state of the art

LZW Dictionary

- Maps a string to a numeric code (or the opposite for decompress)
- Initialize it by adding each string of length 1 as codes 0-255

This looks like multiple values, but this actually is just one char, the byte with value 1 (which doesn't have a human-readable representation)

String	Code
"\0"	0
"\x01"	1
…	…
"a"	97
"b"	98
"c"	99
…	…
"\xFF"	255

LZW Compression Pseudocode

1. Initialize codes 0-255 (***done***)
2. Initialize string **`s`** to the first element in the **`input`**
3. From index 1 to the end of the **`input`**:
   a. Get character **`c`** from the index
   b. if **`s+c`** is in the dictionary
      `1. s = s+c`
   c. else
      1. Add to **`output`** the code for **`s`**
      2. Insert **`s+c`** into the dictionary
      `3. s = c`
4. Add to `output` the code for **`s`**

LZW Compression Pseudocode

Input

Index	0	1	2	3	4	5	6	7
Value	a	b	a	b	a	a	b	b

s	
c	

String	Code
"\0"	0
"\x01"	1
…	…
"a"	97
"b"	98
"c"	99
…	…
"\xFF"	255

2. Initialize string **s** to the first element in the **input**
3. From index 1 to the end of the **input**:
   a. Get character **c** from the index
   b. if **s+c** is in the dictionary
      1. s = s+c
   c. else
      1. Add to **output** the code for **s**
      2. Insert **s+c** into the dictionary
      3. s = c
4. Add to output the code for **s**

Output

Index
Value

LZW Compression Pseudocode

Input

Index	0	1	2	3	4	5	6	7
Value	a	b	a	b	a	a	b	b

s	"a"
c	

String	Code
"\0"	0
"\x01"	1
…	…
"a"	97
"b"	98
"c"	99
…	…
"\xFF"	255

2. Initialize string **s** to the first element in the **input**
3. From index 1 to the end of the **input**:
   a. Get character **c** from the index
   b. if **s+c** is in the dictionary
      1. s = s+c
   c. else
      1. Add to **output** the code for **s**
      2. Insert **s+c** into the dictionary
      3. s = c
4. Add to output the code for **s**

Output

Index
Value

LZW Compression Pseudocode

Input

Index	0	1	2	3	4	5	6	7
Value	a	b	a	b	a	a	b	b

s	"a"
c	

String	Code
"\0"	0
"\x01"	1
…	…
"a"	97
"b"	98
"c"	99
…	…
"\xFF"	255

2. Initialize string **s** to the first element in the **input**
3. From index 1 to the end of the **input**:
   a. Get character **c** from the index
   b. if **s+c** is in the dictionary
      1. s = s+c
   c. else
      1. Add to **output** the code for **s**
      2. Insert **s+c** into the dictionary
      3. s = c
4. Add to output the code for **s**

Output

Index
Value

LZW Compression Pseudocode

Input

Index	0	1	2	3	4	5	6	7
Value	a	b	a	b	a	a	b	b

s	"a"
c	b

String	Code
"\0"	0
"\x01"	1
…	…
"a"	97
"b"	98
"c"	99
…	…
"\xFF"	255

2. Initialize string **s** to the first element in the **input**
3. From index 1 to the end of the **input**:
   a. Get character **c** from the index
   b. if **s+c** is in the dictionary
      1. s = s+c
   c. else
      1. Add to **output** the code for **s**
      2. Insert **s+c** into the dictionary
      3. s = c
4. Add to output the code for **s**

Output

Index
Value

LZW Compression Pseudocode

Input

Index	0	1	2	3	4	5	6	7
Value	a	b	a	b	a	a	b	b

s	"a"
c	b

String	Code
"\0"	0
"\x01"	1
…	…
"a"	97
"b"	98
"c"	99
…	…
"\xFF"	255

2. Initialize string **s** to the first element in the **input**
3. From index 1 to the end of the **input**:
   a. Get character **c** from the index
   b. if **s+c** is in the dictionary
      1. s = s+c
   c. else
      1. Add to **output** the code for **s**
      2. Insert **s+c** into the dictionary
      3. s = c
4. Add to output the code for **s**

Output

Index
Value

LZW Compression Pseudocode

Input

Index	0	1	2	3	4	5	6	7
Value	a	b	a	b	a	a	b	b

s	"a"
c	b

String	Code
"\0"	0
"\x01"	1
…	…
"a"	97
"b"	98
"c"	99
…	…
"\xFF"	255

2. Initialize string **s** to the first element in the **input**
3. From index 1 to the end of the **input**:
   a. Get character **c** from the index
   b. if **s+c** is in the dictionary
      1. s = s+c
   c. else
      1. Add to **output** the code for **s**
      2. Insert **s+c** into the dictionary
      3. s = c
4. Add to output the code for **s**

Output

Index	0
Value	97

LZW Compression Pseudocode

Input

Index	0	1	2	3	4	5	6	7
Value	a	b	a	b	a	a	b	b

s	"a"
c	b

String	Code
"\0"	0
"\x01"	1
…	…
"a"	97
"b"	98
"c"	99
…	…
"\xFF"	255
"ab"	256

2. Initialize string **s** to the first element in the **input**
3. From index 1 to the end of the **input**:
   a. Get character **c** from the index
   b. if **s+c** is in the dictionary
      1. s = s+c
   c. else
      1. Add to **output** the code for **s**
      2. Insert **s+c** into the dictionary
      3. s = c
4. Add to output the code for **s**

Output

Index	0
Value	97

LZW Compression Pseudocode

Input

Index	0	1	2	3	4	5	6	7
Value	a	b	a	b	a	a	b	b

s	"b"
c	b

String	Code
"\0"	0
"\x01"	1
…	…
"a"	97
"b"	98
"c"	99
…	…
"\xFF"	255
"ab"	256

2. Initialize string **s** to the first element in the **input**
3. From index 1 to the end of the **input**:
   a. Get character **c** from the index
   b. if **s+c** is in the dictionary
      1. **s = s+c**
   c. else
      1. Add to **output** the code for **s**
      2. Insert **s+c** into the dictionary
      3. **s = c**
4. Add to output the code for **s**

Output

Index	0
Value	97

LZW Compression Pseudocode

Input

Index	0	1	2	3	4	5	6	7
Value	a	b	a	b	a	a	b	b

s	"b"
c	a

String	Code
"\0"	0
"\x01"	1
…	…
"a"	97
"b"	98
"c"	99
…	…
"\xFF"	255
"ab"	256

2. Initialize string **s** to the first element in the **input**
3. From index 1 to the end of the **input**:
   a. Get character **c** from the index
   b. if **s+c** is in the dictionary
      1. s = s+c
   c. else
      1. Add to **output** the code for **s**
      2. Insert **s+c** into the dictionary
      3. s = c
4. Add to output the code for **s**

Output

Index	0
Value	97

LZW Compression Pseudocode

Input

Index	0	1	2	3	4	5	6	7
Value	a	b	a	b	a	a	b	b

s	"b"
c	a

String	Code
"\0"	0
"\x01"	1
…	…
"a"	97
"b"	98
"c"	99
…	…
"\xFF"	255
"ab"	256

2. Initialize string **s** to the first element in the **input**
3. From index 1 to the end of the **input**:
   a. Get character **c** from the index
   b. if **s+c** is in the dictionary
      1. s = s+c
   c. else
      1. Add to **output** the code for **s**
      2. Insert **s+c** into the dictionary
      3. s = c
4. Add to output the code for **s**

Output

Index	0
Value	97

LZW Compression Pseudocode

Input

Index	0	1	2	3	4	5	6	7
Value	a	b	a	b	a	a	b	b

s	"b"
c	a

String	Code
"\0"	0
"\x01"	1
…	…
"a"	97
"b"	98
"c"	99
…	…
"\xFF"	255
"ab"	256

2. Initialize string **s** to the first element in the **input**
3. From index 1 to the end of the **input**:
   a. Get character **c** from the index
   b. if **s+c** is in the dictionary
      1. s = s+c
   c. else
      1. Add to **output** the code for **s**
      2. Insert **s+c** into the dictionary
      3. s = c
4. Add to output the code for **s**

Output

Index	0	1
Value	97	98

LZW Compression Pseudocode

Input

Index	0	1	2	3	4	5	6	7
Value	a	b	a	b	a	a	b	b

s	"b"
c	a

String	Code
"\0"	0
"\x01"	1
…	…
"a"	97
"b"	98
"c"	99
…	…
"\xFF"	255
"ab"	256
"ba"	257

2. Initialize string **s** to the first element in the **input**
3. From index 1 to the end of the **input**:
   a. Get character **c** from the index
   b. if **s+c** is in the dictionary
      1. s = s+c
   c. else
      1. Add to **output** the code for **s**
      2. Insert **s+c** into the dictionary
      3. s = c
4. Add to output the code for **s**

Output

Index	0	1
Value	97	98

LZW Compression Pseudocode

Input

Index	0	1	2	3	4	5	6	7
Value	a	b	a	b	a	a	b	b

s	"a"
c	a

String	Code
"\0"	0
"\x01"	1
…	…
"a"	97
"b"	98
"c"	99
…	…
"\xFF"	255
"ab"	256
"ba"	257

2. Initialize string **s** to the first element in the **input**
3. From index 1 to the end of the **input**:
   a. Get character **c** from the index
   b. if **s+c** is in the dictionary
      1. s = s+c
   c. else
      1. Add to **output** the code for **s**
      2. Insert **s+c** into the dictionary
      3. s = c
4. Add to output the code for **s**

Output

Index	0	1
Value	97	98

LZW Compression Pseudocode

Input

Index	0	1	2	3	4	5	6	7
Value	a	b	a	b	a	a	b	b

s	"a"
c	b

String	Code
"\0"	0
"\x01"	1
…	…
"a"	97
"b"	98
"c"	99
…	…
"\xFF"	255
"ab"	256
"ba"	257

2. Initialize string **s** to the first element in the **input**
3. From index 1 to the end of the **input**:
   a. Get character **c** from the index
   b. if **s+c** is in the dictionary
      1. s = s+c
   c. else
      1. Add to **output** the code for **s**
      2. Insert **s+c** into the dictionary
      3. s = c
4. Add to output the code for **s**

Output

Index	0	1
Value	97	98

LZW Compression Pseudocode

Input

Index	0	1	2	3	4	5	6	7
Value	a	b	a	b	a	a	b	b

s	"a"
c	b

String	Code
"\0"	0
"\x01"	1
…	…
"a"	97
"b"	98
"c"	99
…	…
"\xFF"	255
"ab"	256
"ba"	257

2. Initialize string **s** to the first element in the **input**
3. From index 1 to the end of the **input**:
   a. Get character **c** from the index
   b. if **s+c** is in the dictionary
      1. **s = s+c**
   c. else
      1. Add to **output** the code for **s**
      2. Insert **s+c** into the dictionary
      3. **s = c**
4. Add to output the code for **s**

Output

Index	0	1
Value	97	98

LZW Compression Pseudocode

Input

Index	0	1	2	3	4	5	6	7
Value	a	b	a	b	a	a	b	b

s	"ab"
c	b

String	Code
"\0"	0
"\x01"	1
…	…
"a"	97
"b"	98
"c"	99
…	…
"\xFF"	255
"ab"	256
"ba"	257

2. Initialize string **s** to the first element in the **input**
3. From index 1 to the end of the **input**:
   a. Get character **c** from the index
   b. if **s+c** is in the dictionary
      1. s = s+c
   c. else
      1. Add to **output** the code for **s**
      2. Insert **s+c** into the dictionary
      3. s = c
4. Add to output the code for **s**

Output

Index	0	1
Value	97	98

LZW Compression Pseudocode

Input

Index	0	1	2	3	4	5	6	7
Value	a	b	a	b	a	a	b	b

s	"ab"
c	a

String	Code
"\0"	0
"\x01"	1
…	…
"a"	97
"b"	98
"c"	99
…	…
"\xFF"	255
"ab"	256
"ba"	257

2. Initialize string **s** to the first element in the **input**
3. From index 1 to the end of the **input**:
   a. Get character **c** from the index
   b. if **s+c** is in the dictionary
      1. s = s+c
   c. else
      1. Add to **output** the code for **s**
      2. Insert **s+c** into the dictionary
      3. s = c
4. Add to output the code for **s**

Output

Index	0	1
Value	97	98

LZW Compression Pseudocode

Input

Index	0	1	2	3	4	5	6	7
Value	a	b	a	b	a	a	b	b

s	"ab"
c	a

String	Code
"\0"	0
"\x01"	1
…	…
"a"	97
"b"	98
"c"	99
…	…
"\xFF"	255
"ab"	256
"ba"	257

2. Initialize string **s** to the first element in the **input**
3. From index 1 to the end of the **input**:
   a. Get character **c** from the index
   b. if **s+c** is in the dictionary
      1. **s = s+c**
   c. else
      1. Add to **output** the code for **s**
      2. Insert **s+c** into the dictionary
      3. **s = c**
4. Add to output the code for **s**

Output

Index	0	1
Value	97	98

LZW Compression Pseudocode

Input

Index	0	1	2	3	4	5	6	7
Value	a	b	a	b	a	a	b	b

s	"ab"
c	a

String	Code
"\0"	0
"\x01"	1
…	…
"a"	97
"b"	98
"c"	99
…	…
"\xFF"	255
"ab"	256
"ba"	257

2. Initialize string **s** to the first element in the **input**
3. From index 1 to the end of the **input**:
   a. Get character **c** from the index
   b. if **s+c** is in the dictionary
      1. s = s+c
   c. else
      1. Add to **output** the code for **s**
      2. Insert **s+c** into the dictionary
      3. s = c
4. Add to output the code for **s**

Output

Index	0	1	2
Value	97	98	256

LZW Compression Pseudocode

Input

Index	0	1	2	3	4	5	6	7
Value	a	b	a	b	a	a	b	b

s	"ab"
c	a

String	Code
"\0"	0
"\x01"	1
…	…
"a"	97
"b"	98
"c"	99
…	…
"\xFF"	255
"ab"	256
"ba"	257
"aba"	258

2. Initialize string **s** to the first element in the **input**
3. From index 1 to the end of the **input**:
   a. Get character **c** from the index
   b. if **s+c** is in the dictionary
      1. s = s+c
   c. else
      1. Add to **output** the code for **s**
      2. Insert **s+c** into the dictionary
      3. s = c
4. Add to output the code for **s**

Output

Index	0	1	2
Value	97	98	256

LZW Compression Pseudocode

Input

Index	0	1	2	3	4	5	6	7
Value	a	b	a	b	a	a	b	b

s	"a"
c	a

String	Code
"\0"	0
"\x01"	1
…	…
"a"	97
"b"	98
"c"	99
…	…
"\xFF"	255
"ab"	256
"ba"	257
"aba"	258

2. Initialize string **s** to the first element in the **input**
3. From index 1 to the end of the **input**:
   a. Get character **c** from the index
   b. if **s+c** is in the dictionary
      1. s = s+c
   c. else
      1. Add to **output** the code for **s**
      2. Insert **s+c** into the dictionary
      3. s = c
4. Add to output the code for **s**

Output

Index	0	1	2
Value	97	98	256

How to do the group activities?

1. Everyone should already be in the Gradescope course (https://www.gradescope.com/courses/822142). If you aren't in the course already, the entry code is **7EE3XJ**
2. Pick up to 2 other people sitting next to you
3. One person in your group should select the current group activity in Gradescope (*In-class Week 1 Lecture 1*)
4. When that person submits, add the other group members to the submission
5. After a few minutes we'll discuss in class

Remember, they're graded credit/no credit. As long as it's clear your group made a reasonable effort (and didn't just put garbage), you'll get credit.

LZW Compression Pseudocode

Input

Index	0	1	2	3	4	5	6	7
Value	a	b	a	b	a	a	b	b

s	"a"
c	a

String	Code
"\0"	0
"\x01"	1
…	…
"a"	97
"b"	98
"c"	99
…	…
"\xFF"	255
"ab"	256
"ba"	257
"aba"	258

2. Initialize string **s** to the first element in the **input**
3. From index 1 to the end of the **input**:
   a. Get character **c** from the index
   b. if **s+c** is in the dictionary
      1. s = s+c
   c. else
      1. Add to **output** the code for **s**
      2. Insert **s+c** into the dictionary
      3. s = c
4. Add to output the code for **s**

Output

Index	0	1	2
Value	97	98	256

LZW Compression Pseudocode

Input

Index	0	1	2	3	4	5	6	7
Value	a	b	a	b	a	a	b	b

s	"a"
c	a

String	Code
"\0"	0
"\x01"	1
…	…
"a"	97
"b"	98
"c"	99
…	…
"\xFF"	255
"ab"	256
"ba"	257
"aba"	258

2. Initialize string **s** to the first element in the **input**
3. From index 1 to the end of the **input**:
   a. Get character **c** from the index
   b. if **s+c** is in the dictionary
      1. s = s+c
   c. else
      1. Add to **output** the code for **s**
      2. Insert **s+c** into the dictionary
      3. s = c
4. Add to output the code for **s**

Output

Index	0	1	2
Value	97	98	256

LZW Compression Pseudocode

Input

Index	0	1	2	3	4	5	6	7
Value	a	b	a	b	a	a	b	b

s	"a"
c	a

String	Code
"\0"	0
"\x01"	1
…	…
"a"	97
"b"	98
"c"	99
…	…
"\xFF"	255
"ab"	256
"ba"	257
"aba"	258

2. Initialize string **s** to the first element in the **input**
3. From index 1 to the end of the **input**:
   a. Get character **c** from the index
   b. if **s+c** is in the dictionary
      1. s = s+c
   c. else
      1. Add to **output** the code for **s**
      2. Insert **s+c** into the dictionary
      3. s = c
4. Add to output the code for **s**

Output

Index	0	1	2	3
Value	97	98	256	97

LZW Compression Pseudocode

Input

Index	0	1	2	3	4	5	6	7
Value	a	b	a	b	a	a	b	b

s	"a"
c	a

String	Code
"\0"	0
"\x01"	1
…	…
"a"	97
"b"	98
"c"	99
…	…
"\xFF"	255
"ab"	256
"ba"	257
"aba"	258
"aa"	259

2. Initialize string **s** to the first element in the **input**
3. From index 1 to the end of the **input**:
   a. Get character **c** from the index
   b. if **s+c** is in the dictionary
      1. s = s+c
   c. else
      1. Add to **output** the code for **s**
      2. Insert **s+c** into the dictionary
      3. s = c
4. Add to output the code for **s**

Output

Index	0	1	2	3
Value	97	98	256	97

LZW Compression Pseudocode

Input

Index	0	1	2	3	4	5	6	7
Value	a	b	a	b	a	a	b	b

s	"a"
c	a

String	Code
"\0"	0
"\x01"	1
…	…
"a"	97
"b"	98
"c"	99
…	…
"\xFF"	255
"ab"	256
"ba"	257
"aba"	258
"aa"	259

2. Initialize string **s** to the first element in the **input**
3. From index 1 to the end of the **input**:
   a. Get character **c** from the index
   b. if **s+c** is in the dictionary
      1. s = s+c
   c. else
      1. Add to **output** the code for **s**
      2. Insert **s+c** into the dictionary
      3. s = c
4. Add to output the code for **s**

Output

Index	0	1	2	3
Value	97	98	256	97

Implementation Notes

- Function declaration:

```
template <typename CompressDictionary>
void Compress(const std::vector<char>& input,
              std::vector<int16_t>& output)
```

Templated – this is so we can provide different dictionary implementations that conform to an interface

Signed 16-bit integer

CompressDictionary

- Guaranteed to have the following functions:

```
// Returns true if the dictionary contains the requested string
bool Contains(const std::string& str) const;

// Adds the string the dictionary if the dictionary is not full
// Returns true if successful, false otherwise
bool TryAdd(const std::string& str);

// Returns the numeric code for the string,
// or -1 if the string isn't in the dictionary
int16_t GetCode(const std::string& str) const;
```

Using CompressDictionary

- In Compress, just declare it as a local variable:

```
CompressDictionary dictionary;
```

- Then you can later call functions on it, like:

```
if (dictionary.Contains(str))
```

LZW Decompression

- It's more complicated, but you can implement from the pseudocode in the instructions

- Some notes on the implementation:
  - You'll have to just create a local unordered_map for the dictionary (Decompress does not take in a template argument)
  - You'll have to initialize codes 0-255 yourself

unordered_map Tips

- In C++20 (which we're using), there's a `.contains` function that tells you if a key is in the map

- The easiest way to add to an unordered_map is with `.emplace(key, value)`

- You can get a value with the `[]` operator – ***BUT*** – keep in mind that `[]` will insert a default value if the element is not in the map

- Because of the potential for default-insertion, you can't use `[]` in a const function (relevant for part 3), you have to use `.find(key)`

PA1 Grading Breakdown

Criteria	Points
Graded Tests	
Part 1	
*** LZW Compression (8 tests, 3 points per)	24
*** LZW Decompression (8 tests, 2 points per)	16
Part 2	
*** File Compression (3 tests, 5 points per)	15
Part 3	
*** LZW Compression w/ Custom Dictionary (8 tests, 2 points per)	16
*** File Compression w/ Custom Dictionary (3 tests, 3 points per)	9
*** Timing Comparison	10
Code Quality (fix all build/clang-tidy warnings to avoid deductions)	10
Total	**100**

Our Development Environment

GitHub

- Git is a popular source control system
- GitHub is a popular website that hosts Git repositories
- We use GitHub classroom to create repositories (just follow the link given in the PA)
- ***We can't grade your code if it's not pushed to your GitHub repo***

Continuous Integration – GitHub Actions

1. You write code

2. You push to GitHub

GitHub Actions

3. GitHub Actions automatically starts a build

4. You can see the success/failure on your repo on GitHub

What do we run on GitHub actions?

1. Compiles your code – reports error if fails
2. Runs clang-tidy to check for static analysis warnings
3. Runs your (student) unit tests – reports error if a test fails
4. Runs our (graded) unit tests and shows you a grade – reports error if a test fails

Future labs may add other features!

GitHub Actions Grade Report

- I wrote a custom test case reporter that gives a nice summary:

Grade Report

Result	Test	Points	Earned	Details
Graded RLE Compress				
✅ PASS	Basic positive runs	2	2	
✅ PASS	Basic negative runs	2	2	
✅ PASS	Mix of positive and negative runs	2	2	
✅ PASS	Long positive run	2	2	
✅ PASS	Long negative run	2	2	
❌ FAIL	Long positive followed by a negative run of size 2	2	0	▶ Failure Info
✅ PASS	Long negative followed by postive run of size 2	2	2	
✅ PASS	Run with 20 zeroes	2	2	
✅ PASS	2 positive, 254 negative, 2 positive	2	2	
	Subtotal	18	16	
Graded RLE Decompress				
✅ PASS	Decompress Basic positive run	2	2	

GitHub Actions and Grading

- ***If your code does not compile on GitHub Actions, you get a 0***

- Even if you pass all graded tests, you are expected to look at your actions results to check for clang-tidy and/or compiler warnings

- Fix your warnings

- ***You will lose lose code quality points for warnings***

- Commit early and often so that you know what GitHub actions thinks about your code – don't wait until the last minute

Building C++ on multiple platforms

- You develop on Windows or Mac (probably)
- The GitHub actions VMs run Linux

- CMake is a cross-platform C++ build system that can generate Visual Studio projects, Xcode projects, Unix-style make files, etc.
- (For PA1 you won't have to tinker with the CMake build files, but you will on later assignments)

Modern C++ Odds and Ends

Don’t do using namespace std

- What happens if you do this?

```
#include <map>
using namespace std;

// This function maps A to B
int map(int A, int B)
{
   // ...
}

int main(int argc, const char* argv[])
{
   // I'm making an STL map!
   map<std::string, int> myMap;
   // ...
   return 0;
}
```

But I don't like typing std::

- You can do `using` for a specific name you want to use, like:

```
#include <iostream>
using std::cout;
using std::endl;

// Later in code...
cout << "cout syntax is annoying" << endl;
```

- Best practice is only do this in a .cpp file, not in a .h file

Null Pointers

- Previously, null pointers were represented with 0 (or NULL, which is just a define as 0).
- This is not strongly typed...

```
void f1(int);
void f1(char *);

// 0 could mean int or it could mean a NULL pointer...
// Compiler always choose the “int” version.
f1(0);
```

- In C++11, there is a now a `nullptr` keyword, which is strongly typed

```
// nullptr is strongly-typed, so it calls the char* version
f1(nullptr);
```

C-style Casts

- In C, to cast from one type to another you typically do:

```
int i;
float f = (float)i;
```

- Problem 1: Searching for all casts in a file is impossible.

- Problem 2: C-style casts allow you to do crazy (unsafe) stuff like:

```
int random = 0x123456;
char* garbage = (char*)random;
(*garbage) = 'a'; // probably will crash
```

C++ Style Casts

- static_cast – Allows you to do implicit conversions (eg. int -> float), as well as within a class hierarchy (without any runtime checks)

```
int i;
float f = static_cast<float>(i);
```

- reinterpret_cast – Allows you to do unsafe pointer casts

```
int random_address = 0x123456;
char* garbage = reinterpret_cast<char*>(random_address);
```

- Having these two different categories allow us to easily see what might be a more “dangerous” cast

- There are also const_cast and dynamic_cast, but these are infrequently used

Override Keyword

- Suppose there is a Class A:

```
class A
{
public:
   virtual void TakeDamage(int amount);
};
```

- Class B inherits from A, and overrides TakeDamage:

```
class B : public A
{
public:
   void TakeDamage();
};
```

- What's wrong with this picture (yes, it compiles!)?

Override Keyword, Cont'd

- The keyword `override` after the function name/parameters says "I guarantee this overrides a function from a parent class"

- So if we wrote the code as:

```
class B : public A
{
public:
   void TakeDamage() override;
};
```

- Error C3668: 'B::TakeDamage' : method with override specifier 'override' did not override any base class methods

Auto keyword

- Auto tells the compiler to deduce the type:

```
// long and annoying
std::vector<int>::iterator myIter = myVect.begin();

// short and sweet
auto myIter = myVect.begin();
```

- Added in C++11, supported in every current compiler
- (Some developers say you should always use auto for every variable, but I don't agree with that)

Range-Based for Loop

- Like the foreach loop in other languages:

```
std::vector<int> vec;
vec.push_back(10);
vec.push_back(20);

for (int i : vec)
{
   std::cout << i << std::endl;
}
```

Range-based for with “auto”

- Can be combined with auto:

```
std::vector<int> vec;
vec.push_back(10);
vec.push_back(20);

for (auto i : vec) // makes copies of each i
{
   i += 10; // DOES NOT persist outside of loop
}
```

Range-based for with “auto”, modifying

- Can be combined with auto:

```
std::vector<int> vec;
vec.push_back(10);
vec.push_back(20);

for (auto& i : vec) // Modify the integers in vector
{
    i += 10; // Persists outside of loop
}
```