```cpp
/* Codeforces 706D. Vasiliy's Multiset (time limit per test 4 secondsI
You are given q queries and a multiset A, initially containing only integer 0.
There are three types of queries:

    "+ x" — add integer x to multiset A.

    "- x" — erase one occurrence of integer x from multiset A. It's guaranteed
that at least one x is present in the multiset A before this query.

    "? x" — you are given integer x and need to compute the value ,
i.e. the maximum value of bitwise exclusive OR (also know as XOR) of integer x
and some integer y from the multiset A.
Multiset is a set, where equal elements are allowed.

Input
The first line of the input contains a single integer q (1 ≤ q ≤ 200000)
— the number of queries Vasiliy has to perform.
Each of the following q lines of the input contains one of three characters
'+', '-' or '?' and an integer xi (1 ≤ xi ≤ 109). It's guaranteed that
there is at least one query of the third type.
Note, that the integer 0 will always be present in the set A.

Output
For each query of the type '?' print one integer — the maximum value of
bitwise exclusive OR (XOR) of integer xi and some integer from the multiset A.

Input:
10
+ 8
+ 9
+ 11
+ 6
+ 1
? 3
- 8
? 3
? 8
? 11

Output:
11
10
14
13

Note
After first five operations multiset A contains integers 0, 8, 9, 11, 6 and 1.
The answer for the sixth query is integer  11 = 3 xor 8— maximum among integers,
3 xor 0 = 0, 3 xor 9 = 10, 3 xor 11 = 8, 3 xor 6 = 5, and 3 xor 1 = 2.
*/
#include<bits/stdc++.h>
using namespace std;
int msz;
struct node{
    int cnt;
    int v;
    node *next[2];
    node(){
        cnt = v = 0;
        next[0] = next[1] = NULL;
    }
}*root;
string DecToBin(int x){
    string tm;
    while(x!=0){
        tm += (x&1)+'0';
        x = x>>1;
    }
    while(tm.size()<msz) tm+='0';
    reverse(tm.begin(),tm.end());
    return tm;
}
```

```cpp
void Insert(string s,int x){
    node *cur = root;
    for(int i=0; i<s.size(); i++){
        int id = s[i]-'0';
        if(cur->next[id]==NULL) cur->next[id] = new node();
        cur = cur->next[id];
        cur->cnt++;
    }
    cur->v = x;
}
void Delete(string s){
    node *cur = root;
    for(int i=0; i<s.size(); i++){
        int id = s[i]-'0';
        cur = cur->next[id];
        cur->cnt--;
    }
}
int query(string s){
    node *cur = root;
    for(int i=0; i<s.size(); i++){
        int id = s[i]-'0';
        int nx = 1-id;
        if(cur->next[nx]==NULL || cur->next[nx]->cnt==0)cur = cur->next[id];
        else cur = cur->next[nx];
    }
    return cur->v;
}
void del(node *cur){
    for(int i=0; i<2; i++){
        if(cur->next[i]) del(cur->next[i]);
    }
    delete(cur);
}
int main(){
    root = new node();
    msz  = floor(log2(1000000000))+1;
    string zero = DecToBin(0);
    Insert(zero,0);

    int q; scanf("%d",&q);
    getchar();
    for(int i=1; i<=q; i++){
        char ch; int x;
        scanf("%c %d",&ch,&x);
        getchar();

        if(ch=='+'){
            string s = DecToBin(x);
            Insert(s,x);
        }
        else if(ch=='-'){
            string s = DecToBin(x);
            Delete(s);
        }
        else{
            string s = DecToBin(x);
            int v = query(s);
            int ans = x^v;
            printf("%d\n",ans);
        }
    }
    del(root);
    return 0;
}
```