Do you want to do research?

Come apply to PURE!

Promoting Undergraduate Research in Engineering

Benefits:

- Research experience
- Networking
- Soft and hard skill development
- 1 credit hour + GPA boost
- Resume Booster

Scan for:
- Interest form
- Website
- Discord

No rec letter needed!
(Optional) Open Lab This Week

This week’s lab is open office hours

Focus is making sure your machine is setup for semester

Installation information available on website
Exam 0 (August 29 — 31)

https://courses.engr.illinois.edu/cs225/fa2023/exams/

An introduction to CBTF exam environment / expectations

Quiz on foundational knowledge from all pre-reqs

Practice questions can be found on PL

Registration starts August 24
Learning Objectives

A brief high level review of C++

- Fundamentals of Classes
- The Rule of Three
- Memory management
- Function parameters and const
- Templates

Introduce Abstract Data Types (ADT)
Encapsulation - Classes

- Organizing code
- Implementation
- Private details
- How I do what I do
- How I can interact with a class
- Interface
  - Public facing portion
  - Check for correctness
  
  OOP Library → int
class Library {
public:

  CheckOutBook();
  CheckInBook();
  Book InLibrary();

private:

  class BookShelf users
  Capcity delete

};

public (in use)

(implementation)
Class Fundamentals

Constructor — Called when you make an instance of class

library(Behavior)

( ...), use(List) ... 

Destructor — Last thing ever called by class
— Destroyed
Class Fundamentals

Does our library class need a destructor?

Unsure
The Rule of Three

If it is necessary to **define any one** of these three functions in a class, it will be necessary to **define all three** of these functions:

1. *Destructor*

2. *Copy Constructor*

3. *Copy Assignment*
```cpp
class Library {
public:
    int numBooks;
    std::string * titles;
    ~Library();
    Library( int num, std::string* list );
};

Library::~Library(){
    delete titles;
    titles = nullptr;
}

Library::Library(int num, std::string* list){
    numBooks = inNum;
    titles = new std::string[ inNum ];
    std::copy(inList, inList + inNum, titles);
}

int main(){
    Library L1( 3, myBooks );
    Library L2( L1 );
    return 0;
}
```
What's wrong with this code?

A. Can't create L2 Library obj
B. Don't delete either Library
C. Deleting L1 deletes L2
‘The Rule of Zero'

If you define a destructor, copy, or assignment operator, *you should define all three!*

Implicit default operators are generated otherwise.

**Tip:** If you can, avoid writing these operators at all!
Memory Management

Stack
- Local variables managed by the computer
- Smaller than you might think (restrict)

Heap
- Dynamic storage managed by you
- New (delete)

Global
- Function
- Static int X;
- int main()
Reference and Dereference

Reference (&) - gives me address

Dereference (*) - gives me value

```cpp
int a = 3;
int b = 5;

int *p = &a;
int &r = b;

cout << p << " " << *p << endl;
cout << r << endl;

p++;
r++;
cout << a << " " << b << endl;
cout << p << " " << *p << endl;
cout << r << endl;
```

```cpp
1 2 3 4 5
6 7 8 9 10
11 12 13 14 15
16 17 18 19 20
21 22 23 24 25
```
Students after class were confused about the values here

Blue is the most relevant detail! Red is also important

```cpp
int a = 3;
int b = 5;

int *p = &a; // Value: 0xfffffc6216cc
int &r = b; // Value: 5

cout << p << " " << *p << endl; // Output: "0xfffffc6216cc 3"
cout << r << endl; // Output: "5"

p++; // This increments the POINTER ADDRESS
r++; // This increments b's value

cout << a << " " << b << endl; // "3 6"
cout << p << " " << *p << endl; // "0xfffffc6216d0 -60680480"
cout << r << endl; // Output: "6"
```
Memory Management - Parameters

Value

Value — Pointer

Reference

```c
int * p = &b;  // allocate a new copy

int * l = new library;
```

```c
int * c = b;  // allocate a new copy
int * r = b;  // allocate a new copy
```
class Library {
public:
    int numBooks;
    std::string * titles;
};

// *** Function A ***
std::string getFirstBook(Library l){
    return (l.numBooks > 0) ? l.titles[0] : "None";
}

// *** Function B ***
std::string getFirstBook(Library * l){
    return(l->numBooks > 0) ? l->titles[0] : "None";
}

// *** Function C ***
std::string getFirstBook(Library & l){
    return (l.numBooks > 0) ? l.titles[0] : "None";
}
Memory Management

Local memory on the stack is managed by the computer

Heap memory allocated by `new` and freed by `delete`

Understand when and how to use reference (`&`) and dereference (`*`) operators

**Tip:** If you can, avoid using `new` at all!
You are building a search tool over a collection of very large image files. One operation you want is to search an image for a particular pixel pattern (and return whether it exists or not). Assuming the query pattern and the input image are both of type `Image`, what might our function header look like?

```cpp
bool findImage(const Image& query, const Image& image);
```
The Const Keyword

**Const** means that an object cannot be modified

Variables

```
const int x = 5;
```

Pointers

\[
\text{Const int*} \\
\quad \Rightarrow \text{const change val}
\]

Reference

```
const int &
```

Method

```
class::Set Image () const
```
int x = 3;
int y = 2;

// *** A ***
const int* a = &x;
a = &y;

// *** B ***
const int* b = &x;
*b = y;

// *** C ***
int* const c = &x;
c = &y;

// *** D ***
int* const d = &x;
*d = y;
Const pointers vs const methods

```c
struct BlackBox {
    void update(const int & obj) {
        myVal = obj;
    }

    void update(int & obj) const {
        myVal = obj;
        obj++;
    }

    void update(const int & obj) const {
        myVal = obj;
        obj++;
    }

    int myVal;
};
```
Templates
T maximum(T a, T b) {
    T result;
    result = (a > b) ? a : b;
    return result;
}
List Abstract Data Type

A list is an ordered collection of items

Items can be either heterogeneous or homogenous

The list can be of a fixed size or is resizable
What types of “stuff” do we want in our list?

A list is an **ordered** collection of items.