（1）试除法求一个数的所有约数

```
//返回的是数组，里面是这个数的所有约数
vector<int> get_divisors(int n)
{
    vector<int> res;
    for(int i = 1; i <= n / i; i++){
        if(n % i == 0){
            res.push_back(i);
            if(i != n / i)  res.push_back(n / i);
        }
    }
    sort(res.begin(), res.end());
    return res;
}
```

（2）求一个数的约数个数

基于算术基本定理

> N = (p1^x1)(p2^x2)(p3^x3)…(pk^xk)
> 约数个数=(x1+1)(x2+1)(x3+1)…(xk+1)

为什么呢？简单证明如下

因为每一种pi都有0->xi种选法，一共xi+1种，一共k个所以迭代k次

这么讲不够直白，接下来举个栗子

> 24 = 2 * **2 * 2 * 3 = 2³ *** 3
> 再用各个质数的指数加一后再相乘即为此数的约数个数，
> 比如 (3+1)**(1+1) = 4 *** 2 = 8， 即表示24有8个约数。
> 24的约数:1、2、3、4、6、8、12、24。

```
思路就是先把原数分解为质因数，最后把每一个数的指数累加即可。
从a1一直分解到an，由于a的数据过大，此处用哈希表进行存储

void dividprime(int n)
```

```
{
    for(int i = 2; i <= n / i; i++){
        while(n % i == 0){
            m[i]++;
            n /= i;
        }
    }
    if(n > 1)   m[n]++;
}

```

（3）约数之和

> 基本思想：
> 如果 N = p1^c1 * p2^c2 *…* pk^ckN
> 约数个数：(c1+1)*(c2+1)*…*(ck+1)
> 约数之和： (p1^0+p1^1+…+p1^ck)*…*(pk^0+pk^1+…+pk^ck)

```
while (b -- ) t = (t * a + 1) % mod;

t = t*p + 1
t = 1
t = p + 1
t = p^2 + p + 1
……
t = p^b + p^b-1 +…+ 1
```

```
#include<iostream>
#include<map>
using namespace std;
typedef long long LL;
const int MOD = 1e9 + 7;

map<int,int> m;

void dividprime(int n)
```

```
{
    for(int i = 2; i <= n / i; i++){
        if(n % i == 0){
            while(n % i == 0){
                m[i]++;
                n /= i;
            }
        }
    }
    if(n > 1)   m[n]++;
}
int main()
{
    int n;
    cin >> n;
    while(n--){
        int x;
        cin >> x;
        dividprime(x);
    }
    LL ans = 1;
    for(auto pos : m){
        int p = pos.first, a = pos.second;
        LL res = 1;
        while(a--) res = (res * p + 1) % MOD;
        ans = ans * res % MOD;
    }
    cout << ans << endl;

    return 0;
}
```