```
//最大公约数
int gcd(int a, int b) {
     if (b == 0)return a;
     else {
          return gcd(b, a%b);
     }
}
//最小公倍数
int min = gcd(a, b);
int lcm = a / min * b;
//分数的四则混合运算
1）分数的表示
struct Fraction {
     int up, down;
};
2）分数的化简
Fraction reductuion(Fraction result) {
     if (result.up < 0) {
          result.up = -result.up;
          result.down = -result.down;
     }
     if (result.up == 0) result.down = 1;
     else {
          int d = gcd(result.up, result.down);
          result.up /= d;
          result.down /= d;
     }
     return result;
}
3）加法
Fraction add(Fraction f1, Fraction f2) {
     Fraction result;
     result.up = f1.up*f2.down + f1.down*f2.up;
     result.down = f1.down*f2.down;
     return reduction(result);
}
4）减法
Fraction sub(Fraction f1, Fraction f2) {
     Fraction result;
     result.up = f1.up*f2.down - f2.up*f1.down;
     result.down = f1.down*f2.down;
     return reduction(result);
}
5）乘法
Fraction mult(Fraction f1, Fraction f2) {
     Fraction result;
     result.up = f1.up*f2.up;
```

```
     result.down = f1.down*f2.down;
     return reduction(result);
}
```

6）除法

```
Fraction divi(Fraction f1, Fraction f2) {
     Fraction result;
     result.up = f1.up*f2.down;
     result.down = f1.down*f2.up;
     return reduction(result);
}
```

7）分数输出

```
void show(Fraction r) {
     r = reduction(r);
     if (r.down == 1) {
          printf("%lld", r.up);
     }
     if (abs(r.up)>r.down) {
          printf("%d%d/%d", r.up / r.down, abs(r.up) % r.down, r.down);
     }
     else {
          printf("%d/%d", r.up, r.down);
     }
}
```

//素数

1）素数判断（优化与未优化）

未优化

```
bool isPrime(int n {
     if (n <= 1) return false;
     for (int i = 2; i < n; i++) {
          if (n%i == 0) return false;
     }
     return true;
}
```

优化后

```
bool isPrime(int n) {
     if (n <= 1)return false;
     int sqr = (int)sqrt(1.0*n);
     for (int i = 2; i < sqr; i++) {
          if (n%i == 0) return false;
     }
     return true;
}

bool isPrime(int n) {
     if (n <= 1) return false;
```

```
     for (long long i = 2; i*i < n; i++) {
          if (n%i == 0)return false;
     }
     return true;
}
```

2）素数表的打印（优化与未优化）

未优化

```
const int MAX = 101;
int Prime[MAX], Pnum = 0;
bool p[MAX] = { 0 };
void Find_Prime() {
     for (int i = 1; i < MAX; i++) {//这里从 1 开始
          if (isPrime(i)) {
               Prime[Pnum++] = i;
               p[i] = true;
          }
     }
}
```

优化

```
const int MAX = 101;
int Prime[MAX], Pnum = 0;
bool p[MAX] = { 0 };
void Find_Prime() {
     for (int i = 2; i < MAX; i++) {
          if (p[i] == false) {
               Prime[Pnum++] = i;
               for (int j = i + i; j < MAX; j += i) {
                    p[j] = true;
               }
          }
     }
}
```

```
//质因子分解
struct fator{
     int x;
     int cnt;
}fac[10];
int num = 0;
```

1)分解求解

```
Find_Prime();
if (n == 1) printf("1=1");
else{
     int sqr = (int)sqrt(1.0*n);
     for (int i = 0; i < Pnum&&prime[i] < sqr; i++) {
```

```
        if (n%prime[i] == 0) {
            fac[num].x = prime[i];
            fac[num].cnt = 0;
            while (n%prime[i] == 0) {
                fac[num].cnt++;
                n /= prime[i];
            }
            num++;
        }
        if (n == 1) break;
    }
    if (n != 1) {
        fac[num].x = n;
        fac[num].cnt = 1;
    }
}
```

2)输出

```
void show(int x) {
    for (int i = 0; i < num; i++) {
        printf("%d^%d", fac[i].x, fac[i].cnt);
        if (i!=num) {
            printf("*");
        }
    }
}
```

3)N 含有多少个因子个数

```
Find_Prime();
if (n == 1)printf("1=1");
else {
    int sqr = (int)sqrt(1.0*n);
    for (int i = 0; i < Pnum&&prime[i] < sqr; i++) {
        if (n%prime[i] == 0) {
            fac[num].x = prime[i];
            fac[num].cnt = 0;
            while (n%prime[i] == 0) {
                fac[num].cnt++;
                n /= prime[i];
            }
            num++;
        }
        if (n == 1)braek;
    }
    if (n != 1) {
        fac[num].x = n;
        fac[num].cnt = 1;
    }
```

```
}


//大整数运算
struct bign {
     int d[1000];
     int len;
     bign() {
          memset(d, 0, sizeof(d));
          len = 0;
     }
};
```
1）大整数的存储
```
bign change(char str[]) {
     bign a;
     a.len = strlen(str);
     for (int i = 0; i < a.len; i++) {
          a.d[i] = str[a.len - 1 - i] - '0';
     }
     return a;
}
```
2）大整数的四则运算

加法
```
bign add(bign a, bign b) {
     bign c;
     int carry = 0;
     for (int i = 0; i < a.len || i < b.len; i++) {
          int temp = a.d[i] + b.d[i] + carry;
          c.d[c.len++] = temp % 10;//注意这个 len 应该单独加的
          carry = temp / 10;
     }
     if (carry != 0) {
          c.d[c.len++] = carry;
     }
     return c;
}
```
减法
```
bign sub(bign a, bign b) {
     bign c;
     for (int i = 0; i < a.len || i < b.len; i++) {
          if (a.d[i] < b.d[i]) {
               a.d[i + 1]--;
               a.d[i] += 10;
          }
          c.d[c.len++] = a.d[i] - b.d[i];
     }
     while (c.len-1>= 1 && c.d[c.len - 1] == 0) {
```

```
        c.len--;
    }
    return c;
}
```

乘法

```
bign mult(bign a, bign b) {
    bign c;
    int carry = 0;
    for (int i = 0; i < a.len || a < b.len; i++) {
        int temp = a.d[i] * b + carry;
        c.d[c.len++] = temp % 10;
        carry = temp / 10;
    }
    while (carry != 0) {
        c.d[c.len++] = carry % 10;
        carry /= 10;
    }
    return c;
}
```

除法

```
bign divide(bign a, bign b,int& r) {
    bign c;
    c.len = a.len;
    for (int i = a.len-1; i >=0; i--) {
        r = r * 10 + a.d[i];
        if (r < b) c.d[i] = 0;
        else {
            c.d[i] = r / b;
            r = r % b;
        }
    }
    while (c.len - 1 >= 1 && c.d[c.len - 1] == 0) {
        c.len--;
    }
    return c;
}
```

3)输出

```
void show(bign x) {
    for (int i = x.len - 1; i >= 0; i--) {
        printf("%d", x.d[i]);
    }
}
```

4)比较

```
int compare(bign a, bign b) {
    if (a.len > b.len)return 1;
    else if (a.len < b.len) return -1;
```

```
     else {
          for (int i = a.len-1; i >= 0; i--) {
               if (a.d[i] > b.d[i])return 1;
               else if (a.d[i] < b.d[i])return -1;
          }
          return 0;
     }
}
//n!的质因子个数
1)优化
int number(int n,int p) {
     int ans = 0;
     while (n) {
          ans += n / p;
          n /= p;
     }
     return ans;
}
2）未优化
int number(int x,int p) {
     int ans = 0;
     for (int i = 2; i <= x; i++) {
               int temp = i;
               while (temp%p == 0) {
                    ans++;
                    temp /= p;
               }
     }
     return ans;
}
```