```cpp
#include<stdio.h>
#include<string.h>
#include<utility>
using namespace std;

#define MAX 1123456
#define left(p) (p) << 1
#define right(p) ((p) << 1) + 1

typedef struct { int barb, bucc; }pirate_t;

int n, land[MAX], lazy[4 * MAX];
pirate_t st[4 * MAX];

void build(int p, int l, int r) {
  int meio = (l + r) / 2;
  if (l == r) { st[p].barb = !land[l]; st[p].bucc = land[l]; return; }
  build(left(p), l, meio);
  build(right(p), meio + 1, r);
  st[p].barb = st[left(p)].barb + st[right(p)].barb;
  st[p].bucc = st[left(p)].bucc + st[right(p)].bucc;
}

void range_update(int p, int l, int r, int i, int j, int op) {
  int meio = (l + r) / 2;
  if (lazy[p]) {
    if (lazy[p] == 3) swap(st[p].barb, st[p].bucc);
    else if (lazy[p] == 1) {
      st[p].barb += st[p].bucc; st[p].bucc = 0;
    } else {
      st[p].bucc += st[p].barb; st[p].barb = 0;
    }
    if (l != r) {
      if (lazy[p] == 3) {
        if (lazy[left(p)] == 3) lazy[left(p)] = 0;
        else if (lazy[left(p)] == 2) lazy[left(p)] = 1;
        else if (lazy[left(p)] == 1) lazy[left(p)] = 2;
        else lazy[left(p)] = 3;
        if (lazy[right(p)] == 3) lazy[right(p)] = 0;
        else if (lazy[right(p)] == 2) lazy[right(p)] = 1;
        else if (lazy[right(p)] == 1) lazy[right(p)] = 2;
        else lazy[right(p)] = 3;
      } else { lazy[left(p)] = lazy[right(p)] = lazy[p]; }
    }
    lazy[p] = 0;
  }
  if (i > r || j < l) return;
  if (i <= l && j >= r) {
    if (op == 3) swap(st[p].barb, st[p].bucc);
    else if (op == 1) {
      st[p].barb += st[p].bucc; st[p].bucc = 0;
    } else {
      st[p].bucc += st[p].barb; st[p].barb = 0;
    }
    if (l != r) {
      if (op == 3) {
        if (lazy[left(p)] == 3) lazy[left(p)] = 0;
        else if (lazy[left(p)] == 2) lazy[left(p)] = 1;
        else if (lazy[left(p)] == 1) lazy[left(p)] = 2;
        else lazy[left(p)] = 3;
        if (lazy[right(p)] == 3) lazy[right(p)] = 0;
        else if (lazy[right(p)] == 2) lazy[right(p)] = 1;
        else if (lazy[right(p)] == 1) lazy[right(p)] = 2;
        else lazy[right(p)] = 3;
      } else { lazy[left(p)] = lazy[right(p)] = op; }
    }
    return;
  }
  range_update(left(p), l, meio, i, j, op);
  range_update(right(p), meio + 1, r, i, j, op);
  st[p].barb = st[left(p)].barb + st[right(p)].barb;
  st[p].bucc = st[left(p)].bucc + st[right(p)].bucc;
}

```

```
75  pirate_t rmq(int p, int l, int r, int i, int j) {
76    int meio = (l + r) / 2;
77    pirate_t p1, p2, ret;
78    if (i > r || j < l) { ret.bucc = -1; return ret; }
79    if (lazy[p]) {
80      if (lazy[p] == 3) swap(st[p].barb, st[p].bucc);
81      else if (lazy[p] == 1) {
82        st[p].barb += st[p].bucc; st[p].bucc = 0;
83      } else {
84        st[p].bucc += st[p].barb; st[p].barb = 0;
85      }
86      if (l != r) {
87        if (lazy[p] == 3) {
88          if (lazy[left(p)] == 3) lazy[left(p)] = 0;
89          else if (lazy[left(p)] == 2) lazy[left(p)] = 1;
90          else if (lazy[left(p)] == 1) lazy[left(p)] = 2;
91          else lazy[left(p)] = 3;
92          if (lazy[right(p)] == 3) lazy[right(p)] = 0;
93          else if (lazy[right(p)] == 2) lazy[right(p)] = 1;
94          else if (lazy[right(p)] == 1) lazy[right(p)] = 2;
95          else lazy[right(p)] = 3;
96        } else { lazy[left(p)] = lazy[right(p)] = lazy[p]; }
97      }
98      lazy[p] = 0;
99    }
100   if (l >= i && r <= j) return st[p];
101   p1 = rmq(left(p), l, meio, i, j);
102   p2 = rmq(right(p), meio + 1, r, i, j);
103   if (p1.bucc == -1) return p2;
104   if (p2.bucc == -1) return p1;
105   ret.barb = p1.barb + p2.barb;
106   ret.bucc = p1.bucc + p2.bucc;
107   return ret;
108 }
109
110 int main(void) {
111   int tcasos, caso, m, t, i, j, tmpsize, query, a, b, q;
112   char tmp[51], c; pirate_t resp;
113   scanf("%d", &tcasos);
114   for (caso = 1; caso <= tcasos; caso++) {
115     n = 0;
116     scanf("%d", &m);
117     while (m--) {
118       scanf("%d", &t);
119       scanf(" %s", tmp);
120       for (i = 0, tmpsize = (int)strlen(tmp); i < t; i++)
121         for (j = 0; j < tmpsize; j++)
122           land[n++] = tmp[j] - '0';
123     }
124     memset(lazy, 0, sizeof(lazy)); build(1, 0, n - 1);
125     scanf("%d", &q); printf("Case %d:\n", caso);
126     for (query = 0; q--; ) {
127       scanf(" %c %d %d", &c, &a, &b);
128       switch (c) {
129       case 'F': range_update(1, 0, n - 1, a, b, 2); break;
130       case 'E': range_update(1, 0, n - 1, a, b, 1); break;
131       case 'I': range_update(1, 0, n - 1, a, b, 3); break;
132       case 'S':
133         resp = rmq(1, 0, n - 1, a, b); query++;
134         printf("Q%d: %d\n", query, resp.bucc);
135         break;
136       }
137     }
138   }
139   return 0;
140 }
```