최대공약수, 최소공배수

GCD(Greatest Common Divisor)

- 최대공약수를 구할 때 유클리드 알고리즘을 이용한다.
- a를 b로 나눈 나머지를 r이라고 할 때, GCD(a, b) == GCD(b, r) 의 관계가 성립된다.
  즉, a%b == r 이므로 GCD(a, b) == GCD(b, a%b) 의 관계이다.
  이 때, r == 0이 되면 그 때의 b가 최대 공약수가 된다.
  ex) GCD(24, 16) = GCD(16, 24%16) = GCD(16, 8)
  GCD(16, 8) = GCD(8, 16%8) = GCD(8, 0) 이 때 8이 최대공약수가 된다.
- GCD는 재귀함수를 이용하면 간단히 구현할 수 있다.

```
int gcd(int a, int b) {
        if (b == 0) {
                return a;
        }
        else {
                return gcd(b, a%b);
        }
}
```

```
int gcd(int a, int b) {
        while (b != 0) {
                int r = a%b;
                a = b;
                b = r;
        }
        return a;
}
```

최대공약수, 최소공배수

GCD(Greatest Common Divisor)

- GCD(a, b)에서 a, b 중 어느 숫자가 큰 숫자가 와도 상관이 없다.

  ① a 〉 b

    GCD(24, 16) = GCD(16, 24%16) = GCD(16, 8)

    GCD(16, 8) = GCD(8, 16%8) = GCD(8, 0)

  ② a 〈 b

    GCD(16, 24) = GCD(24, 16) = 이하 동일

- 3개 이상의 최대공약수는 GCD 결과를 계속 사용하여 구할 수 있다.

  ex) GCD(a, GCD(b, c))

최대공약수, 최소공배수

LCM(Least Common Mutiple)

- 최소 공배수는 두 수 a, b의 최대공약수를 g라고 하였을 때,
  LCM = g * (a/g) * (b/g) = a * b / g

```
int gcd(int a, int b){
    if(b == 0){
        return a;
    }
    else{
        return gcd(b, a%b);
    }
}

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

https://www.acmicpc.net/problem/1934