```
#include <iostream>
#include "rational.h"
using namespace std;

// default constructor
Rational::Rational() {
  numerator = 0;
  denominator = 1;
}

// (numer, denom) constructor
Rational::Rational(int numer, int denom) {
  numerator = numer;
  denominator = denom;
  Reduce();
}


// adding Rationals
Rational Rational::Add(const Rational &r) const {
  int a = numerator;     // this object's numerator
  int b = denominator;   // this object's denominator
  int c = r.numerator;   // the parameter's numerator
  int d = r.denominator; // the parameter's denominator

  Rational result;
  result.numerator = a*d + b*c;
  result.denominator = b*d;
  result.Reduce();

  return result;
}


// multiplying Rationals
Rational Rational::Multiply(const Rational &r) const {
  int a = numerator;     // this object's numerator
  int b = denominator;   // this object's denominator
  int c = r.numerator;   // the parameter's numerator
  int d = r.denominator; // the parameter's denominator

  Rational result;
  result.numerator = a*c;
  result.denominator = b*d;
  result.Reduce();

  return result;
}


// inserting a Rational
void Rational::Insert(ostream &sout) const {
  sout << numerator << '/' << denominator;
  return;
}


// extracting a Rational
void Rational::Extract(istream &sin) {
  // (hopefully everything goes well with user's input)
  char slash;
  sin >> numerator >> slash >> denominator;
  Reduce();   // reduce this rational
  return;
}
```

```
// Euclid's algorithm
// (note:  this is NOT begin defined within the Rational class)
int gcd(int a, int b) {
  if (a==0)
    return b;
  else
    return gcd(b%a,a);
}


void Rational::Reduce() {

  // we can use Euclid's algorithm to determine the gcd.

  if (numerator==0) {
    denominator = 1;
  } else {
    int g = gcd(numerator,denominator);
    numerator /= g;
    denominator /= g;
  }

  if (denominator<0) {
    numerator = -numerator;
    denominator = -denominator;
  }
}
```