lab12

```c
// EE231002, Lab12 Linked Lists
// 108061213, 劉奕緯
// Dec. 10, 2019

#include <stdio.h>          // use printf
#include <stdlib.h>         // use malloc, free, atoi
#include <math.h>           // use pow function

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

// factorize N into its prime factors and return a linked list
// that contains all these prime factors.
FACTOR *factorize(int N);
// 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.
FACTOR *GCD(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.
FACTOR *LCM(FACTOR *A, FACTOR *B);
// This function prints out all the prime factors and their associated powers.
// In addition, it recalculates the product of all the factors
// and prints out at the end.
void write(FACTOR *A);
// free a linked list
void FreeList(FACTOR *head);

int main (int arg, char **argv)
int main(int arg, char **argv)
{
    // test
    FACTOR *a, *b, *c;              // pointer to each linked list
Need a blank line here.
    printf("A = ");                 // output A's factorization
    a = factorize(atoi(argv[1]));
    write(a);
    printf("B = ");                 // output B's factorization
    b = factorize(atoi(argv[2]));
    write(b);
    printf("GCD = ");
    c = GCD(a, b);                  // perform GCD(A,B)
    write(c);                       // and output the result
    free(c);                        // free the space GCD(A,B) taken
```

```
47      printf("LCM = ");
48      write(LCM(a, b));               // perform LCM(A,B) and output the result
49      return 0;
50  }
    Need a blank line here.
51  // factorize N into its prime factors and return a linked list
52  // that contains all these prime factors.
53  FACTOR *factorize(int N)
54  {
55      int i, j;                       // loop index, i for factor, j for power
56      FACTOR *head = NULL;            // pointer to the first linkde list node
57      FACTOR *cur, *pre;              // current node, previous node
58
59      for (i = 2;i <= N;i++) {        // seaching factor
        for (i = 2; i <= N; i++) {        // seaching factor
60          if (N % i == 0) {           // if i is factor
61              for (j = 0; (N % i) == 0; j++) N /= i;
62                                      // then j is corrosponding power
63              cur = malloc(sizeof(FACTOR));
64                                      // as a space for new node and cur
65                                      // point to the memeory
66              cur->power = j;         // store into value into node
67              cur->prime = i;
68              cur->next = NULL;       // this is latest node, i store it in the
69                                      // end of linklist.
70              if (head == NULL) head = pre = cur;
71              else pre->next = cur;
72                                      // previous node->next point to cur node
73              pre = cur;              // current is previous now.
74                                      // Note that the following method to create
75                                      // a linked list are by the same means.
76          }
77      }
78      return head;
79  }
    Need a blank line here.
80  // This function prints out all the prime factors and their associated powers.
81  // In addition, it recalculates the product of all the factors
82  // and prints out at the end.
83  void write(FACTOR *A)
84  {
85      FACTOR *cur;                    // current node
86      int product = 1;                // totoal product
87
88      cur = A;                        // cur is the first node
89      if (cur->power == 1) printf("%d ",cur->prime);
        if (cur->power == 1) printf("%d ", cur->prime);
90                                      // we dont print "^1".
91      else printf("%d^%d ", cur->prime, cur->power);
92      product *= pow(cur->prime, cur->power);
```

```
93                                  // caculate the product
94      cur = cur->next;            // next node
95      while (cur != NULL) {       // print out all list by same means above
96          if (cur->power == 1) printf("* %d ",cur->prime);
            if (cur->power == 1) printf("* %d ", cur->prime);
97          else printf("* %d^%d ", cur->prime, cur->power);
98          product *= pow(cur->prime, cur->power);
99          cur = cur->next;
100     }
101     printf("= %d\n", product);
102 }
    Need a blank line here.
103 // This function takes two linked lists of prime factors as input, and
104 // finds the Greatest Common Divisor of these two inputs.
105 // Note that it returns a linked list of prime factors.
106 FACTOR *GCD(FACTOR *A, FACTOR *B)
107 {
108     FACTOR *head = NULL;        // pointer to first node
109     FACTOR *pre, *cur;          // pointer to previous and current node
110
111     while (A != NULL && B != NULL) {
112                                 // if they are not equal, Make them equal
113         if (A->prime > B->prime) B = B->next;
114         else if (A->prime < B->prime) A = A->next;
115         else {                  // if they equal store the prime and the
116                                 // smaller power.
117             cur = malloc(sizeof(FACTOR));
118             cur->next = NULL;
119             cur->prime = A->prime;
120             if(A->power > B->power) cur->power = B->power;
                if (A->power > B->power) cur->power = B->power;
121             else cur->power = A->power;
122             if (head == NULL) head = pre = cur;
123             else pre->next = cur;
124             pre = cur;
125             A = A->next;        // next A's
126             B = B->next;        // next B's
127         }
128     }                           // end till a list has no common factor
129     if (head == NULL) {         // if they have no common factor
130                                 // then there greatest common factor is 1
131         head = malloc(sizeof(FACTOR));
132         head->next = NULL;
133         head->prime = head->power = 1;
134     }
        This is not needed.
135     return head;
136 }
    Need a blank line here.
137 // This function takes two linked lists of prime factors as input,
```

```
138 // and finds the Least Common Multiple of these two inputs.
139 // Note that it also returns a linked list of prime factors.
140 FACTOR *LCM(FACTOR *A, FACTOR *B)
141 {
142     FACTOR *head = NULL;            // pointer to first node
143     FACTOR *cur, *pre;              // pointer to current and previous nodes
144
145     while (A != NULL && B != NULL) {
146         if (A->prime < B->prime) {  // store the smaller factor and its power
147             cur = malloc(sizeof(FACTOR));
148             cur->next = NULL;
149             cur->prime = A->prime;
150             cur->power = A->power;
151             if (head == NULL) head = pre = cur;
152             else pre->next = cur;
153             A = A->next;            // next factor
154             pre = cur;
155         }
156         else if (A->prime > B->prime) {
157                                     // store the smaller factor and its power
158             cur = malloc(sizeof(FACTOR));
159             cur->next = NULL;
160             cur->prime = B->prime;
161             cur->power = B->power;
162             if (head == NULL) head = pre = cur;
163             else pre->next = cur;
164             pre = cur;
165             B = B->next;
166         }
167         else {                      // if they have common factor then store
168                                     // then store the larger power.
169             cur = malloc(sizeof(FACTOR));
170             cur->next = NULL;
171             cur->prime = B->prime;
172             if(A->power > B->power) cur->power = A->power;
                if (A->power > B->power) cur->power = A->power;
173             else cur->power = B->power;
174             if (head == NULL) head = pre = cur;
175             else pre->next = cur;
176             pre = cur;
177             A = A->next;            // next A
178             B = B->next;            // next B
179         }
180     }
181     // if we got all factor of one number
182     // then get all remain factor of the other number
183     while (B != NULL) {
184         cur = malloc(sizeof(FACTOR));
185         cur->next = NULL;
186         cur->prime = B->prime;
```

```
187         cur->power = B->power;
188         pre->next = cur;
189             pre = cur;
            pre = cur;
190         B = B->next;
191     }
192     while (A != NULL) {
193         cur = malloc(sizeof(FACTOR));
194         cur->next = NULL;
195         cur->prime = A->prime;
196         cur->power = A->power;
197         pre->next = cur;
198             pre = cur;
            pre = cur;
199         A = A->next;
200     }
201     return head;
202 }
    Need a blank line here.
203 // delet linked list
204 void FreeList(FACTOR *head)
205 {
206     FACTOR *cur, *next;             // current node, next node
207
208     while (cur != NULL) {
               cur is not initialized
209         next = cur->next;           // store the address to the next node
210         free(cur);                  // free up current node
211         cur = next;                 // go to next node
212     }
213 }
```

[Format] can be improved.
[Coding] lab12.c spelling errors: caculate(1), corrosponding(1), delet(1), dont(1), linkde(1), linklist(1), memeory(1), seaching(1), totoal(1)

Score: 76