12003 - Array Transformer:

Write a program to transform an array A[1],A[2]. . . A[n] according to m instructions. Each instruction (L,R,v,p) means: First, calculate how many numbers from A[L] to A[R] (inclusive) are strictly less than v, call this answer k. Then, change the value of A[p] to (u*k)/(R-L+1), here we use integer division (i.e. ignoring fractional part).

The first line of input contains three integer n, m, u (1<=n<=300000, 1<=m<=50000, 1<=u<=10^9). Each of the next n lines contains an integer A[i] (1<=A[i]<=u). Each of the next
m lines contains an instruction consisting of four integers L, R, v, p (1<=L<=R<=n, 1<=v<=u, 1<=p<=n).

Print n lines, one for each integer, the final array.

Sample inout:

```
10 1 11
1 2 3 4 5 6 7 8 9 10
2 8 6 10
```

Sample Output:

```
1 2 3 4 5 6 7 8 9 6
```

Explanation: There is only one instruction: L=2, R=8, v=6, p=10. There are 4 numbers (2,3,4,5) less than 6, so k = 4. The new number in A[10] is (11*4)/(8-2+1) = 44/7 = 6.

Sample Code:

```
#define ll long long #define pb push_back
const int MAXN=300005; const int BLOCK_SIZE = 550;
ll a[MAXN];
vector<ll>block[BLOCK_SIZE+5];
ll BinarySearch(ll b, ll v)
{
    ll sz = block[b].size();
    if(sz==0)return 0;
    if(block[b][0]>=v)return 0;
    if(block[b][sz-1]<v)return sz;

    ll lo = 0; ll hi = block[b].size()-1;
    ll cnt = 0;
    while(lo<=hi)
    {
        ll md = (lo+hi)/2;
        if(block[b][md]<v){
            cnt = md+1; lo = md+1;
        }
        else{
            hi = md-1;
        }
    }
    return cnt;
}
```

```
void update(ll p,ll v)
{
    ll bp = p/BLOCK_SIZE;
    ll vp = lower_bound(block[bp].begin(),block[bp].end(),a[p])-block[bp].begin();
    block[bp][vp] = v;
    sort(block[bp].begin(),block[bp].end());
    a[p]=v;
}
ll query(ll l,ll r,ll v)
{
    ll bl = l/BLOCK_SIZE; ll br = r/BLOCK_SIZE; ll k = 0;

    if(bl==br){
        for(int i=l; i<=r; i++) if(a[i]<v)k++;

        return k;
    }
    ll b,e; b = l; e = ((bl+1)*BLOCK_SIZE)-1;

    for(int i=b; i<=e; i++) if(a[i]<v)k++;

    e = r; b = (br*BLOCK_SIZE);
    for(int i=b; i<=e; i++) if(a[i]<v)k++;

    for(int i=bl+1; i<=br-1; i++) k += BinarySearch(i,v);

    return k;
}
int main()
{
    ll n,m,u;
    while(scanf("%lld %lld %lld",&n,&m,&u)==3)
    {
        for(int i=0; i<n; i++) scanf("%lld",&a[i]);
        for(int i=0; i<n; i++) block[i/BLOCK_SIZE].pb(a[i]);
        for(int i=0; i<BLOCK_SIZE; i++)
            sort(block[i].begin(),block[i].end());

        while(m--) {
            ll l,r,v,p;
            scanf("%lld %lld %lld %lld",&l,&r,&v,&p);
            l--; r--; p--;
            ll k = query(l,r,v);
            ll now = (u*k)/(r-l+1);
            update(p,now);
        }
        for(int i=0; i<n; i++)printf("%lld\n",a[i]);
        for(int i=0; i<BLOCK_SIZE; i++)block[i].clear();
    }
    return 0;
}
```