【没选修实验课的同学专用的编程题】--Miller-Rabin

用C/C++实现Miller-Rabin素数测试算法。请提交固定链接的MarkDown页面。选修实验课的同学也可以考虑实现，不过就不强制要求。

```
// https://github.com/gou4shi1/oj/blob/master/poj1811.cpp
#include <algorithm>
#include <cstdio>
#include <cstring>
#include <iostream>
#include <time.h>

using namespace std;
typedef long long ll;

const int INF = 0x7f7f7f7f;
const int maxn = 100 + 10;

// ret = a * b ( mod n )
ll mul_mod ( ll a, ll b, ll n ) {
    a %= n;
    b %= n;
    ll ret = 0;
    ll tmp = a;
    while ( b ) {
        if ( b & 1 ) {
            ret += tmp;
            if ( ret > n )
                ret -= n;
        }
        tmp <<= 1;
        if ( tmp > n )
            tmp -= n;
        b >>= 1;
    }
    return ret;
}

// ret = a^n ( mod MOD )
ll pow_mod ( ll a, ll n, ll mod ) {
    ll ret = 1;
    ll tmp = a % mod;
    while ( n ) {
        if ( n & 1 )
            ret = mul_mod ( ret, tmp, mod );
        tmp = mul_mod ( tmp, tmp, mod );
        n >>= 1;
    }
    return ret;
}
```

```
/*
 * witness whether a^(n-1) = 1 ( mod n )
 * n-1 = x*2^t, 中间判断
 */
bool witness ( ll a, ll n, ll x, ll t ) {
    ll ret = pow_mod ( a, x, n );
    ll last = ret;
    for ( int i = 1; i <= t; ++i ) {
        ret = mul_mod ( ret, ret, n );
        if ( ret == 1 && last != 1 && last != n - 1 )
            return true;
        last = ret;
    }
    if ( ret != 1 )
        return true;
    else
        return false;
}

/*
 * 合数可能被判成素数, 概率(1/4)^NUM
 * 素数一定返回true
 * n-1 = x*2^t, 中间判断
 */
bool Miller_Rabin ( ll n ) {
    if ( n < 2 )
        return false;
    if ( n == 2 )
        return true;
    if ( !( n & 1 ) )
        return false;

    ll x = n - 1, t = 0;
    while ( !( x & 1 ) )
        x >>= 1, ++t;

    srand ( time ( NULL ) );

    const static int NUM = 8;
    for ( int i = 0; i < NUM; ++i ) {
        ll a = rand () % ( n - 1 ) + 1;
        if ( witness ( a, n, x, t ) )
            return false;
    }
    return true;
}

ll factor[ maxn ]; //素因数分解结果
int tot;           //素因子个数
// gcd要保证a>b否则死循环
ll gcd ( ll a, ll b ) {
    ll t;
```

```
    while ( b ) {
        t = a;
        a = b;
        b = t % b;
    }
    return a >= 0 ? a : -a;
}
/*
 * 分解x, c是常数参数?算导上是-1
 * 生日悖论来提高概率
 * 可能返回n本身
 */
ll polard_rho ( ll n, ll c ) {
    ll i = 1, k = 2;
    srand ( time ( NULL ) );
    ll x = rand () % ( n - 1 ) + 1;
    ll y = x;
    while ( 1 ) {
        ++i;
        // f(x)=(x^2+a)%n  生成伪随机数,最终会进入循环
        x = ( mul_mod ( x, x, n ) + c ) % n;
        ll d = gcd ( y - x, n );
        if ( d != 1 && d != n )
            return d;
        if ( y == x )
            return n;
        if ( i == k )
            y = x, k += k;
    }
}

void findfac ( ll n ) {
    if ( n == 1 )
        return;
    if ( Miller_Rabin ( n ) ) {
        factor[ tot++ ] = n;
        return;
    }

    const static int k = 107;
    ll p = n;
    int c = k;
    while ( p >= n )
        p = polard_rho ( p, c-- );
    findfac ( p );
    findfac ( n / p );
}

/*
 * poj1811
 * 输入n(2<n<2^54),如果是素数,输出prime
 * 否则输出最小的素因子
 */
```

```cpp
int main () {
#ifdef LOCAL
    freopen ( "in", "r", stdin );
    // freopen("out","w",stdout);
#endif
    int T;
    scanf ( "%d", &T );
    while ( T-- ) {
        ll n;
        scanf ( "%lld", &n );
        if ( Miller_Rabin ( n ) )
            printf ( "Prime\n" );
        else {
            tot = 0;
            findfac ( n ); // factor其实就是所有的因子了
            ll ans = factor[ 0 ];
            for ( int i = 1; i < tot; ++i )
                ans = min ( ans, factor[ i ] );
            printf ( "%lld\n", ans );
        }
    }

    return 0;
}
```