1748. The Most Complex Number

Time limit: 1.0 second
Memory limit: 64 MB

Let us define a *complexity* of an integer as the number of its divisors. Your task is to find the most complex integer in range from 1 to n. If there are many such integers, you should find the minimal one.

Input

The first line contains the number of testcases t ($1 \le t \le 100$). The i-th of the following t lines contains one integer n_i ($1 \le n_i \le 10^{18}$).

Output

For each testcase output the answer on a separate line. The i-th line should contain the most complex integer in range from 1 to n_i and its complexity, separated with space.

Sample

input	output
5 1 10 100 1000 10000	1 1 6 4 60 12 840 32 7560 64

```
/*
We have an array of prime [16], why only these prime numbers, because of these prime numbers The product
is greater than 10^16,
and in terms of the nature of the inverse prime,
such as 2^t1*3^t2*5^t3*...p1^x***p2^y, suppose p1 is greater than all of prime[] Prime number,
because the product of these prime numbers is greater than 10^16, if
we add p1 in this continuous product, then there must be at least one prime[i] not in this continuous
product,
but for the total number of factors In fact,
we care about the size of the power, not the size of the prime factor, which means that if the prime
factor is larger, the number of factors will be smaller.
The introduction of anti-prime number knowledge: the
first point of anti-prime number: g(x) indicates that x contains the number of factors, and 0<i<=x then
g(i)<=x; the
second characteristic of anti-prime number: 2^t1* 3^t2^5^t3*7^t4..... Here is t1>=t2>=t3>=t4...
Prove: if ti<tj, where i<j, since pi is less than pj, then pi ^tj*pj^ti<pi^ti*pj^tj, so there
are cases where the number of factors is the same, but x is smaller, which contradicts the definition of
anti-prime numbers.
 */

 #include<stdio.h>
#defineUnsigned Long Long LLU
#define pnum. 17
int Prime [pnum] = { 2 , . 3 , . 5 , . 7 , . 11 , 13 is , . 17 , . 19 , 23 is , 29 , 31 is , 37 [ , 41 is
, 43 is , 47 , 53 is ,
        59 };
LLU n, nmin;
int nmax;
void dfs( int pstart, int limit, LLU now, int nnum) {
    int I;
    LLU Prime = P [Pstart];
    Double nnow;
    IF (now> n-) {
        return ;
    }
    IF (nNum> Nmax of) {
        Nmin = now;
        Nmax of nNum =;
    }
    IF (Nmax of && now nNum == <Nmin ) {
        Nmin = now;
    }
    for (I = . 1 ; I <= limit; P * = Prime [Pstart], I ++) {
        nnow = ( Double ) now;
        IF (nnow * P> n-) {
            BREAK ;
        } the else {
            Dfs(pstart + 1 , i, now * p, nnum * (i + 1 ));
        }
    }
}
Int main () {
#ifndef ONLINE_JUDGE
    The freopen ( " data.in. " , " R & lt " , stdin);
#endif
    int T;
    the while (~ Scanf ( " % D " , & T)) {
        the while (T-- ) {
            Scanf ( " % LLU " , & n-);
            Nmin = . 1 , Nmax of = . 1 ;
            DFS ( 0 , 60 , . 1 , . 1 );
            the printf ( "% llu %d\n " , nmin, nmax);
        }
    }
    return  0 ;
}
```

```cpp
#include <cstdio>
#include <vector>

using namespace std;

bool is_prime(int n){
    if(n == 2) return true;
    if(n % 2 == 0) return false;

    for(int i = 2;i <= n / i;++i)
        if(n % i == 0) return false;

    return true;
}

vector<int> p;
long long n,best,ans;

void solve(int pos, long long x, long long div, int last){
    if(div > best || (div == best && x < ans)){
        best = div;
        ans = x;
    }

    if(pos == 35) return;

    for(int i = 1;i <= last && x <= n / p[pos];++i){
        x *= p[pos];
        solve(pos + 1,x,div * (i + 1),i);
    }
}

int main(){
    for(int i = 2;p.size() < 35;++i)
        if(is_prime(i))
            p.push_back(i);

    int T;

    scanf("%d",&T);

    while(T--){
        scanf("%lld",&n);

        best = 1;
        ans = 1;

        solve(0,1,1,60);

        printf("%lld %lld\n",ans,best);
    }

    return 0;
}
```

```c
#include <stdio.h>
unsigned long long n, res;
int p, primes[] = {2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 51, 53, 59, 61, 67, 71};

unsigned long long mul(unsigned long long a, unsigned long long b){
    unsigned long long res = 0;
    while (b){
        if (b & 1LL) res = (res + a);
        if (res >= n) return 0;
        a = (a << 1LL);
        b >>= 1LL;
    }
    return res;
}

void backtrack(int i, int lim, unsigned long long val, unsigned long long r){
    if (r > res) res = r;
    if (i == p) return;
    int d;
    unsigned long long x = val;
    for (d = 1; d <= lim; d++){
        x = mul(x, primes[i]);
        if (x == 0) return;
        backtrack(i + 1, d, x, r * (d + 1));
    }
}

int main(){
    /* Tested for n <= 10^18 */
    p = sizeof(primes) / sizeof(int);
    while (scanf("%llu", &n) != EOF){
        res = 0;
        backtrack(0, 100, 1, 1);
        printf("Maximum number of divisors of any number less than %llu = %llu\n", n, res);
    }
    return 0;
}

Input
10
100
1000000
1000000000
1000000000000
1000000000000000
10000000000000000000

Output
Maximum number of divisors of any number less than 10 = 4
Maximum number of divisors of any number less than 100 = 12
Maximum number of divisors of any number less than 1000000 = 240
Maximum number of divisors of any number less than 1000000000 = 1344
Maximum number of divisors of any number less than 1000000000000 = 6720
Maximum number of divisors of any number less than 1000000000000000 = 26880
Maximum number of divisors of any number less than 10000000000000000000 = 103680
```