Contents

1 Util

1.1 GCD

GCD by Euclid's algorithm.
Requires `math`.

```
//If you need it for long values, remember to change abs to labs
int gcd(int x, int y){return y ? gcd(y , x % y) : abs(x);}
```

1.2 LCM

Requires 1.1 `gcd`.

```
typedef long long ll;
ll lcm(int x, int y){
    if(x && y) return abs(x) / gcd(x,y) * ll(y);
}
```

1.3 Bezout's identity

Find some a and b such that $ax + by = gcd(x, y)$
Requires `utility`

```
typedef pair<int, int> bezout;

bezout find_bezout(int x, int y){
    if(y == 0) return bezout(1,0);
    bezout u = find_bezout(y, x % y);
    return bezout(u.second, u.first - (x/y) * u.second);
}
```

1.4 Digit sum

Find the sum of all decimal digits present in some interval $a < b < 10^9$.

```
#include <cstdio>

int c[2][10], pow10[10];
void count(int x, int *cnt){
    int d, dcnt = 0, r = 0, rem0 = 0, v;
    while(x){
        d = x % 10; x /= 10;
        if(dcnt){
        v = d * pow10[dcnt - 1] * dcnt;
        for(int i = 0; i < 10; ++i) cnt[i] += v;
            if(!d) rem0 += (pow10[dcnt] - 1) - r;
    }
    v = pow10[dcnt];
    for(int i = 1; i < d; ++i) cnt[i] += v;
        if(d) cnt[d] += r + 1;
        r = pow10[dcnt++] * d + r;
    }
    cnt[0] -= rem0;
}

int main(void){
    pow10[0] = 1;
    for(int i = 1; i < 10; ++i)
        pow10[i] = 10 * pow10[i - 1];
    for(int a, b; scanf("%d %d", &a, &b) == 2 && a && b;){
        long long ans = 0;
        for(int i = 0; i < 10; ++i)
            c[0][i] = c[1][i] = 0;
        count(b, c[1]);
        count(a - 1, c[0]);
        for(int i = 1; i < 10; ++i)
            ans += (long long)i * (c[1][i] - c[0][i]);
        printf("%lld\n", ans);
    }
    return 0;
}
```

1.5 Musical chairs

N children are seated on N chairs arranged around a circle. The chairs are numbered from 1 to N . Your program pre-selects a positive number D . The program starts going in circles counting the children starting with the first chair. Once the count reaches D , that child leaves the game, removing his/her chair. The program starts counting again, beginning with the next chair in the circle. The last child remaining in the circle is the winner. Given N and D determine the winner.

```
// :3

#include <cstdio>

```

```
int dp[1048576];
int main(void){
    dp[1] = 0;
    for(int d, i, n; scanf("%d %d", &n, &d) == 2 && (n || d); ){
        for(i = 2; i <= n; ++i) dp[i] = (dp[i - 1] + d) % i;
            printf("%d %d %d\n", n, d, dp[n] + 1);
    }
    return 0;
}
```

2 Primality

2.1 Primality testing : simple

Requires math.

```
bool is_prime(int n){
    if(n < 0) return is_prime(-n);
    if(n < 5 || n % 2 == 0 || n % 3 == 0) return (n == 2 || n == 3);
    int maxP = sqrt(n) + 2;
    for(int p = 5; p < maxP; p += 6)
        if(n % p == 0 || n % (p+2) == 0 ) return false;
    return true;
}
```

2.2 Prime factors

Squeeze the prime factors out of n.
Requires math.

```
typedef map<int,int> prime_map;

void squeeze(prime_map &M, int &n, int p) {for(; n % p == 0; n /= p)M[p]++;}
prime_map factor(int n){
    prime_map M;
    if(n < 0)
        return factor(-n);
    if(n < 2)
        return M;
    squeeze(M, n, 2); squeeze(M, n, 3);
    int maxP = sqrt(n) + 2;
    for(int p = 5; p < maxP; p += 6){
        squeeze(M, n, p); squeeze(M, n, p+2);
    }
    if(n > 1)M[n]++;
    return M;
}
```

3 Congruences

3.1 Linear congruence

Find the lowest non-negative solution to $ax \equiv b \bmod m$
Requires 1.3 `find_bezout`

```
/*
 *    Find the lowest non-negative solution to a*x = b(mod m)
 *    Return -1 if the congruence is not possible.
 */
int mod(int x, int m){return x % m + (x < 0) ? m : 0;}

int solve_mod(int a, int b, int m){
    if(m < 0) return solve_mod(a, b, -m);
    if(a < 0 || a >= m || b < 0 || b >= m)
        return solve_mod(mod(a, m), mod(b, m), m);
    bezout t = find_bezout(a, m);
    int d = t.first * a + t.second * m;
    if(b % d) return -1;
    else return mod(t.first * (b / d), m);
}
```