Outline

1. Last class:
    a. Intro. to class: encapsulate related data variables and functions that work on these data into a type
    b. Intro. to syntax of 1) define a class type, 2) defining member functions, 3) constructors
    c. Private members vs. public members (variables or functions)
        i. Private members can only be accessed by member functions of the same class
        ii. Public members can be accessed by member functions, other classes, and any functions (such as main).
2. Trace programs with objects (i.e., a variable that is of a class type):
    a. Every object takes up a block of memory to store all its member variables
    b. When calling a member function on an object (i.e., invoking object), the object is passed as an implicit parameter to the function, as a result, you don't need to refer to the invoking object.

       Note: you could refer explicitly to the invoking object, using **this** pointer.

```
class Box
{
  public:
    // Constructor definition
    Box(double l=2.0, double b=2.0, double h=2.0)
    {
      cout <<"Constructor called." << endl;
      length = l;    // same as  (*this).length = l;
      breadth = b;
      height = h;
    }
    double Volume()
    {
      return length * breadth * height; //refer to
          // the invoking object's length, breadth and height …
    }
   bool compare(Box box)
    {
     // Comparing the volumes of the invoking object, and the "box"
    // (object passed as parameter)
      return Volume() > box.Volume();
    }
  private:
    double length;     // Length of a box
    double breadth;    // Breadth of a box
    double height;     // Height of a box
```

```
};

int main()
{
  Box Box1(3.3, 1.2, 1.5);    // Declare box1
  Box Box2(8.5, 6.0, 2.0);    // Declare box2

 // In the following call to Volume(), Box1 is the invoking object
  cout <<"Box1's volume" << Box1.Volume() << endl;

 // In this call, Box2 is the invoking object
  cout <<"Box2's volume" << Box2.Volume() << endl;

 //Box1  is the invoking object, Box2 is passed as parameter
  if(Box1.compare(Box2))
  {
    cout << "Box2 is smaller than Box1" <<endl;
  }
  else
  {
    cout << "Box2 is equal to or larger than Box1" <<endl;
  }
  return 0;
}
```

3. Error handling
    a. Usually we can use return value to indicate something goes wrong, e.g., when inserting a value to an array that is already full
    b. assert () function is usually used in debugging phase of a program , as a product software shouldn't end so abruptly (no chance to save results into memory, etc.)
    c. When something really bad happens (such that the program should not continue), you should use exit function to terminate the whole program

        ```
        exit (1);   // terminate the program, and return a value of 1 to indicate failure
        ```

        of course, if you are in main, you can still use

        ```
        return (1); // terminate current function, if it's the main function, then the program ends
        ```

    d. If something goes wrong in constructor, we should terminate the whole program

4. Midterm Reviews
    a. What you should do to prepare:
       Midterm practice labs, exercises
       Review handouts, notes, books
       Review labs, homework, and quiz
       Make up a cheat sheet: what can be on it, and what cannot be on it.

    b. In-class Review: hw0, hw1, quiz1, lab2, lab3, lab4, lab5 and lab6
    c. Extra office hour: this Friday