Intro Classes and Overloading

Classes in C++

File layout

- .h
  - Declare structure of the object including member variables
  - Declare member functions (methods)

- .cpp
  - Define functions of the class

Constructors

- Run when an object is created before the object is available to make the object exist in a coherent state.

- Automatic Default Constructor
  - Automatic Default: created by the compiler
  - Default: takes no arguments
  - Initializes member objects using their default constructor
  - Non-object values undefined
  - Only generated if no constructors written

Constructors User Defined

- Functions with the same name as the class define constructors
- Initialization Lists
  - Comma separated list
  - member_variable_(expression)
  - Run before the body of the constructor
- Default Arguments
  - Provide values for missing agruments

```
explicit StudentRecord(std::string name, int grade = 0);


StudentRecord::StudentRecord(std::string name, int grade) : name_(name),
grade_(grade) {
      //body
}
```

Gradebook class

What System do you use for this class?

A. Windows
B. Macintosh
C. Linux
D. More than one of the above

What development environment did you set up for C++?

A. CLion
B. Visual Studio
C. Xcode
D. Editors and command line
E. What is a development environment?

Operator Overloading

Make code more readable and intuitive by allowing more natural expression

Change the behavior of many of the standard operators based on the types that are being used.

We have seen the overload of [] with **`map`** and **`vector`**

We have seen the overload of -> and * with the iterator in **`map`**

Operators that can be overloaded in C++

Operator Category	Operators
Arithmetic	+ - * / % ++ --
Bitwise	& | ^ ~ >> <<
Relational	< <= > >= == !=
Logical	! && ||
Assignment	= += -= *= /= %= ˆ= &= |= >>= <<= <= >=
Other	() [] -> , ->

Stream extraction and insertion

```
std::ostream& operator<<(std::ostream& os, const T& obj){
     // write obj to stream
     return os;
}

std::istream& operator>>(std::istream& is, T& obj){
     // read obj from stream
     if( /* T could not be constructed */ )
          is.setstate(std::ios::failbit);
     return is;
}
```

Student Record