```
#include <bits/stdc++.h>
using namespace std;

typedef long long ll;

const int MOD = 998244353;

// Fast exponentiation: computes x^y % MOD
ll power_mod(ll x, ll y) {
    ll res = 1;
    x %= MOD;
    while(y > 0){
        if(y & 1){
            res = res * x % MOD;
        }
        x = x * x % MOD;
        y >>=1;
    }
    return res;
}

int main(){
    ios::sync_with_stdio(false);
    cin.tie(0);
    int T;
    cin >> T;
    while(T--){
        ll N, M;
        cin >> N >> M;
        ll K = N / 2;
```

```
ll M_mod = N % 2;
ll pow_m_mod = (M_mod ? power_mod(M, 1) : 1LL);
// If M_mod ==1, pow(M,1)=M % MOD
if(M_mod){
    pow_m_mod = M % MOD;
}
// Precompute powers of M
// Now, compute sum_f
ll sum_f =0;
for(ll d=1; d<=M; d++){
    ll a = M / d;
    ll b = M % d;
    ll term1 = (b * ((a +1) * (a +1) % MOD)) % MOD;
    ll term2 = ((d - b) * (a * a % MOD)) % MOD;
    ll sum_c_r_sq = (term1 + term2) % MOD;
    ll pow_sum = (K ==0) ? 1 : power_mod(sum_c_r_sq, K);
    ll contrib = pow_sum * pow_m_mod % MOD;
    sum_f = (sum_f + contrib) % MOD;
}
// Compute term to subtract
ll pow_M_K = (K ==0) ? 1 : power_mod(M, K);
ll term = pow_M_K;
if(M_mod){
    term = term * power_mod(M, 1) % MOD;
}
else{
    term = term * 1 % MOD;
}
ll add_val = (1 + M) % MOD;
term = term * add_val % MOD;
```

```
        // Subtract term
        sum_f = (sum_f - term) % MOD;
        if(sum_f <0){
            sum_f += MOD;
        }
        cout << sum_f << "\n";
    }
}
```