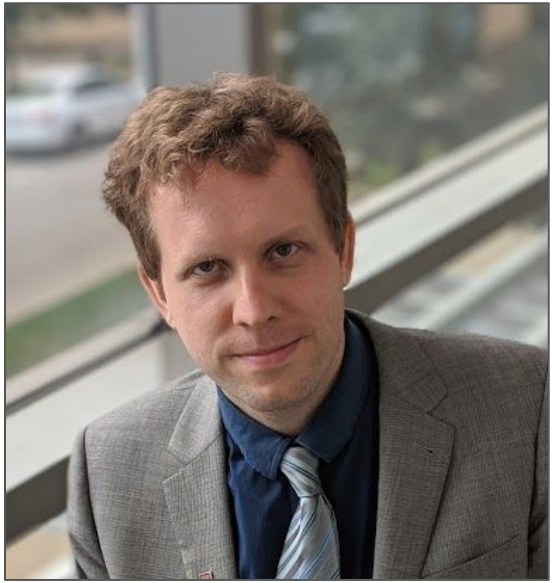
Welcome to CS 240!

CS 240 - The University of Illinois

Wade Fagen-Ulmschneider

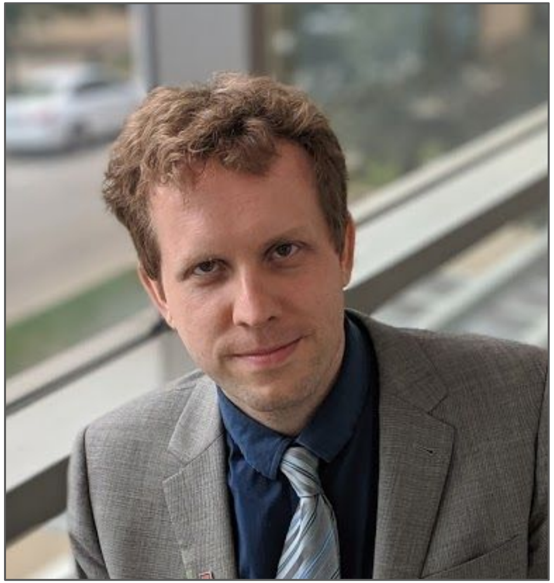
January 18, 2022

No good party starts without introductions...



Wade Fagen-Ulmschneider (waf)
Teaching Associate Professor of Computer Science
Grainger College of Engineering

Nerding out in life...

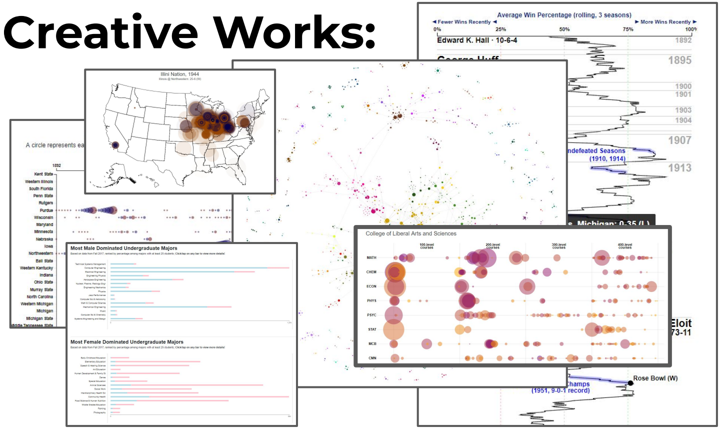


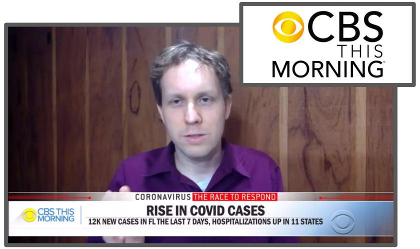
Wade Fagen-Ulmschneider (waf)
Teaching Associate Professor of Computer Science
Grainger College of Engineering

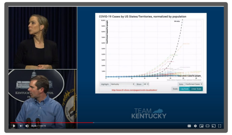
Industry:

Cisco → Google → Morgan Stanley

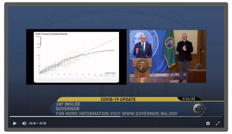
Creative Works:







Gov. Beshear (KY)

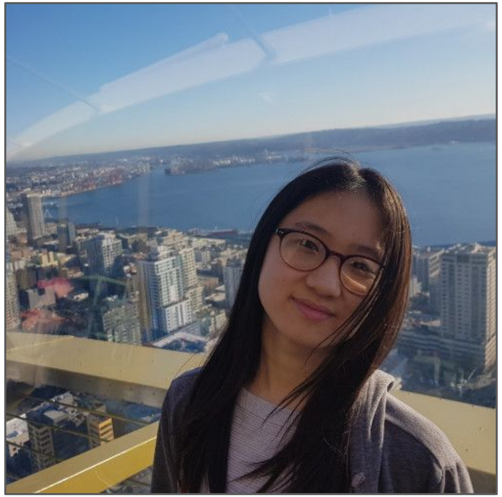


Gov. Inslee (WA)

Courses:

- MOOC: Accel. CS Fund.
- CS 241
- CS 105
- CS 305
- CS 225
- CS 240
- STAT/CS/IS 107

Teaching Assistants



Eunice Zhou

Patrick Crain

Course Assistants

Bora Shim

Jeremy Shaffar

Jackson Kennel

Kevin Chen

You:

Overview: You Already Know

Overview: You Already Know

C++ Programming (CS 225)

Overview: You Already Know

C++ Programming (CS 225)

Data Structures (CS 225)

Overview: You Already Know

C++ Programming (CS 225)

Data Structures (CS 225)

Algorithm Analysis (CS 173)

Overview: You Already Know

C++ Programming (CS 225)

Data Structures (CS 225)

Algorithm Analysis (CS 173)

Programming Skills (CS 125/126/225)

Overview: After CS 240

Overview: After CS 240

Foundational Computer Architecture

Overview: After CS 240

Foundational Computer Architecture

Operating System Design

Overview: After CS 240

Foundational Computer Architecture

Operating System Design

Multiprogramming and Resource Sharing

Overview: After CS 240

Foundational Computer Architecture

Operating System Design

Multiprogramming and Resource Sharing

Cloud-based Infrastructure

Overview: After CS 240

Foundational Computer Architecture

Operating System Design

Multiprogramming and Resource Sharing

Cloud-based Infrastructure

Building Cloud-scale Applications

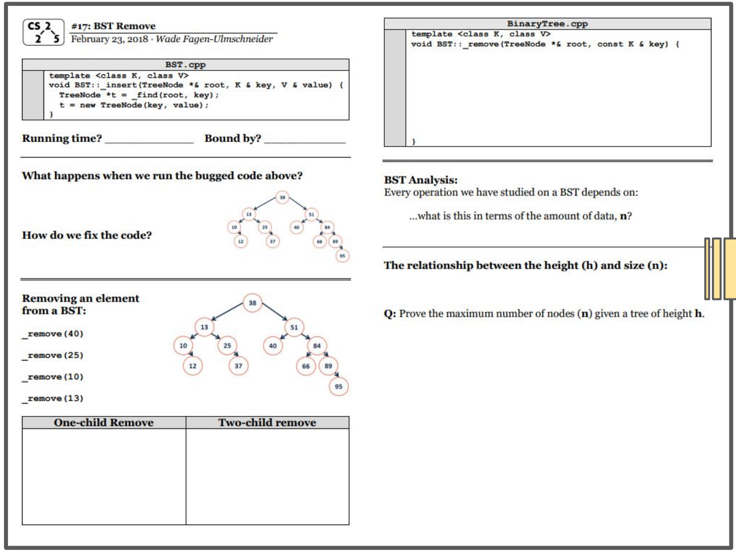
Course Structure

Course Structure

★ **Lecture: Tuesday/Thursdays**

Course Structure

★ **Lecture: Tuesday/Thursdays**



CS 225 #17: BST Remove

February 23, 2018 · *Wade Fagen-Ulmschneider*

BST.cpp

```
template <class K, class V>
void BST::_insert(TreeNode *& root, K & key, V & value) {
  TreeNode *t = _find(root, key);
  t = new TreeNode(key, value);
}
```

Running time? ______________ **Bound by?** ______________

What happens when we run the bugged code above?

How do we fix the code?

Removing an element from a BST:

_remove(40)

_remove(25)

_remove(10)

_remove(13)

One-child Remove	Two-child remove

BinaryTree.cpp

```
template <class K, class V>
void BST::_remove(TreeNode *& root, const K & key) {

}
```

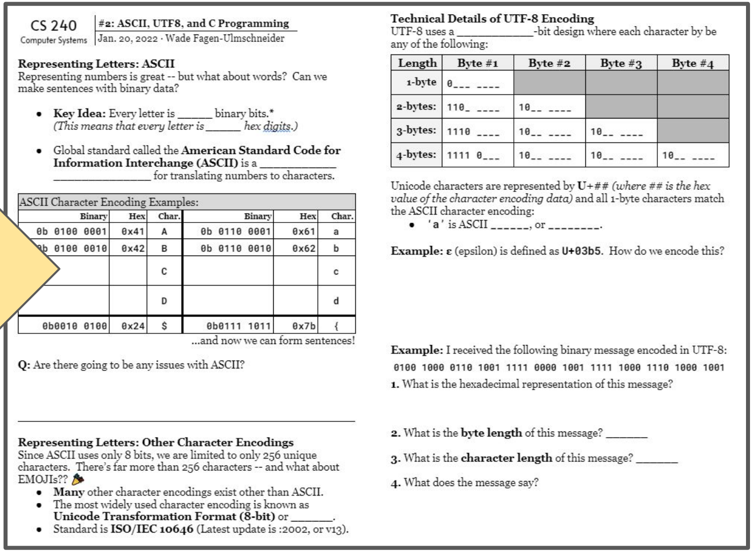
BST Analysis:
Every operation we have studied on a BST depends on:

...what is this in terms of the amount of data, **n**?

The relationship between the height (h) and size (n):

Q: Prove the maximum number of nodes (**n**) given a tree of height **h**.

CS 225 (Spring 2018)



CS 240 Computer Systems #2: ASCII, UTF8, and C Programming

Jan. 20, 2022 · Wade Fagen-Ulmschneider

Representing Letters: ASCII
Representing numbers is great -- but what about words? Can we make sentences with binary data?

- **Key Idea:** Every letter is ______ binary bits.* *(This means that every letter is ______ hex digits.)*
- Global standard called the **American Standard Code for Information Interchange (ASCII)** is a ______________ for translating numbers to characters.

ASCII Character Encoding Examples:

Binary	Hex	Char.	Binary	Hex	Char.
0b 0100 0001	0x41	A	0b 0110 0001	0x61	a
0b 0100 0010	0x42	B	0b 0110 0010	0x62	b
		C			c
		D			d
0b0010 0100	0x24	$	0b0111 1011	0x7b	{

...and now we can form sentences!

Q: Are there going to be any issues with ASCII?

Representing Letters: Other Character Encodings
Since ASCII uses only 8 bits, we are limited to only 256 unique characters. There's far more than 256 characters -- and what about EMOJIs??

- **Many** other character encodings exist other than ASCII.
- The most widely used character encoding is known as **Unicode Transformation Format (8-bit)** or ______.
- Standard is **ISO/IEC 10646** (Latest update is :2002, or v13).

Technical Details of UTF-8 Encoding
UTF-8 uses a ____________-bit design where each character by be any of the following:

Length	Byte #1	Byte #2	Byte #3	Byte #4
1-byte	0___ ____			
2-bytes:	110_ ____	10__ ____		
3-bytes:	1110 ____	10__ ____	10__ ____	
4-bytes:	1111 0___	10__ ____	10__ ____	10__ ____

Unicode characters are represented by **U+##** *(where ## is the hex value of the character encoding data)* and all 1-byte characters match the ASCII character encoding:

- 'a' is ASCII ______, or ________.

Example: ε (epsilon) is defined as **U+03b5**. How do we encode this?

Example: I received the following binary message encoded in UTF-8:

`0100 1000 0110 1001 1111 0000 1001 1111 1000 1110 1000 1001`

1. What is the hexadecimal representation of this message?
2. What is the **byte length** of this message? ______
3. What is the **character length** of this message? ______
4. What does the message say?

CS 240 (This Thursday, Lecture #2!)

Course Structure

- ★ **Lecture: Tuesday/Thursdays**
- ★ **Weekly MPs and PL Homework**

Course Structure

- Lecture: Tuesday/Thursdays
- Weekly MPs and PL Homework
- Two Exams in the CBTF

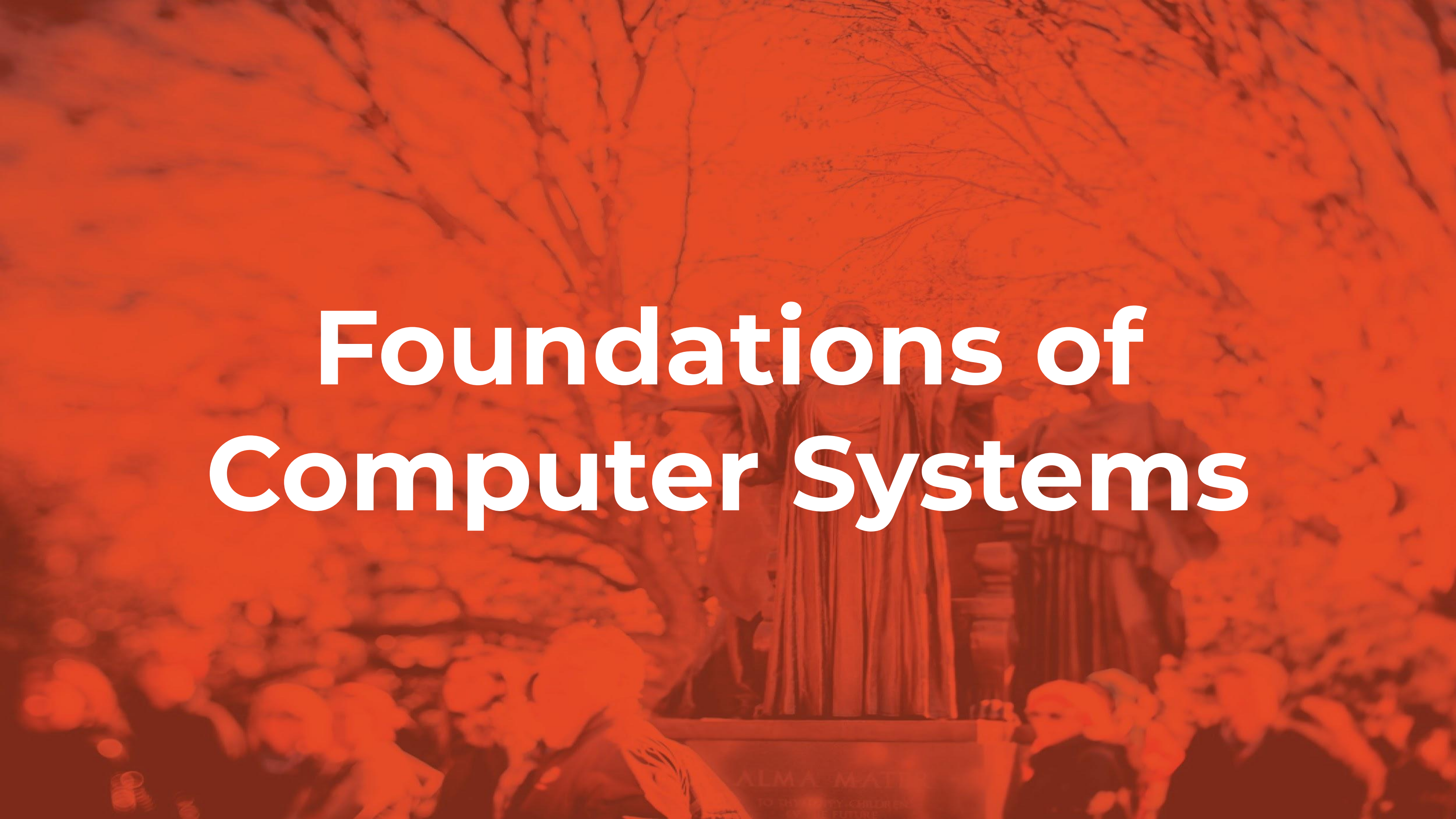
Course Structure

- ★ Lecture: Tuesday/Thursdays
- ★ Weekly **MPs** and **PL Homework**
- ★ Two Exams in the CBTF
- ★ Final Course Project

Everything Else:

https://courses.grainger.illinois.edu/cs240/

Foundations of Computer Systems



Computer Systems Foundations

Computer Systems Foundations

#1: Data

Computer Systems Foundations

#2: Central Processing Unit

Computer Systems Foundations

#3: Memory and Storage

Computer Systems Foundations

#4: Peripherals

Computer Systems Foundations

#5: Operating System

Computer Systems Foundations

#6: Processes

System-Level Abstractions

System Level Abstractions

System Level Abstractions

#1: Virtual Machine

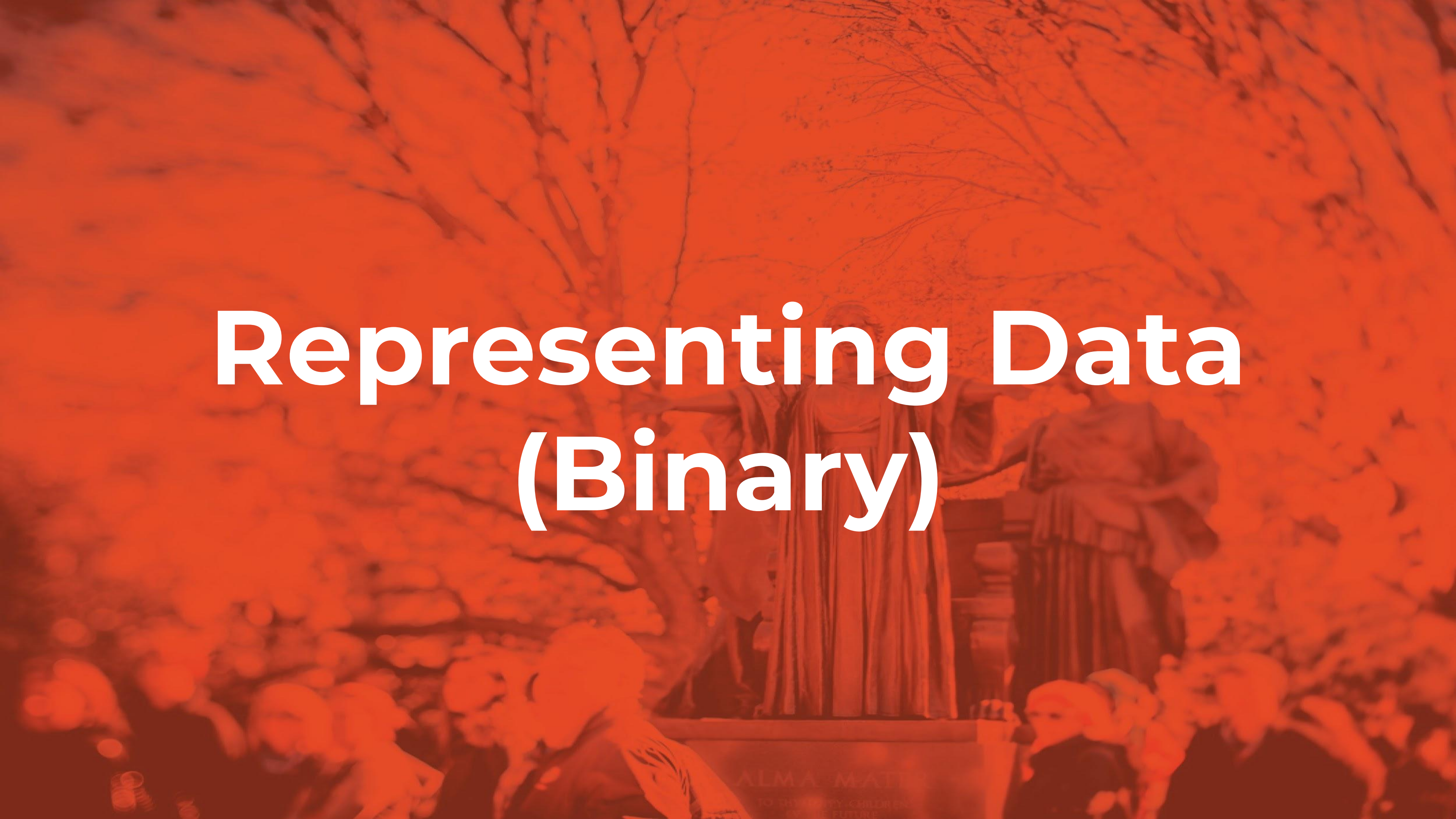
System Level Abstractions

#2: Containers

System Level Abstractions

#3: Nodes / Servers in the “Cloud”

Representing Data (Binary)



Representing Data

All data within a computer is:

$$1_2 = \qquad\qquad {}_{10}$$

$$10_2 = \qquad\qquad {}_{10}$$

$$11_2 = \qquad\qquad {}_{10}$$

$$100_2 = \qquad\qquad {}_{10}$$

$$_{2} = \qquad\qquad {}_{10}$$

$$1_{2} = \qquad\qquad {}_{10}$$

$$10_{2} = \qquad\qquad {}_{10}$$

$$11_{2} = \qquad\qquad {}_{10}$$

$$100_{2} = \qquad\qquad {}_{10}$$

$101\ 1000_2 =$ $_{10}$

Place Value of Digits

$$1 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0_2$$

$$2^6 \quad 2^5 \quad 2^4 \quad 2^3 \quad 2^2 \quad 2^1 \quad 2^0$$

Place Value of Digits

1	0	1	1	0	0	0_2
64	32	16	8	4	2	1_{10}

Place Value of Digits

	1	0	1	1	0	0	0_2
×	64	32	16	8	4	2	1_{10}

Place Value of Digits

	1	0	1	1	0	0	0_2
×	64	32	16	8	4	2	1_{10}
	64	0	16	8	0	0	0_{10}

Place Value of Digits

$1 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0_2$

$64 \quad 32 \quad 16 \quad 8 \quad 4 \quad 2 \quad 1_{10}$

$64 + 0 + 16 + 8 + 0 + 0 + 0_{10}$

Place Value of Digits

$$1 \quad 0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0_2$$

$$64 \quad 32 \quad 16 \quad 8 \quad 4 \quad 2 \quad 1_{10}$$

$$64 + 0 + 16 + 8 + 0 + 0 + 0_{10}$$

$$\boxed{=88_{10}}$$

$$4_{10} = \qquad\qquad _2 = \texttt{0b}$$

$$7_{10} = \qquad\qquad _2 = \texttt{0b}$$

$$18_{10} = \qquad\qquad _2 = \texttt{0b}$$

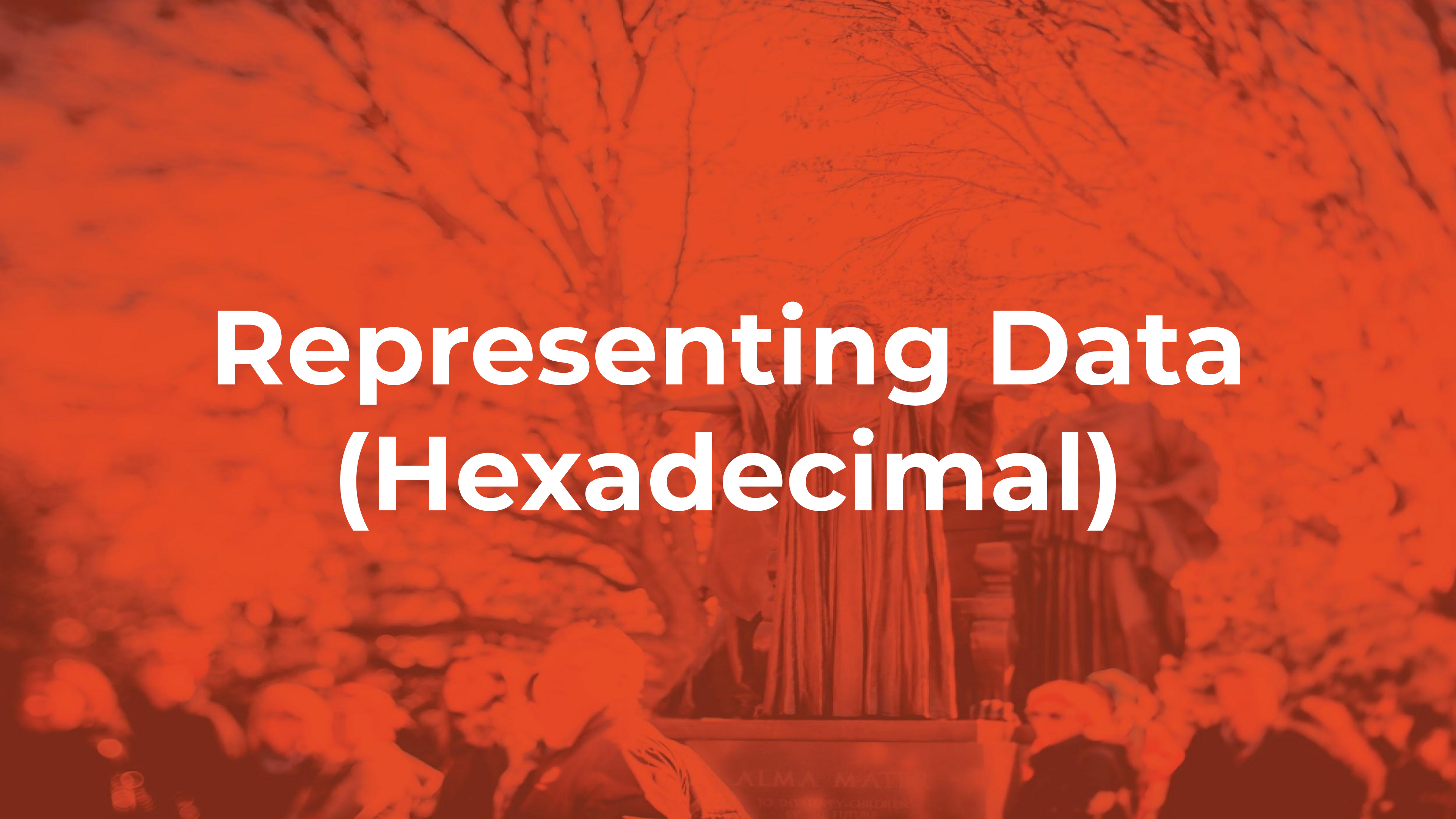
01/binary.c

```c
#include <stdio.h>

int main() {
  int v1 = 0b10010;
  int v2 = 0b11001;
  int v3 = v1 + v2;
  printf("%d\n", v3);

  return 0;
}
```

Representing Data (Hexadecimal)



Binary Digits

Number of Students at Illinois:

`0b 1100 1100 0110 1011`

Hexadecimal

Digits:

Place Value of Digits

0x	c	0	f	f	e	e
	16^5	16^4	16^3	16^2	16^1	16^0

Place Value of Digits

0x	c	0	f	f	e	e
	16^5	16^4	16^3	16^2	16^1	16^0
	12×16^5	0×16^4	15×16^3	15×16^2	14×16^1	14×16^0

Place Value of Digits

0x	c	0	f	f	e	e
	1048576	65536	4096	256	16	1
	12×16^5	0×16^4	15×16^3	15×16^2	14×16^1	14×16^0
	12582912	0	61440	3840	224	15

Place Value of Digits

0x c 0 f f e e

$$\boxed{=12{,}648{,}430_{10}}$$

$$11_{10} = \texttt{0x}$$

$$34_{10} = \texttt{0x}$$

$$87_{10} = \texttt{0x}$$

$$255_{10} = \texttt{0x}$$

1 = 0x1
2 = 0x2
3 = 0x3
4 = 0x4
5 = 0x5
6 = 0x6
7 = 0x7
8 = 0x8

9 = 0x9
10 = 0xa
11 = 0xb
12 = 0xc
13 = 0xd
14 = 0xe
15 = 0xf

Students at Illinois:

`0b 1100 1100 0110 1011`

People Following Tay on Twitter:

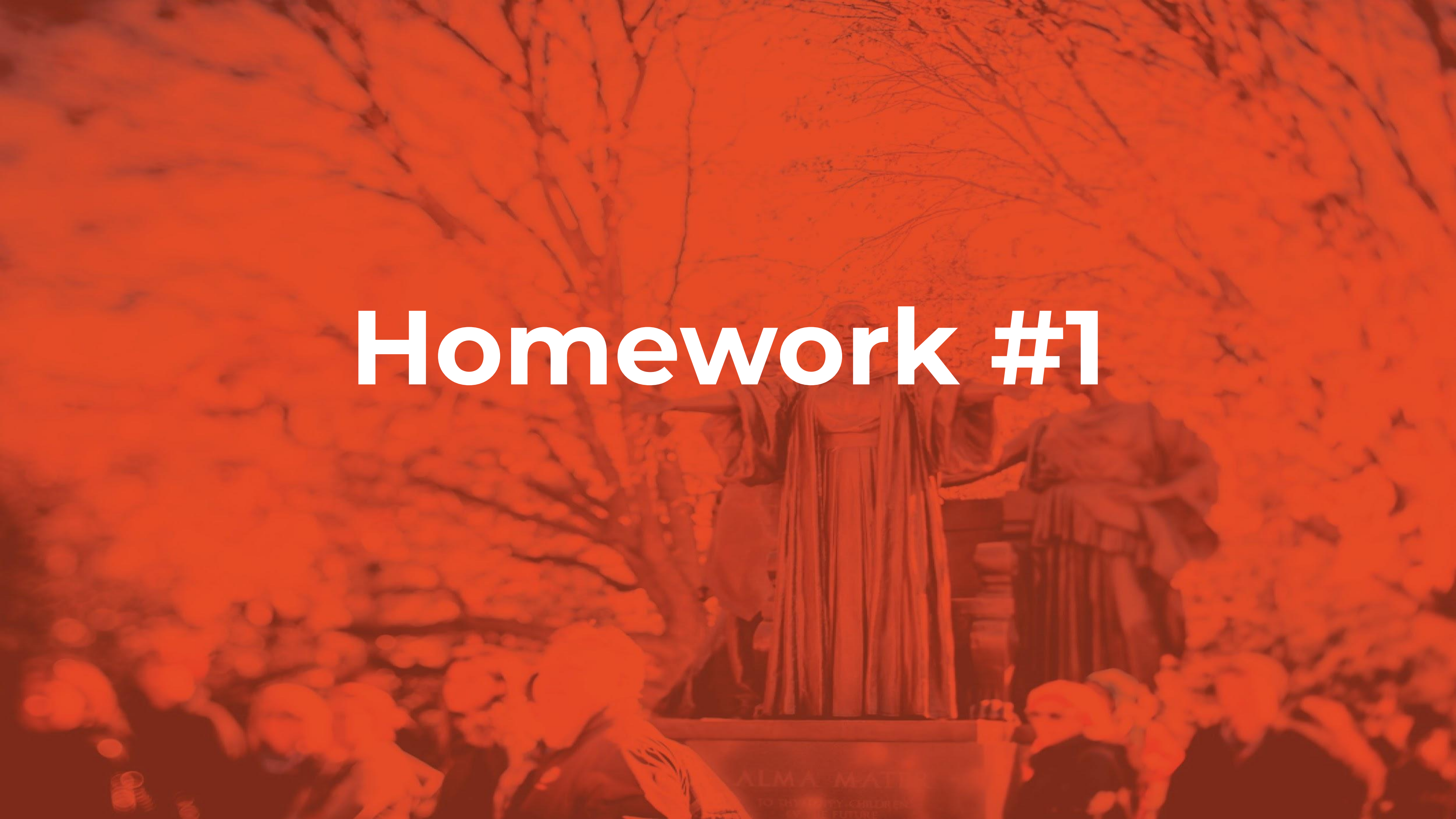
101 0100 1001 0010 1010 0110 0000

01/hex.c

```c
#include <stdio.h>

int main() {
  int h1 = 0xc0ffee;
  int h2 = 0xf00d;
  printf("%x\n", h1 + h2);

  return 0;
}
```


Homework #1

PrairieLearn #01