Problem Link :-

https://www.hackerearth.com/problem/algorithm/ballgame/

Basic Idea :-

One can guess from the constraits that complexity of the algorithm should be either O(nm + q) or O(qm). And there is a solution with the second one.

Let's try to solve the reversed problem — answer what position some number will be at after all the operations. Check the impact of some operation on position pos. Let the operation be on some segment [l, r]. If pos is outside this segment then you can skip it. Otherwise reverse will swap a_{pos} and $a_{r-(pos-l)}$, shift will swap a_{pos} and a_{pos-1} (if pos = l then it will be r instead of (pos - 1)).

This task can be translated to the given one just by reversing the operation list.

Overall complexity: O(qm).

Ideal Solution :-

```
#include<bits/stdc++.h>
#define ll long long
#define fastIo ios_base::sync_with_stdio(0);cin.tie(0);cout.tie(0);
#define endl "\n"
using namespace std;
int main(){
	fastIo
	ll n,q,m;
	cin>>n>>q>>m;
	vector<ll> a(n);
	for(ll i=0;i<n;i++)
		cin>>a[i];
	vector<vector<ll>> v(q,vector<ll>(3));
	for(ll i=0;i<q;i++) cin>>v[i][0]>>v[i][1]>>v[i][2];
	for(ll i=0;i<m;i++){
		ll x;
		cin>>x;
		x=x-1;
		for(ll j=q-1;j>=0;j--){
			ll l=v[j][1]-1, r=v[j][2]-1;
			if(x>=l && x<=r){
				if(v[j][0]==1){
					if(x==l) x=r;
					else x=x-1;
				}
				else{
					x=(l+r-x)%n;
				}
			}
		}
		cout<<a[x]<<" ";
	}
  return 0;
}
```