EECS 10: Assignment 6

Prepared by: Prof. Rainer Doemer, Hadil Mustafa

October 27, 2010

Due on Monday 11/15/2010 12:00 noon. Note: this is a two-week assignment.

1 Calculations with Rational Numbers [80 points]

The goal of this assignment is to practice using functions in C programming by performing algebraic operations with rational numbers. Here are the rules for rational number operations:

$$\frac{a}{b} + \frac{c}{d} = \frac{ad + bc}{bd}$$
$$\frac{a}{b} - \frac{c}{d} = \frac{ad - bc}{bd}$$
$$\frac{a}{b} \times \frac{c}{d} = \frac{ac}{bd}$$
$$\frac{a}{b} \div \frac{c}{d} = \frac{ad}{bc}$$

1.1 The Option Menu

Your program should be a menu driven program. At the start of your program, the user is prompted to input a rational number by providing its numerator and denomiator. For instance, it may look like this:

```
Welcome to my rational number calculator!
Please input a rational number.
Numerator: 7
Denominator: 12
```

Then the program should display the current result and a menu. In our example, it looks like this:

```
-------------------------------------------
The current result is: 7/12
1. Subtract a rational number from the current result;
2. Add a rational number to the current result;
3. Multiply the current result by a rational number;
4. Divide the current result by a rational number;
5. Take the reciprocal of the current result;
6. Quit

Please enter a selection:
```

Your program should let the user select an option, 1 through 6. If the user selects 6, your program simply exits. For other selections, your program should perform the corresponding rational number operations.

When the user selects an operation (1 to 4), first ask the user to input another rational number as an operand (with the exception of the reciprocal and quit operations). Once the user inputs the operand (numerator

followed by denominator), your program should perform the required operation and display the current result and the option menu again.
In our example, the user selects 2, the add operation. So the program should ask the user to input a rational number to be added to 7/12, like this:

```
Please input a rational number operand.
Numerator: 5
Denominator: 12
```

Then the program should bring up the current result and menu again. This time the current result should display 1 (7/12 + 5/12 = 1).

```
-------------------------------------------
The current result is: 1
1. Subtract a rational number from the current result;
2. Add a rational number to the current result;
3. Multiply the current result by a rational number;
4. Divide the current result by a rational number;
5. Take the reciprocal of the current result;
6. Quit

Please enter a selection:
```

1.2 Printing the Current Rational Number

The current rational number always holds the result of the previous operation. At the beginning, there is no previous operation, so the current result just shows the rational number you input at the start. In our example, it is 7/12.

Before printing it, the current rational number should always be converted to the cononical (normal) form. For example, 12/28 should be reduced to 3/7 before printing.

Also, make sure that in the canonical form, the denominator of the rational number is always positive, while the numerator can be either positive or negative. That is, $-3/-7$ should be changed to 3/7, and $3/-7$ should be changed to $-3/7$.

Finally, there are two special cases you must handle:

1. a rational number like 0/7 should be printed out as 0
2. a rational number like 3/1 should be printed out as 3

1.3 The Greatest Common Divisor

In order to implement the function that reduces a rational number to its canonical form, you will need a helper function that calculates the Greatest Common Divisor (GCD) for two whole numbers. Your `gcd()` function should take two parameters and return the GCD of the parameters as a whole number. To get a hint on how to write this function, refer to the *pseudocode* below (from http://en.wikipedia.org/wiki/Euclidean_algorithm#Using_iteration):

```
function gcd(a, b)
    while b != 0
        t := b
        b := a mod b
        a := t
    return a
```

2 Implementation

To implement the program, you need to define a set of functions; each function does a specific job, such as addition, subtraction, and division etc.

2.1 Function Declarations

As a starting point, use the following function declarations:

```
int gcd(int, int);        /*greatest common divisor*/
void reduceResult();        /*change the current rational
                                    number to its reduced format*/
void printRational();     /*print the current rational number*/
void rAdd(int, int);      /*rational number addition*/
void rSubtract(int, int); /*rational number subtraction*/
void rMultiply(int, int); /*rational number multiplication*/
void rDivide(int, int);   /*rational number division*/
void rReciprocal();       /*reciprocal of the current number*/
```

Functions `rAdd`, `rSubtract`, `rMultiply`, and `rDivide` require two integers as their input parameters. Those two integers are the numerator and the denominator of the number to apply in the current operation; functions `reduceResult` and `printRational` require no input parameters because these will work only on the current number.

2.2 Global Variables

We will store the current rational number as global variables, as follows:

```
int currN;
int currD;
```

This way, all of your functions in the program have access to the current number.

In addition, it will be helpful to have the number that the user inputs as a global variable. We recommend the following to store the numerator and denominator of the user input.

```
int newN;
int newD;
```

2.3 Error handling

A robust program is able to handle all situations, and reports any problems to the user if an error occurs. To get full credit, your program should be able to handle the following errors:

- Notify the user with `ERROR: Denominator cannot be zero!` if the denominator of the input rational number is zero, and reprompt the user until a proper denominator is entered.
- Notify the user with `ERROR: Division by zero!` if the user selects the division operation and inputs zero as the operand, and reprompt the user until a proper operand is entered.

2.4 Budgeting your time

You have two weeks to complete this assignment, but we encourage you to get started early as this assignment is more difficult than previous ones. We suggest you budget your time as follows:

- **Week 1:**
  1. input, reduce, and print a rational number
  2. create your menu
- **Week 2:**
  1. complete the arithmetic operations
  2. add error handling

3 Extra credit: Mixed fractions [10 points]

A mixed fraction is a whole number and a fraction combined, such as 1 3/4. Change your program so that it can accept mixed fractions as input, and display the results as mixed fractions as well.

Hint: We highly recommended that when you accept a mixed fraction as input, convert it immediately to its non-mixed form (i.e., the form you use in the non-bonus part of this assignment). This way, you do not have to change the way you perform your calculations at all. All that changes is the input and output portions of your program.

The new option menu should look something like this:

```
Please input a rational or mixed rational number.
Whole part: 1
Numerator: 3
Denominator: 4

-----------------------------------------
The current result is: 1 3/4
1. Subtract a rational number from the current result;
2. Add a rational number to the current result;
3. Multiply the current result by a rational number;
4. Divide the current result by a rational number;
5. Take the reciprocal of the current result;
6. Quit

Please enter a selection:
```

4 What to submit

You need to save your program as *rational.c*.

The way you demonstrate your code will depend on whether or not you completed the extra credit. If you did not complete the extra credit, perform the following steps to generate the script file:

1. Compile and run your program
2. Input the rational number 1/3
3. Add the current result with 56/3
4. Subtract 17/9 from the current result

5. Multiply the current result by -5/4
6. Divide the current result by 5/2
7. Take the reciprocal twice
8. Add the current result with 77/9
9. Add the current result with 9/0 /*error handling*/
10. Divide the current result by 0/7 /*error handling*/
11. Exit your program

If you completed the extra credit, perform the following steps to generate the script file:

1. Compile and run your program
2. Input the rational number 1/3
3. Add 18 and 2/3 to the current result
4. Subtract 1 and 8/9 from the current result
5. Multiply the current result by -1 and 1/4
6. Divide the current result by 2 and 1/2
7. Take the reciprocal twice
8. Add the current result with 8 and 5/9
9. Add the current result with 9/0 /*error handling*/
10. Divide the current result by 0/7 /*error handling*/
11. Exit your program

For your reference, the result obtained after step 8 should be 0 in both cases.

Name the script file *rational.script*. If you forgot how to create a script file, check the instructions for homework 1.

You also need to create a *rational.txt* file, which briefly explains your program (if you did the extra credit, you also need to include comments on handling mixed fractions).

The submission is similar to the previous homeworks. You need to create a directory called *hw6*; put all the files *rational.c*, *rational.script*, and *rational.txt* in the directory and run the command */ecelib/bin/turnin* to submit your homework.

Note: We do require the *exact* file names. If you use different file names, we will not see your files for grading.

Also, please pay attention to any announcements on the course noteboard.