```
#include<bits/stdc++.h>
#include <ext/pb_ds/assoc_container.hpp>
#include <ext/pb_ds/tree_policy.hpp>
using namespace std;
using namespace __gnu_pbds;
template <typename T>
using ordered_set= tree<T, null_type,less<T>,
rb_tree_tag,tree_order_statistics_node_update>;  //ordered_set
template <typename T>
using multi_ordered_set= tree<T, null_type,less_equal<T>,
rb_tree_tag,tree_order_statistics_node_update>;  //multiple_ord
ered_set

// #Define
#define Good_Luck
ios_base::sync_with_stdio(0);cin.tie(0);cout.tie(0);
#define ll long long
#define int long long
#define ld long double
#define pb push_back
#define eb emplace_back
#define pp pop_back
#define pf push_front
#define ub upper_bound
#define lb lower_bound
#define MP make_pair
#define YES cout<<"YES\n"
#define NO  cout<<"NO\n"
#define emo cout<<"('_')\n"
#define all(v) v.begin(),v.end()
#define rall(v) v.rbegin(),v.rend()
#define extra(n) fixed<<setprecision(n)
#define For(n) for(ll i=0;i<n;i++)
#define vll vector<ll>
#define pll pair<ll,ll>
#define mpl map<ll,ll>
#define umpl unordered_map<ll,ll>
#define lll list<ll>
#define stl stack<ll>
#define qll queue<ll>
#define pql priority_queue<ll>
#define sll set<ll>
#define msl multiset<ll>
#define osl ordered_set<ll>
#define mosl multi_ordered_set<ll>
#define mem(v,flag) memset(v, flag, sizeof(v))

// Const
const ll mod=1000000007;
const ll N=200005;
const ll inf=LLONG_MAX;
const ll minf=LLONG_MIN;

// Mathematical functions
ll gcd(ll a, ll b) {if (b==0) return a; return gcd(b,a%b);} //__gcd
ll lcm(ll a, ll b) {return (a/gcd(a,b) * b);}
ll power(ll x, ll y) { ll a=1;for(ll i=0;i<y;i++) a*= x;return a;}
ll square_root(ll x) {
   ll low=1,high=3e9,ans=1; while(low<=high) {
      ll mid = (low + high)/2;
      if (mid*mid<=x){ans=mid;low=mid+1;} else high=mid-1;
   } return ans;
}

// Sorting
bool sorta(pll a,pll b){ return a.second<b.second;}
bool sortd(pll a,pll b){ return a.second>b.second;}

bool isPrime(ll n){
   if (n<=1)return false; if(n<=3)return true; if(n%2==0 || n%3==0)
return false;
   for (ll i=5;i*i<=n;i=i+6) if(n%i==0 || n%(i+2)==0) return false;
   return true;
}
ll factorial(ll n){
   if(n<0) return -1;
   else if(n==0) return 1;
   else return n*factorial(n-1);
}
ll permutation(ll n, ll r) {
   if (n < r) return -1;
   return factorial(n)/factorial(n-r);
}
ll combination(ll n,ll r){
   if(n<r) return -1;
   else return factorial(n)/factorial(n-r)/factorial(r);
}
void primeFactors(int n){
   while(n%2==0){
      cout<<2<<" ", n=n/2;
   }
   for(int i=3;i*i<=n;i=i+2){
      while(n%i==0){
         cout<<i<<" ", n=n/i;
      }
   }
   if(n>2) cout<<n<<" ";
}
struct SimpleHash {
   long long len, base, mod;
   vector<long long> P, H, R;
   SimpleHash() {}
   SimpleHash(string str, long long b, long long m) {
      base = b, mod = m, len = str.size();
      P.resize(len + 4, 1), H.resize(len + 3, 0), R.resize(len + 3, 0);
      for (long long i = 1; i <= len + 3; i++)
         P[i] = (P[i - 1] * base) % mod;
      for (long long i = 1; i <= len; i++)
         H[i] = (H[i - 1] * base + str[i - 1]+1007) % mod;
      for (long long i = len; i >= 1; i--)
         R[i] = (R[i + 1] * base + str[i - 1]+1007) % mod;
   }
   inline long long range_hash(long long l, long long r) {
      long long hashval = (H[r + 1] - (P[r - l + 1] * H[l] % mod))%mod;
      return (hashval < 0 ? hashval + mod : hashval);
   }
   inline long long reverse_hash(long long l, long long r) {
      long long hashval = R[l + 1] - (P[r - l + 1] * R[r + 2] % mod);
      return (hashval < 0 ? hashval + mod : hashval);
   }
};
struct DoubleHash {
    SimpleHash sh1, sh2;
    DoubleHash() {}
    DoubleHash(string str) {
        sh1 = SimpleHash(str, 1949313259, 2091573227);
        sh2 = SimpleHash(str, 1997293877, 2117566807);
    }
    long long concate(DoubleHash& B, long long l1 , long long r1 ,
long long l2 , long long r2) {
        long long len1 = r1 - l1+1 , len2 = r2 - l2+1;
        long long x1 = sh1.range_hash(l1, r1) ,
        x2 = B.sh1.range_hash(l2, r2);
        x1 = (x1 * B.sh1.P[len2]) % 2091573227;
        long long newx1 = (x1 + x2) % 2091573227;
        x1 = sh2.range_hash(l1, r1);
        x2 = B.sh2.range_hash(l2, r2);
        x1 = (x1 * B.sh2.P[len2]) % 2117566807;
        long long newx2 = (x1 + x2) % 2117566807;
        return (newx1 << 32) ^ newx2;
    }
    inline long long range_hash(long long l, long long r) {
        return (sh1.range_hash(l, r) << 32) ^ sh2.range_hash(l, r);
    }
    inline long long reverse_hash(long long l, long long r) {
        return (sh1.reverse_hash(l, r) << 32) ^ sh2.reverse_hash(l, r);
    }
};
class DisjointSet {
    private:
        int parent[N];
        int size[N];
    public:
        DisjointSet() {
            for(int i=0;i<N;i++) {
                parent[i]=i;
                size[i]=1;
            }
        }
        void make_set(int v) {
            parent[v]=v;
            size[v]=1;
        }
        int find_set(int v) {
            if(v==parent[v]) return v;
            return parent[v]=find_set(parent[v]); // Path compression
        }
        void union_sets(int a,int b) {
            a=find_set(a);
            b=find_set(b);
            if(a!=b) {
                // Union by size
                if(size[a]<size[b]) swap(a,b);
                parent[b]=a;
                size[a]+=size[b];
            }
        }
    };
void print(vector<int>v){
    for(int i=0;i<v.size();i++) cout<<v[i]<<" ";
    cout<<endl;
}
void print(set<int>s){
    for(auto it:s) cout<<it<<" ";
    cout<<endl;
}
void print(multiset<int>s){
    for(auto it:s) cout<<it<<" ";
    cout<<endl;
}
void print(map<int,int>mp){
    for(auto it:mp) cout<<it.first<<" "<<it.second<<endl;
}
void print(stack<int>st){
    while(!st.empty()) cout<<st.top()<<" ",st.pop();
    cout<<endl;
}
void print(queue<int>q){
    while(!q.empty()) cout<<q.front()<<" ",q.pop();
    cout<<endl;
}
int time_limit=1e9,time_count=0;
bool checkTime(){
    time_count++;
    if(time_count>time_limit){
        cout<<"Time Limit Apprehension (-_-)"<<endl;
        return false;
    }
    return true;
}
vll intToBin(int n){
    vll bin(32);
    for(int i=0;i<32;i++) bin[31-i]=n&1, n>>=1;
    // for(int i=0;i<32;i++) cout<<bin[i]<<" ";
    // cout<<endl;
    return bin;
}
ll bigmod(ll a,ll p,ll m){
    if(p == 0) return 1;
    ll q = bigmod(a, p/2, m);
    if(p % 2 == 0) return (q*q) % m;
    return (q*((q*a) % m)) % m;
}
// #define LOCAL
// #include "debug.h"
#define dbg(x)

// MyTask

void solve(int &t,int &T){
    // ll in,n,m,i,j,k,x,y;
    int n; cin>>n;

}

main()
{
    Good_Luck;
    int T=1;
    cin>>T;
    for(int t=1;t<=T;t++){
        solve(t,T);
    }
}
```