```
//  SPECIFICATION FILE (fraction.h)
//  This module exports a FracType class with two
//  operator functions,
//  operator* and operator==.  A more general FracType class
//  might include more operations.

#include "bool.h"

class FracType {
public:
   void Write() const;
// POST: The value of this fraction has been displayed as:
//           <numerator> / <denominator>
//       with no blanks

   float FloatEquiv() const;
// POST: FCTVAL == float equivalent of this fraction

   void Simplify();
// POST: Fraction is reduced to lowest terms. (No integer > 1
//       evenly divides both the numerator and denominator)

   FracType operator*( /* in */ FracType frac2 ) const;
// PRE:  This fraction and frac2 are in simplest terms
// POST: FCTVAL == this fraction * frac2  (fraction
//                 multiplication), reduced to lowest terms
// NOTE: Numerators or denominators of large magnitude
//       may produce overflow
```

```
   Boolean operator==( /* in */ FracType frac2 ) const;
// PRE:  This fraction and frac2 are in simplest terms
// POST: FCTVAL == TRUE, if this fraction == frac2
//               == FALSE, otherwise

   FracType( /* in */ int initNumer,
             /* in */ int initDenom );
// Constructor
// PRE:  Assigned(initNumer)  &&  initDenom > 0
// POST: Fraction has been created and can be thought of
//       as the fraction  initNumer / initDenom
// NOTE: (initNumer < 0) --> fraction is a negative number
private:
   int numer;
   int denom;
};
```