**SPOJ - SQFREE : Square-free integers:**

In number theory, an integer square-free if it is not divisible by a perfect square numbers (4,9,25,36,49, etc) except 1.

First line contains an integer T, the number of test cases ($T \le 100$). The following T lines each contains one positive integer: n, where $n \le 10^{14}$.

T lines, on each line output the number of (positive) square-free integers not larger than n.

| Sample input: | Sample Output: |
|---|---|
| 3 | |
| 1 | 1 |
| 1000 | 608 |
| 100000000000000 | 60792710185947 |

---

Idea:

- ➔ First, we calculate the number of integers from 1 to n that is divisible by any perfect square numbers.
- ➔ Answer = n – number of non-square free integers from 1 to n.
- ➔ say, cnt = number of non-square free integers from 1 to n. ( so, we want to calculate how many integers from 1 to n are divisible by any perfect square numbers(4,9,16,25…) ).
- ➔ First observation is, we can ignore perfect square numbers like ($4^2=16$, $6^2=36$…). Because the number divided by ($4^2=16$) is also divided by ($2^2=4$). So we can only count the numbers which is divisible by any of ($2^2=4$,$3^2=9$,$5^2=25$,$7^2=49$) them.
- ➔ Now, it is a basic inclusion-exclusion problem.
- ➔ Here we can't use bitmask because there are many perfect square numbers in $10^{14}$. We have to use backtracking.
- ➔ The most important observation is after multiply some perfect square numbers(maximum 7) the value will exceeds n. so we return from this.

Sample Code:
```
#pragma comment(linker,"/STACK:2000000")
#pragma comment(linker,"/HEAP:2000000")

#define ll long long  #define mx 10000005
bool prime[mx]; ll n,cnt; vector<ll>vv;

void sieve(){
    vv.push_back(4);
    for(ll i=3; i<=10000000; i+=2){
        if(prime[i]==false){
            vv.push_back(i*i);
            for(ll j=i*i; j<=10000000; j+=i+i){
                prime[j]=true;
            }
        }
    }
}
void fun(ll pos, ll k, ll val){
    if(pos==vv.size()){
        if(k==0)return;    // this case(k==0) must be consider
        if(k%2==1)cnt += (n/val);
        else cnt -= (n/val);
        return;
    }
    ll can=(n/val); //  don't use if(val*vv[pos]<=n).it may cause overflow.
    if(vv[pos]<=can){
        fun(pos+1,k+1,val*vv[pos]);
        fun(pos+1,k,val);
    }
    else fun(vv.size(),k,val);
    return;
}
ll solve(){
    cnt=0;     // cnt holds how many numbers is not square free
    fun(0,0,1);
    return n-cnt;
}
int main(){
    sieve();
    int t; scanf("%d",&t);
    for(int ks=1; ks<=t; ks++){
        scanf("%lld",&n);
        ll ans = solve();
        printf("%lld\n",ans);
    }
}
```