#include<bits/stdc++.h>

using namespace std;

#define ll long long

#define fastio ios\_base::sync\_with\_stdio(false);cin.tie(NULL);cout.tie(NULL);

#define endl '\n'

#define test ll t; cin >> t; while(t--)

#define vec(ver, n)  vector<ll> ver(n); for (ll i = 0; i < n; i++) cin >> ver[i];

#define pb push\_back

#define all(x) (x).begin(),(x).end()

bool pow2(ll n)

{

    if(n==0) return false;

    return (ceil(log2(n)) == floor(log2(n)));

}

const int32\_t N=1e5+10;

const long long INF=1e18 + 2;

const long long \_INF=-1e18;

const int mod1=998244353;

const int mod=1000000007;

ll bin\_expo(ll x, ll y) { ll res = 1; while (y) {if (y % 2) res = (res \* x % mod) % mod; x = (x \* x) % mod; y /= 2; } return res;}

ll mod\_inv(ll x) {return bin\_expo(x, mod - 2);}

ll mod\_add(ll a, ll b) {a = a % mod; b = b % mod; return (((a + b) % mod) + mod) % mod;}

ll mod\_mul(ll a, ll b) {a = a % mod; b = b % mod; return (((a \* b) % mod) + mod) % mod;}

ll mod\_sub(ll a, ll b) {a = a % mod; b = b % mod; return (((a - b) % mod) + mod) % mod;}

ll mod\_div(ll a, ll b) {a = a % mod; b = b % mod; return (mod\_mul(a, mod\_inv(b)) + mod) % mod;}

unsigned long long fac[200000 + 1];//fac[0] = 1;

unsigned long long power(unsigned long long x,int y, int p){

    unsigned long long res = 1; // Initialize result

    x = x % p; // Update x if it is more than or

    // equal to p

    while (y > 0)

    {

        // If y is odd, multiply x with result

        if (y & 1)

            res = (res \* x) % p;

        // y must be even now

        y = y >> 1; // y = y/2

        x = (x \* x) % p;

    }

    return res;

}

// Returns n^(-1) mod p

unsigned long long modInverse(unsigned long long n,int p)

{

    return power(n, p - 2, p);

}

// Returns nCr % p using Fermat's little

// theorem.

unsigned long long nCrModPFermat(unsigned long long n,int r, int p)

{

    // If n<r, then nCr should return 0

    if (n < r)

        return 0;

    // Base case

    if (r == 0)

        return 1;

    // Fill factorial array so that we

    // can find all factorial of r, n

    // and n-r

    return (fac[n] \* modInverse(fac[r], p) % p\* modInverse(fac[n - r], p) % p)% p;

}

signed main(){

    fastio

    fac[0] = 1;

    for(int i = 1; i<= 200000; i++){

        fac[i] = (fac[i-1]\*i) % mod;

    }

    test{

    }

    return 0;

}