1 rational.h

typedef struct rational_ {
    int numerator;
    int denominator;
} rational;

rational new_rat(int numerator, int denominator);
rational add_rat(rational x, rational y);
rational sub_rat(rational x, rational y);
rational mul_rat(rational x, rational y);
rational div_rat(rational x, rational y);
void print_rat(rational x);

Computation structures / concepts:

• Abstract datatype interface (data and functions)

2 rational.c

#include <stdio.h>
#include "rational.h"

static int gcd(int a, int b) 
{
    int rem;

    while ((a % b) != 0) {
        rem = a % b;
        a = b;
        b = rem;
    }

    return(b);
rational new_rat(int numerator, int denominator)
{
    rational answer;
    int divisor, sign = 1;

    if (numerator < 0) {
        sign = -1;
        numerator = -numerator;
    }

    divisor = gcd(numerator, denominator);
    numerator /= divisor;
    denominator /= divisor;

    answer.numerator = numerator * sign;
    answer.denominator = denominator;
    return (answer);
}

rational add_rat(rational x, rational y)
{
    int numerator, denominator;

    numerator = (x.numerator * y.denominator) + (x.denominator * y.numerator);
    denominator = x.denominator * y.denominator;

    return(new_rat(numerator, denominator));
}

rational sub_rat(rational x, rational y)
{
    int numerator, denominator;

    numerator = (x.numerator * y.denominator) - (x.denominator * y.numerator);
    denominator = x.denominator * y.denominator;

    return(new_rat(numerator, denominator));
}

rational mul_rat(rational x, rational y)
rational div_rat(rational x, rational y)
{
    int numerator, denominator;

    numerator = x.numerator * y.denominator;
    denominator = x.denominator * y.numerator;

    return(new_rat(numerator, denominator));
}

void print_rat(rational x)
{
    printf("%d/%d", x.numerator, x.denominator);
}

Computation structures / concepts:

- Abstract datatype code
- Information hiding using “static” keyword