EE231002 Introduction to Programming

Lab12. GCD and LCM

Due: Dec. 23, 2017

Given any positive integer N, $N > 1$, one can express this integer as a product of its prime factors. For example, if $N = 100$ then

$$N = 2^2 \times 5^2. \tag{12.1}$$

Since the number of prime factors are different from integer to integer, one of the good way to represent these prime factors is to use linked lists. The structure is defined below.

```
typedef struct factor {     // node for a prime factor
    int prime;              // prime factor
    int power;              // associated power
    struct factor *next;    // pointer for the next prime factor
} FACTOR;
```

Given two positive integers, N_1 and N_2, once both are expressed in the product of prime factors form, their Greatest Common Divisor (GCD) and Least Common Multiple (LCM) can also be found easily. For example, if $N_1 = 100$ and $N_2 = 225$, then

$$N_1 = 2^2 \times 5^2 \tag{12.2}$$

$$N_2 = 3^2 \times 5^2 \tag{12.3}$$

$$\text{GCD}(N_1, N_2) = 5^2 = 25 \tag{12.4}$$

$$\text{LCM}(N_1, N_2) = 2^2 \times 3^2 \times 5^2 = 900 \tag{12.5}$$

Using linked lists based on the structure defined above, please implement the following four functions.

1. `FACTOR *factorize(int N);`
   This function factorizes the input N into its prime factors and their associated powers, and returns a linked list that contains all these prime factors.

2. `FACTOR *GCD(FACTOR *A, FACTOR *B);`
   This function takes two linked lists of prime factors as input, and finds the Greatest Common Divisor of these two inputs. Note that it returns a linked list of prime factors.

3. `FACTOR *LCM(FACTOR *A, FACTOR *B);`
   This function takes two linked lists of prime factors as input, and finds the Least Common Multiple of these two inputs. Note that it also returns a linked list of prime factors.

4. `void write(FACTOR *A);`
   This function prints out all the prime factors and their associated powers. In addition, it recalculates the product of all the factors and prints it out at the end.

Your program should take two command line arguments as the inputs to your program, find their prime factors, their GCD, LCM, and then print them out. Examples of program execution are shown below.

```
$ ./a.out 100 225
A = 2^2 * 5^2 = 100
B = 3^2 * 5^2 = 225
GCD = 5^2 = 25
LCM = 2^2 * 3^2 * 5^2 = 900
$ ./a.out 24 35
A = 2^3 * 3 = 24
B = 5 * 7 = 35
GCD = 1 = 1
LCM = 2^3 * 3 * 5 * 7 = 840
```

Notes.

1. Create a directory **lab12** and use it as the working directory.
2. Name your program source file as **lab12.c**.
3. The first few lines of your program should be comments as the following.

   ```
   /* EE231002 Lab12. GCD and LCM.
      ID, Name
      Date:
   */
   ```

4. After you finish verifying your program, you can submit your source code by

   ```
   $ ~ee2310/bin/submit lab12 lab12.c
   ```

   If you see a "submitted successfully" message, then you are done. In case you want to check which file and at what time you submitted your labs, you can type in the following command:

   ```
   $ ~ee2310/bin/subrec lab12
   ```

   It will show the submission records of lab12.

5. You should try to write the program as efficient as possible. The format of your program should be compact and easy to understand. These are part of the grading criteria.