lab11

```
$ gcc lab11.c

$ 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
```

---

score: 96.0
o. [Output] Program output is correct, good.
o. [Format] Program format can be improved.
o. [Program] has memory leak problem.

lab11.c

```c
// EE231002 Lab11. Linked Lists
// 110060007, 黃俊穎
// 2021/12/20

#include <stdio.h>
#include <stdlib.h>

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;

// factorize num, N = num
FACTOR *factorize(int N);
// set GCD link, input *A and *B links
FACTOR *GCD(FACTOR *A, FACTOR *B);
// set LCM link, input *A and *B links
FACTOR *LCM(FACTOR *A, FACTOR *B);
// print out prime factors and powers, input *A link
void write(FACTOR *A);

int main(int argc, char *argv[])
{
    int num1, num2;               // input numbers
    FACTOR *A, *B;                // save links

    num1 = atoi(argv[1]);
    num2 = atoi(argv[2]);         // convert input numbers to num1 & num2

    // print out factorized results
    printf("A = ");
    A = factorize(num1);
    write(A);
    printf("B = ");
    B = factorize(num2);
    write(B);

    // print out GCD
    printf("GCD = ");
```

```
41      write(GCD(A, B));
42
43      // print out LCM
44      printf("LCM = ");
45      write(LCM(A, B));
46
47      return 0;
48  }
49
50  // factorize num, N = num
51  FACTOR *factorize(int N)
52  {
53      int i, power; // i for prime in loop detecting, power for power number
54      FACTOR *first = NULL, *temp, *new_factor;   // linked list
55
56      // factorize num by each prime, until num = 1
57      // increment power each time if num is divided by the same prime
58      for (i = 2; N != 1; i++) {
59          power = 0;
60          while (N % i == 0) {
61              N /= i;
62              power++;
63          }
64          // set linked list
65          if (power != 0) {
66              new_factor = malloc(sizeof(FACTOR));
67              new_factor->prime = i;
68              new_factor->power = power;
69              new_factor->next = NULL;
70              if (first == NULL)
71                  first = new_factor;
72              else
73                  temp->next = new_factor;
74              temp = new_factor;
75          }
76      }
77      return first;
78  }
79
80  // set GCD link, input *A and *B links
81  FACTOR *GCD(FACTOR *A, FACTOR *B)
```

```
82 {
83     // linked list pointers declaration
84     FACTOR *first = NULL, *temp, *new_factor = NULL;
85 
86     // search for the common prime num, exit until a or b end
87     while (A != NULL && B != NULL) {
88         if (A->prime < B->prime)
89             A = A->next;           // shift A
90         else if (A->prime > B->prime)
91             B = B->next;           // shift B
92         else {
93             // when find out common prime num, store prime num
94             new_factor = malloc(sizeof(FACTOR));
95             new_factor->prime = A->prime;
96 
97             // store smaller power
98             if (A->power >= B->power)
99                 new_factor->power = B->power;
100            else
101                new_factor->power = A->power;
102
103            // connect nodes
104            new_factor->next = NULL;
105            if (first == NULL)
106                first = new_factor;
107            else
108                temp->next = new_factor;
109            temp = new_factor;
110
111            // shift A and B
112            A = A->next;
113            B = B->next;
114        }
115    }
116
117    // if 2 numbers relatively prime, return 1 * 1, else return linked list
118    if (new_factor == NULL) {
119        new_factor = malloc(sizeof(FACTOR));
120        new_factor->prime = 1;
121        new_factor->power = 1;
122        new_factor->next = NULL;
```

```
123         return new_factor;
124     }
125     else
126         return first;
127 }
128
129 // set LCM link. input *A and *B links
130 FACTOR *LCM(FACTOR *A, FACTOR *B)
131 {
132     // linked list pointers declaration
133     FACTOR *first = NULL, *temp, *new_factor;
134
135     // store all prime num, exit until a or b ended
136     while (A != NULL && B != NULL) {
137         // if A and B have different prime num, store it.
138         // or store bigger power one
139         if (A->prime < B->prime) {
140             new_factor = malloc(sizeof(FACTOR));
141             new_factor->prime = A->prime;
142             new_factor->power = A->power;
143             new_factor->next = NULL;
144             if (first == NULL)
145                 first = new_factor;
146             else
147                 temp->next = new_factor;
148             temp = new_factor;
149             A = A->next;            // shift A
150         } else if (A->prime > B->prime) {
151             new_factor = malloc(sizeof(FACTOR));
152             new_factor->prime = B->prime;
153             new_factor->power = B->power;
154             new_factor->next = NULL;
155             if (first == NULL)
156                 first = new_factor;
157             else
158                 temp->next = new_factor;
159             temp = new_factor;
160             B = B->next;            // shift B
161         } else {
162             // store larger power when same prime
163             new_factor = malloc(sizeof(FACTOR));
```

```
164             new_factor->prime = A->prime;
165
166             if (A->power >= B->power)
167                 new_factor->power = A->power;
168             else
169                 new_factor->power = B->power;
170
171             // connect nodes
172             new_factor->next = NULL;
173             if (first == NULL)
174                 first = new_factor;
175             else
176                 temp->next = new_factor;
177             temp = new_factor;
178
179             // shift A and B
180             A = A->next;
181             B = B->next;
182         }
183     }
184     // A or B has come to NULL, need to finish linking the remaining primes
185     if (A == NULL && B != NULL) { // B is the one remaining
186         while (B != NULL) {
187             new_factor = malloc(sizeof(FACTOR));
188             new_factor->prime = B->prime;
189             new_factor->power = B->power;
190             new_factor->next = NULL;
191             temp->next = new_factor;
192             temp = new_factor;
193             B = B->next;            // shift B
194         }
195     } else { // A is the one remaining
196         while (A != NULL){
        while (A != NULL) {
197             new_factor = malloc(sizeof(FACTOR));
198             new_factor->prime = A->prime;
199             new_factor->power = A->power;
200             new_factor->next = NULL;
201             temp->next = new_factor;
202             temp = new_factor;
203             A = A->next;            // shift A
```

```
204         }
205     }
206     // return linked list
207     return first;
208 }
209
210 // print out prime factors and powers, input *A
211 void write(FACTOR *A)
212 {
213     int product = 1, power;             // initialize variables
214
215     // print out linked list
216     for (; A != NULL; A = A->next) {
217         // if power = 1, don't print power, otherwise, print it out
218         if (A->power == 1)
219             printf("%d", A->prime);
220         else
221             printf("%d^%d", A->prime, A->power);
222
223         // if isn't last prime, print '*'
224         if (A->next != NULL)
225             printf(" * ");
226
227         // save power product
228         power = A->power;
229
230         // calculate total product
231         for (; power != 0; power--) {
232             product *= A->prime;
233         }
234     }
235     // print product
236     printf(" = %d\n", product);
237 }
```