```cpp
#include<bits/stdc++.h>
using namespace std;
#define MAXN 1024005
string s;
struct data{
    int one, lazy;
}tree[4*MAXN];
void relaxation(int nd,int b,int e){
    if(tree[nd].lazy==2){
        tree[nd].one = (e-b+1) - tree[nd].one;
    }else{
        tree[nd].one = (e-b+1) * tree[nd].lazy;
    }
}
void pushDown(int nd, int b,int e){
    if(tree[nd].lazy!=-1) {
        relaxation(nd,b,e);
        if(b!=e){
            int l=2*nd, r=2*nd+1, m=(b+e)/2;
            if(tree[nd].lazy==2) {
                if(tree[l].lazy == 0) tree[l].lazy = 1;
                else if(tree[l].lazy == 1) tree[l].lazy = 0;
                else if(tree[l].lazy == 2) tree[l].lazy =-1;
                else if(tree[l].lazy ==-1) tree[l].lazy = 2;

                if(tree[r].lazy == 0) tree[r].lazy = 1;
                else if(tree[r].lazy == 1) tree[r].lazy = 0;
                else if(tree[r].lazy == 2) tree[r].lazy =-1;
                else if(tree[r].lazy ==-1) tree[r].lazy = 2;
            }else {
                tree[l].lazy = tree[nd].lazy;
                tree[r].lazy = tree[nd].lazy;
            }
        }
        tree[nd].lazy = -1;
    }
}
void build(int nd, int b, int e){
    if(b==e){
        tree[nd].one = s[b]-'0';
        tree[nd].lazy = -1;
        return;
    }
    int l=2*nd, r=2*nd+1, m = (b+e)/2;
    build(l,b,m);
    build(r,m+1,e);
    tree[nd].one  = tree[l].one  + tree[r].one;
    tree[nd].lazy = -1;
}
int fun(int nd, int b, int e){
    if(tree[nd].lazy != -1){
        if(tree[nd].lazy == 0) return 0;
        else if(tree[nd].lazy == 1) return (e-b+1);
        else if(tree[nd].lazy == 2) return (e-b+1) - tree[nd].one;
    }else{
        return tree[nd].one;
    }
}
void update(int nd,int b,int e,int x,int y,int c){
    pushDown(nd,b,e);
    int l=2*nd, r=2*nd+1, m = (b+e)/2;
    if(b==x && e==y){
        if(c==0) tree[nd].lazy = 0;
        else if(c==1) tree[nd].lazy = 1;
        else if(c==2) tree[nd].lazy = 2;
        pushDown(nd,b,e);
        return;
    }
    if(y<=m) update(l,b,m,x,y,c);
    else if(x>m)update(r,m+1,e,x,y,c);
    else {
        update(l,b,m,x,m,c);
        update(r,m+1,e,m+1,y,c);
    }
    tree[nd].one  = fun(l,b,m)  + fun(r,m+1,e);
}
```

```cpp
int query(int nd,int b,int e,int x,int y){
    pushDown(nd,b,e);
    if(b==x && e==y)return tree[nd].one;
    int l=2*nd, r=2*nd+1, m = (b+e)/2;
    if(y<=m) return query(l,b,m,x,y);
    else if(x>m) return query(r,m+1,e,x,y);
    else return query(l,b,m,x,m) + query(r,m+1,e,m+1,y);
}
int main(){
    ios::sync_with_stdio(false); cin.tie(NULL); cout.tie(NULL);
    int tt; cin>>tt;
    for(int ks=1; ks<=tt; ks++){
        cout<<"Case "<<ks<<":"<<endl;
        s="";
        int m; cin>>m;
        while(m--){
            int t; cin>>t;
            string h; cin>>h;
            while(t--) s += h;
        }
        int n = s.size();

        build(1,0,n-1);

        int q,k=0; cin>>q;
        while(q--){
            char c; int x,y;
            cin>>c>>x>>y;
            if(c=='F'){
                update(1,0,n-1,x,y,1);
            }else if(c=='E'){
                update(1,0,n-1,x,y,0);
            }else if(c=='I'){
                update(1,0,n-1,x,y,2);
            }else{
                int ans = query(1,0,n-1,x,y);
                cout<<"Q"<<++k<<": "<<ans<<endl;
            }
        }
    }
    return 0;
}
/*
2
2
5
10
2
1000
5
F 0 17
I 0 5
S 1 10
E 4 9
S 2 10
3
3
1
4
0
2
0
2
I 0 2
S 0 8
*/
```