GCD & LCM

多位的gcd，采用前2个与第3个求gcd得。
LCM算法基于gcd

```
lcm(a, b) = a * b / gcd(a, b)
```

全递归精简版

```
// import `swap`
#include <algorithm>
using namespace std;

long long gcd(long long a, long long b){
  if (a < b) swap(a, b);
  return b == 0 ? a : gcd(b, a % b);
}

long long lcm(long long a, long long b) {
  return a * b / gcd(a, b);
}

long long ngcd(long long *a, long long n) {
  return n == 1 ? a[0] : gcd(a[n-1], ngcd(a, n-1));
}

long long nlcm(long long *a, long long n) {
  return n == 1 ? a[0] : lcm(a[n-1], nlcm(a, n-1));
}
```

ngcd/nlcm非递归版

```
// import `swap`
#include <algorithm>
using namespace std;

long long gcd(long long a, long long b){
  if (a < b) swap(a, b);
  return b == 0 ? a : gcd(b, a % b);
}

long long lcm(long long a, long long b) {
  return a * b / gcd(a, b);
}

long long ngcd(long long *a, long long n) {
  if(n == 1) return a[0];
  long long r = gcd(a[0], a[1]);
  for(int i=2;i<n;i+=1){
    r = gcd(r, a[i]);
  }
  return r;
}

long long nlcm(long long *a, long long n) {
  if(n == 1) return a[0];
  long long r = lcm(a[0], a[1]);
  for(int i=2;i<n;i+=1){
    r = lcm(r, a[i]);
  }
  return r;
}
```