Quiz Number 6
(Closed Book/Closed Notes: 12 Minutes)
Tuesday, April 25, 2006

// in EggRoll.cc
class EggRoll
{
    private:
    int num; // egg roll number

    public:
    EggRoll() { cout << "c'tor:1" << endl; num = 100; } // c'tor: sets default egg roll number to 100
    EggRoll(int x) { cout << "c'tor:2" << endl; num = x; } // c'tor: accepts egg roll number
    int getNum() { return num; } // return the roll number
    void incrNum() { ++num; } // increase the roll number
};

const EggRoll operator+(const EggRoll& lhs, const EggRoll& rhs)
{
    cout << "* in global operator+" << endl;
    return EggRoll(lhs.getNum() + rhs.getNum()); // add the two and return a new eggroll object
}

int main()
{
    EggRoll e(10); // new eggroll object with roll number 10
    EggRoll b;
    EggRoll c = e + 10; // a third eggroll object
    return 0;
}

Question 1:
A. Would the code above work? If not, fix it only by making addition(s) to the existing code.
   No. See fix above.

B. Irrespective of whether the code compiles, write the output generated if the code were executed.
   c'tor:2
   c'tor:1
   c'tor:2
   * in global operator+
   c'tor:2

Question 2:
A. If one wants to increment the "egg roll number" of a given EggRoll object (using only the above class definition), can we:
   - have the prefix ++ operator overloaded? (yes / no)
   - have the postfix ++ operator overloaded? (yes / no)

   Give a reason for your choices (in one line).
   Increasing the number can be done with either, using incrNum().
B. Provide implementations for both the postfix and prefix `++` operators.

```cpp
const EggRoll& operator++(EggRoll& myRoll)
{ // prefix ++ operator
  myRoll.incrNum(); // increase the roll number
  return myRoll; // return the updated object
}

const EggRoll operator++(EggRoll& myRoll, int dummy)
{ // postfix ++ operator
  EggRoll old = myRoll; // copy the object
  myRoll.incrNum(); // increase the number for the existing object
  return old; // return a copy of the old object
}
```