16.216: ECE Application Programming

Summer 2014

Programming Assignment #4: Reducing Fractions

Due **Friday, 6/6/14**, 11:59:59 PM

1. Introduction

This program will use an iterative algorithm—an algorithm that runs until a given condition is met—to reduce a fraction to its lowest terms. You will also use loops to ensure your application runs until the user explicitly ends the program.

2. Deliverables

Submit your source file directly to Dr. Geiger (Michael_Geiger@uml.edu) as an e-mail attachment. Ensure your source file name is ***prog4_GCD.c***. You should submit only the .c file. Failure to meet this specification will reduce your grade, as described in the program grading guidelines.

3. Specifications

Input: Your program will repeatedly prompt the user to enter a fraction in the form `<num>/<den>`, then reduce the fraction to its lowest terms using the greatest common divisor of `<num>` and `<den>`. Note that:

- Both the numerator `<num>` and denominator `<den>` are integers.
- The fraction should be entered with no spaces between the two numbers and the `'/'` character that separates them.
- If the denominator is 0, print an error message.

Once complete, the program should ask the user if he or she would like to reduce another fraction. If the user enters 'Y' or 'y', the program should return to its initial prompt. If the user enters 'N' or 'n', the program should exit. In all other cases, the program should print an error and repeat the question.

Output: Assuming there are no errors, your program should evaluate the inputs and print the original and reduced fraction, as well as the GCD. Sample input/output pairs are shown below, with the user input underlined:

-
  ```
  Enter fraction numerator/denominator: 16/80
  16/80 reduces to 1/5 (GCD = 16)
  ```

-
  ```
  Enter fraction numerator/denominator: 12/99
  12/99 reduces to 4/33 (GCD = 3)
  ```

-
  ```
  Enter fraction numerator/denominator: 100/40
  100/40 reduces to 5/2 (GCD = 20)
  ```

-
  ```
  Enter fraction numerator/denominator: 13/15
  13/15 reduces to 13/15 (GCD = 1)
  ```

Error checking: Your program should print an error under any of the following conditions:

- The denominator of the fraction is 0. In this case, ask the user if he or she would like to read another fraction.
- The fraction is improperly formatted. In this case, prompt for and read a new fraction after printing the error message.
- The user enters something other than 'Y', 'y', 'N', or 'n' when asked about reducing another fraction.

4. Formulating a solution—hints, tips, etc.

Calculating the greatest common divisor (GCD): To reduce a fraction to its lowest terms, you can divide both the numerator and denominator by their GCD. To calculate the GCD, you can use Euclid's algorithm—given two numbers, *x* and *y*, follow the steps below, which are repeated until the GCD is calculated:

1. If *y* is 0, *x* is the GCD.
2. Otherwise, calculate *r*, the remainder of *x* / *y*.
3. Let *x* = *y*, and *y* = *r*.
   - In other words, *x* holds the "old" value of *y*, and the new value of *y* is the remainder from Step 2.
4. Return to Step 1.

Reading character input: In most cases, `scanf()` will skip whitespace (spaces, tabs, newlines) when reading input. The exception to that rule comes when using the %c format specifier, which usually reads the next character in the input stream—space or otherwise. Given the following input:

```
5 3
X
```

Say you have the following code, assuming a and b are `ints` and c is a `char`:

```
scanf("%d %d", &a, &b);
scanf("%c", &c);
```

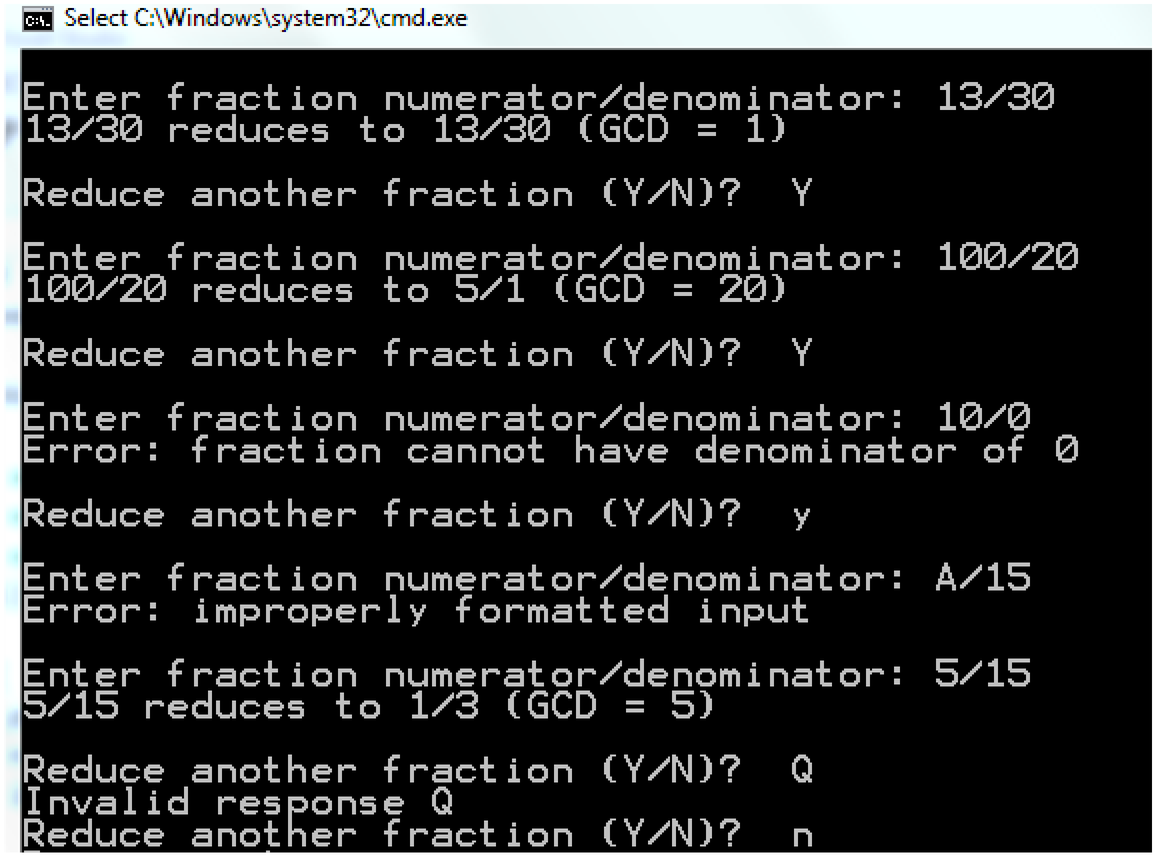
a and b will be 5 and 3, as expected; c, however, will hold the newline character, since that is the first input character after the integer 3. To avoid this problem, you can put a space in your format string, which will cause `scanf()` to skip whitespace characters and read the first character that follows the whitespace. Replace the second line above with:

```
scanf(" %c", &c);
```

Note that newlines in your input may not be obvious—you may enter values, print outputs based on those values, and then prompt for another input value.

5. Test Cases

Your output should match these test cases exactly for the given input values. I will use these test cases in grading of your lab, but will also generate additional cases that will not be publicly available. Note that these test cases do not cover all possible program outcomes. You should create your own tests to help debug your code and ensure proper operation for all possible inputs.



```
Enter fraction numerator/denominator: 13/30
13/30 reduces to 13/30 (GCD = 1)

Reduce another fraction (Y/N)?  Y

Enter fraction numerator/denominator: 100/20
100/20 reduces to 5/1 (GCD = 20)

Reduce another fraction (Y/N)?  Y

Enter fraction numerator/denominator: 10/0
Error: fraction cannot have denominator of 0

Reduce another fraction (Y/N)?  y

Enter fraction numerator/denominator: A/15
Error: improperly formatted input

Enter fraction numerator/denominator: 5/15
5/15 reduces to 1/3 (GCD = 5)

Reduce another fraction (Y/N)?  Q
Invalid response Q
Reduce another fraction (Y/N)?  n
```