### Cubes Unite!

Consider a Tower made of three Cubes:

```cpp
#include "cs225/Cube.h"
using cs225::Cube;

class Tower {
public:
    Tower(Cube c, Cube *ptr, const Cube &ref);
    Tower(const Tower & other);

private:
    Cube cube_;  // Copy
    Cube *ptr_;  // Deep copy
    const Cube &ref_;  // Deep copy
};
```

---

#### Automatic Copy Constructor Behavior:

The behavior of the automatic copy constructor is to make a copy of every variable. We can mimic this behavior in our Tower class:

```cpp
Tower::Tower(const Tower & other) {
    cube_ = other.cube_;  // Deep copy
    ptr_ = other.ptr_;  // Deep copy
    ref_ = other.ref_;  // Deep copy
}
```

...we refer to this as a deep copy because:

- Every variable is copied,
- The Copy Constructor's exact behavior is mimicked.

---

#### Destructor

The last and final member function called in the lifecycle of a class is the destructor. The automatic destructor:

1. Like a constructor and copy constructor, an automatic destructor exists only when no custom destructor is defined.
2. [Invoked]:
3. [Functionality]:

---

#### Custom Destructor:

```cpp
class Cube {
public:
    Cube();  // default ctor
    Cube(double length);  // 1-param ctor
    Cube(const Cube & other);  // custom copy ctor
    ~Cube();  // destructor, or dtor
...  // necessary if you need to delete any heap memory!
};
```
**Overloading Operators**

C++ allows custom behaviors to be defined on over 20 operators:

<table>
<thead>
<tr>
<th>Arithmetic</th>
<th>+  -  *  /  %  ++  --</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bitwise</td>
<td>&amp;</td>
</tr>
<tr>
<td>Assignment</td>
<td>=</td>
</tr>
<tr>
<td>Comparison</td>
<td>==  !=  &gt;  &lt;  &gt;=  &lt;=</td>
</tr>
<tr>
<td>Logical</td>
<td>!  &amp;&amp;</td>
</tr>
<tr>
<td>Other</td>
<td>[]  ()  -&gt;</td>
</tr>
</tbody>
</table>

**General Syntax:**

Adding overloaded operators to Cube:

```c++
#pragma once
class Cube {
public:
    // ...
    // ...
};
```

**Functionality Table:**

<table>
<thead>
<tr>
<th></th>
<th>Copies an object</th>
<th>Destroys an object</th>
</tr>
</thead>
<tbody>
<tr>
<td>Copy constructor</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Copy Assignment operator</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Destructor</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**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.

2.

3.

**One Very Powerful Operator: Assignment Operator**

```c++
Cube & operator=(const Cube & other);
```

**CS 225 and Rule Three/Five/Zero**

In CS 225 We will:

1. mp_stickers out today
2. Daily POTDs every M-F for daily extra credit!