

```

1  #include<bits/stdc++.h>
2  using namespace std;
3  #define MAXN 1024005
4  string s;
5  struct data{
6      int one, lazy;
7  }tree[4*MAXN];
8  void relaxation(int nd,int b,int e){
9      if(tree[nd].lazy==2){
10         tree[nd].one = (e-b+1) - tree[nd].one;
11     }else{
12         tree[nd].one = (e-b+1) * tree[nd].lazy;
13     }
14 }
15 void pushDown(int nd, int b,int e){
16     if(tree[nd].lazy!=-1) {
17         relaxation(nd,b,e);
18         if(b!=e){
19             int l=2*nd, r=2*nd+1, m=(b+e)/2;
20             if(tree[nd].lazy==2) {
21                 if(tree[l].lazy == 0) tree[l].lazy = 1;
22                 else if(tree[l].lazy == 1) tree[l].lazy = 0;
23                 else if(tree[l].lazy == 2) tree[l].lazy = -1;
24                 else if(tree[l].lazy == -1) tree[l].lazy = 2;
25
26                 if(tree[r].lazy == 0) tree[r].lazy = 1;
27                 else if(tree[r].lazy == 1) tree[r].lazy = 0;
28                 else if(tree[r].lazy == 2) tree[r].lazy = -1;
29                 else if(tree[r].lazy == -1) tree[r].lazy = 2;
30             }else {
31                 tree[l].lazy = tree[nd].lazy;
32                 tree[r].lazy = tree[nd].lazy;
33             }
34         }
35         tree[nd].lazy = -1;
36     }
37 }
38 void build(int nd, int b, int e){
39     if(b==e){
40         tree[nd].one = s[b]-'0';
41         tree[nd].lazy = -1;
42         return;
43     }
44     int l=2*nd, r=2*nd+1, m = (b+e)/2;
45     build(l,b,m);
46     build(r,m+1,e);
47     tree[nd].one = tree[l].one + tree[r].one;
48     tree[nd].lazy = -1;
49 }
50 int fun(int nd, int b, int e){
51     if(tree[nd].lazy != -1){
52         if(tree[nd].lazy == 0) return 0;
53         else if(tree[nd].lazy == 1) return (e-b+1);
54         else if(tree[nd].lazy == 2) return (e-b+1) - tree[nd].one;
55     }else{
56         return tree[nd].one;
57     }
58 }
59 void update(int nd,int b,int e,int x,int y,int c){
60     pushDown(nd,b,e);
61     int l=2*nd, r=2*nd+1, m = (b+e)/2;
62     if(b==x && e==y){
63         if(c==0) tree[nd].lazy = 0;
64         else if(c==1) tree[nd].lazy = 1;
65         else if(c==2) tree[nd].lazy = 2;
66         pushDown(nd,b,e);
67         return;
68     }
69     if(y<=m) update(l,b,m,x,y,c);
70     else if(x>m)update(r,m+1,e,x,y,c);
71     else {
72         update(l,b,m,x,m,c);
73         update(r,m+1,e,m+1,y,c);
74     }
75     tree[nd].one = fun(l,b,m) + fun(r,m+1,e);
76 }

```

```

77  int query(int nd,int b,int e,int x,int y){
78      pushDown(nd,b,e);
79      if(b==x && e==y)return tree[nd].one;
80      int l=2*nd, r=2*nd+1, m = (b+e)/2;
81      if(y<=m) return query(l,b,m,x,y);
82      else if(x>m) return query(r,m+1,e,x,y);
83      else return query(l,b,m,x,m) + query(r,m+1,e,m+1,y);
84  }
85  int main(){
86      ios::sync_with_stdio(false); cin.tie(NULL); cout.tie(NULL);
87      int tt; cin>>tt;
88      for(int ks=1; ks<=tt; ks++){
89          cout<<"Case "<<ks<<": "<<endl;
90          s="";
91          int m; cin>>m;
92          while(m--){
93              int t; cin>>t;
94              string h; cin>>h;
95              while(t--) s += h;
96          }
97          int n = s.size();
98
99          build(1,0,n-1);
100
101          int q,k=0; cin>>q;
102          while(q--){
103              char c; int x,y;
104              cin>>c>>x>>y;
105              if(c=='F'){
106                  update(1,0,n-1,x,y,1);
107              }else if(c=='E'){
108                  update(1,0,n-1,x,y,0);
109              }else if(c=='I'){
110                  update(1,0,n-1,x,y,2);
111              }else{
112                  int ans = query(1,0,n-1,x,y);
113                  cout<<"Q"<<"+k<<": "<<ans<<endl;
114              }
115          }
116      }
117      return 0;
118  }
119  /*
120  2
121  2
122  5
123  10
124  2
125  1000
126  5
127  F 0 17
128  I 0 5
129  S 1 10
130  E 4 9
131  S 2 10
132  3
133  3
134  1
135  4
136  0
137  2
138  0
139  2
140  I 0 2
141  S 0 8
142  */

```