```cpp
/**Header file**/
#include<cstdio>
#include<cstring>
#include<cmath>
#include<cstdlib>
#include<cctype>
#include<algorithm>
#include<string>
#include<vector>
#include<queue>
#include<map>
#include<set>
#include<sstream>
#include<stack>
#include<list>
#include<iostream>
#include<assert.h>

/**Define file I/O **/
#define f_input freopen("input.txt","r",stdin)
#define f_output freopen("output.txt","w",stdout)

/**Define memory set function**/
#define mem(x,y) memset(x,y,sizeof(x))
#define CLEAR(x) memset(x,0,sizeof(x))

/**Define function and object**/
#define pb push_back
#define Sort(v) sort(v.begin(),v.end())
#define RSort(v) sort(v.rbegin(),v.rend())
#define CSort(v,C) sort(v.begin(),v.end(),C)
#define all(v) (v).begin(),(v).end()
#define sqr(x) ((x)*(x))
#define find_dist(a,b) sqrt(sqr(a.x-b.x)+sqr(a.y-b.y))

/**Define constant value**/
#define ERR 1e-9
#define pi 2*acos(0)
#define PI 3.141592653589793

/**Define input**/
#define scanint(a) scanf("%d",&a)
#define scanLLD(a) scanf("%lld",&a)
#define scanstr(s) scanf("%s",s)
#define scanline(l) scanf(" %[^\n]",l);

/**Define Bitwise operation**/
#define check(n, pos) (n & (1<<(pos)))
#define biton(n, pos) (n | (1<<(pos)))
#define bitoff(n, pos) (n & ~(1<<(pos)))

/**Define color**/
#define WHITE 0
#define GREY 1
#define BLACK 2

using namespace std;
```

```
58
59 /**Typedef**/
60 typedef vector<int> vint;
61 typedef vector< vint > vint2D;
62 typedef vector<string> vstr;
63 typedef vector<char>vchar;
64 typedef vector< vchar >vchar2D;
65 typedef queue<int> Qi;
66 typedef queue< Qi > Qii;
67 typedef map<int,int> Mii;
68 typedef map<string,int> Msi;
69 typedef map<int,string> Mis;
70 typedef stack<int> stk;
71 typedef pair<int,int> pp;
72 typedef pair<int, pp > ppp;
73 typedef long long int LLD;
74 const int inf=0x7FFFFFFF;
75
76 /**Template & structure**/
77 struct point_int{int x,y;point_int(){}point_int(int a,int b){x=a,y=
    b;}}; ///Point for x,y (int) coordinate in 2D space
78 struct point_double{double x,y;point_double(){}point_double(double
    a,double b){x=a,y=b;}}; ///Point for x,y (double) coordinate in
     2D space
79 struct Node{int v,w;Node() {}bool operator<(const Node &a)const{
    return w>a.w;}Node(int _v,int _w){v=_v,w=_w;}};///Node for
    Dijkstra
80 template<class T>T gcd(T a,T b){return b == 0 ? a : gcd(b, a % b);}
81 template<typename T>T lcm(T a, T b) {return a / gcd(a,b) * b;}
82 template<class T>T big_mod(T n,T p,T m){if(p==0)return (T)1;T x=
    big_mod(n,p/2,m);x=(x*x)%m;if(p&1)x=(x*n)%m;return x;}
83 template<class T>T multiplication(T n,T p,T m){if(p==0)return (T)0;
    T x=multiplication(n,p/2,m);x=(x+x)%m;if(p&1)x=(x+n)%m;return x
    ;}
84 template<class T>T my_pow(T n,T p){if(p==0)return 1;T x=my_pow(n,p
    /2);x=(x*x);if(p&1)x=(x*n);return x;} ///n to the power p
85 template <class T> double getdist(T a, T b){return sqrt((a.x - b.x)
     * (a.x - b.x) + (a.y - b.y) * (a.y - b.y));}/// distance
    between a & b
86 template <class T> T extract(string s, T ret) {stringstream ss(s);
    ss >> ret; return ret;}/// extract words or numbers from a line
87 template <class T> string tostring(T n) {stringstream ss; ss << n;
    return ss.str();}/// convert a number to string
88 template<class T> inline T Mod(T n,T m) {return (n%m+m)%m;} ///For
    Positive Negative No.
89 template<class T> T MIN3(T a,T b,T c) {return min(a,min(b,c));} ///
     minimum of 3 number
90 template<class T> T MAX3(T a,T b,T c) {return max(a,max(b,c));} ///
    maximum of 3 number
91 template <class T> void print_vector(T &v){int sz=v.size();if(sz)
    cout<<v[0];for(int i = 1; i < sz; i++)cout << ' '<<v[i];cout<<
    endl;}/// prints all elements in a vector
92 bool isVowel(char ch){ ch=toupper(ch); if(ch=='A'||ch=='U'||ch=='I'
    ||ch=='O'||ch=='E') return true; return false;}
93 bool isConsonant(char ch){if (isalpha(ch) && !isVowel(ch)) return
    true; return false;}
94
```

```
/**Shortcut input function**/
int read_int(){int n;scanf("%d",&n);return n;}
int read_LLD(){LLD n;scanf("%lld",&n);return n;}
inline int buffer_input() { char inp[1000]; scanstr(inp); return
    atoi(inp); }

/**Direction**/
///int col[8] = {0, 1, 1, 1, 0, -1, -1, -1};int row[8] = {1, 1, 0,
    -1, -1, -1, 0, 1}; ///8 Direction
///int col[4] = {1, 0, -1, 0};int row[4] = {0, 1, 0, -1}; ///4
    Direction
///int dx[]={2,1,-1,-2,-2,-1,1,2};int dy
    []={1,2,2,1,-1,-2,-2,-1};///Knight Direction
///int dx[]={-1,-1,+0,+1,+1,+0};int dy[]={-1,+1,+2,+1,-1,-2}; ///
    Hexagonal Direction


/*****************************Ajaira Jinish Sesh
    *****************************/
string inp;
char xx[100000];
int tree[10000000];
int lazy[10000000];
void build(int left,int right,int indx)
{
    if(left==right)
    {
        tree[indx]=inp[left]-48;
        lazy[indx]=inp[left]-48;
        return;
    }
    int mid=(left+right)/2;
    build(left,mid,2*indx);
    build(mid+1,right,2*indx+1);
    tree[indx]=tree[2*indx]+tree[2*indx+1];
    lazy[indx]=-1;
}
void lazy_propagation(int l,int r,int indx,int val)
{
    if(val==-1)return;
    if(val!=2)
    {
        tree[indx]=(r-l+1)*val;
        lazy[indx]=val;
        return;
    }
    tree[indx]=r-l+1-tree[indx];
    if(lazy[indx]==1)lazy[indx]=0;
    else if(lazy[indx]==0)lazy[indx]=1;
    else if(lazy[indx]==2)lazy[indx]=-1;
    else lazy[indx]=2;
}
void update(int l,int r,int indx,int x,int y,int val)
{
    int mid=(l+r)/2;
    if(lazy[indx]!=-1)
    {
```

```
146         if(l!=r)
147         {
148             lazy_propagation(l,mid,2*indx,lazy[indx]);
149             lazy_propagation(mid+1,r,2*indx+1,lazy[indx]);
150             tree[indx]=tree[2*indx]+tree[2*indx+1];
151             lazy[indx]=-1;
152         }
153     }
154     if(x<=l&&y>=r)
155     {
156         if(l!=r)
157         {
158             lazy_propagation(l,mid,2*indx,val);
159             lazy_propagation(mid+1,r,2*indx+1,val);
160             tree[indx]=tree[2*indx]+tree[2*indx+1];
161             lazy[indx]=-1;
162         }
163         else
164         {
165             lazy_propagation(l,r,indx,val);
166             lazy[indx]=-1;
167         }
168         return;
169     }
170     if(x<=mid)update(l,mid,2*indx,x,y,val);
171     if(y>mid)update(mid+1,r,2*indx+1,x,y,val);
172     tree[indx]=tree[2*indx]+tree[2*indx+1];
173     lazy[indx]=-1;
174 }
175 int query(int l,int r,int indx,int x,int y)
176 {
177     int mid=(l+r)/2;
178     if(lazy[indx]!=-1)
179     {
180         if(l!=r)
181         {
182             lazy_propagation(l,mid,2*indx,lazy[indx]);
183             lazy_propagation(mid+1,r,2*indx+1,lazy[indx]);
184             tree[indx]=tree[2*indx]+tree[2*indx+1];
185             lazy[indx]=-1;
186         }
187     }
188     if(x<=l&&y>=r)return tree[indx];
189     int a1=0,a2=0;
190     if(x<=mid)a1=query(l,mid,2*indx,x,y);
191     if(y>mid)a2=query(mid+1,r,2*indx+1,x,y);
192     return a1+a2;
193 }
194 int main()
195 {
196     #ifdef _ANICK_
197     //f_input;
198     #endif // _ANICK_
199     int test=buffer_input();
200     for(int t=1;t<=test;t++)
201     {
202         inp="";
```

```
        int n=buffer_input();
        while(n--)
        {
            int m=buffer_input();
            scanf("%s",xx);
            for(int i=0;i<m;i++)
            {
                inp+=xx;
            }
        }
        int N=inp.size()-1;
        build(0,N,1);
        int Q=buffer_input();
        printf("Case %d:\n",t);
        int q=1;
        while(Q--)
        {
            char X[100];
            scanf("%s",X);
            int x=buffer_input();
            int y=buffer_input();
            if (strcmp(X,"F") == 0)
            {
                update(0,N,1,x,y,1);
            }

            else if (strcmp(X,"E") == 0)
            {
                update(0,N,1,x,y,0);
            }

            else if (strcmp(X,"I")==0)
            {
                update(0,N,1,x,y,2);
            }

            else
            {
                printf ("Q%d: %d\n",q++,query(0,N,1,x,y));
            }
        }
    }
    return 0;
}
```