

# Mid Quarter Review

CSE 333 Spring 2021

**Instructor:** Justin Hsia, Travis McGaha

**Teaching Assistants:**

Arthava Deodhar

Callum Walker

Cosmo Wang

Dylan Hartono

Elizabeth Haker

Kyrie Dowling

Leo Liao

Markus Schiffer

Neha Nagvekar

Nonthakit Chaiwong

Ramya Challa

# Administrivia

- ❖ Exercise 8 due Monday
- ❖ No Lecture Friday!
- ❖ Midterm “exam” due Friday (5/7) @ 11:59 pm
- ❖ Homework 3 due Thursday (5/20) @ 11:59 pm
  - Due date is two weeks from tomorrow
  - Partner sign-up & Partner Finding form already open on Ed

# Lecture Outline

- ❖ **Reflection**
- ❖ Review Questions!

# Main Topics (so far)

- ❖ C
  - Low-level programming language
- ❖ C++
  - The 800-lb gorilla of programming languages
  - “better C” + classes + STL + smart pointers + ...
    - More to come soon 😊
- ❖ Memory management
- ❖ System interfaces and services
  - More to come later in the quarter

# Low-Level Programming

- ❖ The layer between CSE 351 and Java
  - Understanding the “layer below” makes you a better programmer at the layer above
  - More to come on this with Inheritance
  
- ❖ Memory & Resource management
  - Pointers!!!!
  - C/C++ won't clean up for you!
  
- ❖ Robustness and error handling
  - Code must be well thought out
  - We must be disciplined!

# The Operating System

- ❖ The Operating System
  - Complicated software that has permission to interact with hardware
- ❖ System Calls
  - Interface for users to request protected OS operations
  - Some library calls (fread/fwrite/...) will also have to go through the OS via system calls.
- ❖ I/O
  - Reading/Writing to disk takes a LONG time
    - (relative to other operations)
  - Strategies like buffering should be used to minimize number of disk accesses.
- ❖ More OS features coming later!

# Lecture Outline

- ❖ Reflection
- ❖ **Review Questions!**



[pollev.com/cse333travis](https://pollev.com/cse333travis)

- ❖ We don't have a "normal" midterm this quarter
- ❖ For the rest of lecture, we will do some conceptual questions to reflect on what we learned in the first half of the course.

# Low Level Programming Review

- ❖ The “plan” for this question
  - Send everyone to breakouts
  - 1 person per breakout shares their screen and goes to the shared google slides here:
    - Lecture A:  
<https://docs.google.com/presentation/d/1ZwvmEkl0zqW6RcmVaNh1Hb41ITqiHes4qwbebWVG4UM/edit?usp=sharing>
    - Lecture B:  
<https://docs.google.com/presentation/d/1gaH16FYe3X37V2Mzs06aWj3qZCCyITJZW3SpBdnxyAA/edit?usp=sharing>
  - Work with & Discuss with neighbors!

# Low Level Programming Review

- ❖ Complete the function "rand\_string", which generates a random string of random length. Assume we have the following functions available to you:

- `int32_t rand_len();`

- `// returns a random int in the range of 1 - 256`

- `char rand_char();`

- `// returns a random printable character`
  - `// (no '\0' or other special characters)`

[pollev.com/cse333travis](http://pollev.com/cse333travis)

- ❖ Provided are three different ways to read the contents of a large file. Rank the implementations by their efficiency.
  - Assume that buffers are allocated and files opened/closed for you

```
fread(buf, LEN, sizeof(char), file);  
return buf;
```

Implementation #1

```
while(read(fd, buf+numread, sizeof(char)) != 0) {  
    // ...  
    numread += sizeof(char);  
}  
return buf;
```

Implementation #2

```
while((res = read(fd, buf+numread, LEN - numread) != 0) {  
    // ...  
    numread += res;  
}  
return buf;
```

Implementation  
#3

[pollev.com/cse333travis](https://pollev.com/cse333travis)

## Two questions on Makefiles!

- ❖ What benefit does a Makefile have over a shell script?
  - (A shell script would just run all the commands to recompile)
- ❖ Why do we include header files in the sources list, but not in the compilation command?

▪ Example:

```
main: main.cc util.h
    gcc -g -Wall -std=c17 -o main main.cc
```

 **Poll Everywhere**[pollev.com/cse333travis](https://pollev.com/cse333travis)

Should we use a reference?

**A. We must NOT use a reference**

**B. It's OK but discouraged to use a reference**

**C. It's OK and encouraged to use a reference**

**D. We must use a reference**

**E. We're lost...**

```
Complex1.h
#ifndef _COMPLEX_H_
#define _COMPLEX_H_

#include <iostream>

namespace complex {

class Complex {
public:
    // Copy constructor, should we
    // pass a reference or not? (Answer: ?)
    Complex(const Complex &copyme) {
        real_ = copyme.real_;
        imag_ = copyme.image_;
    }

private:
    double real_, imag_;
}; // class Complex

} // namespace complex

#endif // _COMPLEX_H_
```

 **Poll Everywhere**[pollev.com/cse333travis](http://pollev.com/cse333travis)

Should we use a reference?

- A. We must NOT use a reference
- B. It's OK but discouraged to use a reference
- C. It's OK and encouraged to use a reference
- D. We must use a reference
- E. We're lost...

```
#include <iostream> Complex2.h
namespace complex {
class Complex {
public:
    // Should operator+ return a reference or not?
    // (Answer: ?)
    Complex &operator+(const Complex &a) const {
        Complex tmp(0,0);
        tmp.real_ = this->real_ + a.real_;
        tmp.imag_ = this->imag_ + a.imag_;
        return tmp;
    }

private:
    double real_, imag_;
}; // class Complex
} // namespace complex
```