伸展树

A simple problem with integers

```
//POJ-3468    A simple problem with integers
//You have N integers, A1, A2, ... , AN. You need to deal with two kinds of operations.
One type of operation is to add some given number to each number in a given interval.
The other is to ask for the sum of numbers in a given interval.
//"C a b c" means adding c to each of Aa, Aa+1, ... , Ab. -10000 ≤ c ≤ 10000.
//"Q a b" means querying the sum of Aa, Aa+1, ... , Ab.
//Hint: The sums may exceed the range of 32-bit integers.

#define SIZE 100010                         //树的容量
typedef int type;                       //V的类型
#define key        (CH[ CH[r][1] ][0])     //关键树

//-------------------------下面开始节点域声明-----------------------------
//////////////////////////
int CH[SIZE][2], P[SIZE];               //0 左 1 右、父
type V[SIZE];                           //值
int len;                                //资源使用量
//////////////////////////
int CNT[SIZE];                              //本棵树的节点个数
//////////////////////////
int ADD[SIZE];                              //标记-是否加上ADD
long long SUM[SIZE];                            //本树中最小的值
//////////////////////////
//////////////////////////
//-------------------------下面开始操作节点-------------------------------
int newNode() {                             //新建一个节点（先从回收站里找）
    return len ++;
}
void initNode(int x, type v, int ch0, int ch1, int p) { //初始化每个节点
    V[x] = v;
    CH[x][0] = ch0;
    CH[x][1] = ch1;
    P[x] = p;

    CNT[x] = 1;

    SUM[x] = V[x] = v;
    ADD[x] = 0;
}
//-------------------------下面开始操作伸展树-----------------------------
void initTree() {                       //只需掉一次，清资源，设NULL节点
    len = 1;                            //从 1 开始，0 代表NULL
    initNode(0, 0, 0, 0, 0);
    CNT[0] = 0;
}
struct SplayTree {
    int r;    //根
    //-------------------------下面是基本函数-----------------------------
    SplayTree(type lv, type rv) {    //创建一个带哨兵的空SplayTree，lv/rv赋值成任何其实无所谓
        int a = newNode();
        int b = newNode();
        initNode(a, lv, 0, b, 0);
        initNode(b, rv, 0, 0, a);
        push_up(b);        push_up(a);
        r = a;
    }
    void rotate(int x, bool c) {     //将节点i，c=0 左旋转zag，c=1 右旋转zig
        int y = P[x];

        push_down(CH[x][0]);
        push_down(CH[x][1]);
        push_down(CH[y][c]);

        CH[y][!c] = CH[x][c];
        if(CH[x][c])  P[CH[x][c]] = y;
        CH[x][c] = y;

        P[x] = P[y];
        P[y] = x;

        if(P[x]) CH[ P[x] ][ CH[P[x]][1]==y ] = x;
        else     r = x;
        push_up(y);
    }
    void splay(int x, int f=0) {     //表示把结点 x 转到结点 f 的下面，默认为节点x成为根
        int y;
        bool b1, b2;
        push_down(x);
        while((y=P[x]) != f) {       //用到了P[0]=0 来处理x=0 的情况
            if(P[y] == f) {
                rotate(x, CH[y][0] == x);
            } else {
                b1 = (y==CH[P[y]][0]);
                b2 = (x==CH[  y ][0]);
                rotate((b1^b2) ? x : y, b2);
                rotate(x, b1);
            }
        }
        push_up(x);
    }
    //选择第rank个节点，并且将这个节点splay到f下面，从 0 开始
    void select(int rank, int f=0) {
        if(rank >= CNT[r] || rank<0)     return;
        int i = r;
        while(true) {
            push_down(i);
            if(CNT[ CH[i][0] ] == rank) break;
            if(CNT[CH[i][0]] > rank) {
                i = CH[i][0];
            } else {
```

```
                rank -= CNT[CH[i][0]]+1;
                i = CH[i][1];
            }
        }
        splay(i, f);
    }
    void interval(int begin, int end) {   //选中[begin, end)左闭右开的区间
        select(begin-1);
        select(end, r);
    }
    //--------------------下面开始insert和remove----------------------
    void insert(type * arr, int n) { //在root后面插入arr(似满二叉树), assume n>0 !
        int x = newNode();
        initNode(x, arr[n-1], insertDfs(arr,n-1,x), CH[r][1], r);

        if(CH[x][1])  P[ CH[x][1] ] = x;
        CH[r][1] = x;
        splay(x);
    }
    int insertDfs(type * arr, int n, int father) {
        if(n<=0)  return 0;
        int idx = n/2;
        int x = newNode();
        initNode(x, arr[idx], insertDfs(arr,idx,x),
insertDfs(arr+idx+1,n-idx-1,x), father);
        push_up(x);
        return x;
    }
    //--------------------下面开始push----------------------
    void push_up(int idx) {
        if(!idx)  return;
        //断言此处idx没有任何标记!!!!!!!!!!

        CNT[idx] = CNT[CH[idx][0]]+CNT[CH[idx][1]]+1;
        SUM[idx] = SUM[CH[idx][0]]+SUM[CH[idx][1]]+V[idx];
    }
    void push_down(int idx) {
        if(!idx)  return;
        if(ADD[idx]) {
            if(CH[idx][0])     ADD[CH[idx][0]]+=ADD[idx];
            if(CH[idx][1])     ADD[CH[idx][1]]+=ADD[idx];
            V[idx] += ADD[idx];
            SUM[idx] += (long long)ADD[idx]*CNT[idx];
            ADD[idx] = 0;
        }
    }
    //-------------------------一些其他函数--------------------
    void addKeyTree(int v) {              //将关键树每个节点都加上v
        if(!key)  return;
        ADD[key] += v;
        splay(key);
    }
    //-----------------------打印函数--------------------
    void outputKeySUM() {
        printf("%lld\n", SUM[key]);
    }
};

int arr[SIZE];

int main() {
    int n, q;
    int a, b, d;
    char c;
    while(scanf("%d%d", &n, &q) != EOF) {
        for(int i = 0; i < n; i ++) scanf("%d", arr+i);
        initTree();
        SplayTree sp(0,0);
        sp.insert(arr, n);

        while(q --) {
            scanf(" %c%d%d", &c, &a, &b);
            if(c == 'Q') {
                sp.interval(a, b+1);
                sp.outputKeySUM();
            } else {
                scanf("%d", &d);
                sp.interval(a, b+1);
                sp.addKeyTree(d);
            }
        }
    }
    return 0;
}
```

GSS6

```
/**
    SPOJ-4487 Can you answer these queries VI

    I x y: insert element y at position x (between x - 1 and x).
    D x  : delete the element at position x.
    R x y: replace element at position x with y.
    Q x y: print max{Ai + Ai+1 + .. + Aj | x <= i <= j <= y}.
 */

#define SIZE 200010                         //树的容量
typedef int type;                          //v的类型
const int inf = 10010;                     //inf
#define key        (CH[ CH[r][1] ][0])     //关键树

//-------------------------下面开始节点域声明-----------------------------
////////////////////////////
int CH[SIZE][2], P[SIZE];             //0 左 1 右、父
type V[SIZE];                         //值
int len;                              //资源使用量
////////////////////////////
int CNT[SIZE];                             //本棵树的节点个数
////////////////////////////
int SUM[SIZE];                             //本树v的和
int LL[SIZE], RR[SIZE], MM[SIZE];     //从左面连续的值的和，右面连续的值的和，连续的最大值
的和
```

```
//-------------------------下面开始操作节点-------------------------------
int newNode() {                                 //新建一个节点（先从回收站里找）
     return len ++;
}
void initNode(int x, type v, int ch0, int ch1, int p) { //初始化每个节点
     V[x] = v;
     CH[x][0] = ch0;
     CH[x][1] = ch1;
     P[x] = p;

     CNT[x] = 1;

     SUM[x] = V[x];
     MM[x] = LL[x] = RR[x] = V[x];
}
//-------------------------下面开始操作伸展树-----------------------------
void initTree() {                       //只需掉一次，清资源，设NULL节点
     len = 1;                           //从 1 开始，0 代表NULL
     initNode(0, -inf, 0, 0, 0);
     CNT[0] = SUM[0] = 0;
}
struct SplayTree {
     int r;    //根
     //-------------------------下面是基本函数---------------------------
     SplayTree(type lv, type rv) {     //创建一个带哨兵的空SplayTree，lv/rv赋值成任何其实无所谓
          int a = newNode();
          int b = newNode();
          initNode(a, lv, 0, b, 0);
          initNode(b, rv, 0, 0, a);
          push_up(b);        push_up(a);
          r = a;
     }
     void rotate(int x, bool c) {      //将节点i，c=0 左旋转zag，c=1 右旋转zig
          int y = P[x];

          CH[y][!c] = CH[x][c];
          if(CH[x][c])  P[CH[x][c]] = y;
          CH[x][c] = y;

          P[x] = P[y];
          P[y] = x;

          if(P[x]) CH[ P[x] ][ CH[P[x]][1]==y ] = x;
          else      r = x;
          push_up(y);
     }
     void splay(int x, int f=0) {      //表示把结点 x 转到结点 f 的下面，默认为节点x成为根
          int y;
          bool b1, b2;
          while((y=P[x]) != f) {       //用到了P[0]=0 来处理x=0 的情况
               if(P[y] == f) {
                    rotate(x, CH[y][0] == x);
               } else {
                    b1 = (y==CH[P[y]][0]);
                    b2 = (x==CH[  y ][0]);
                    rotate((b1^b2) ? x : y, b2);
                    rotate(x, b1);
               }
          }
          push_up(x);
     }
     //选择第rank个节点，并且将这个节点splay到f下面，从 0 开始
     void select(int rank, int f=0) {
          if(rank >= CNT[r] || rank<0)     return;
          int i = r;
          while(true) {
               if(CNT[ CH[i][0] ] == rank) break;
               if(CNT[CH[i][0]] > rank) {
                    i = CH[i][0];
               } else {
                    rank -= CNT[CH[i][0]]+1;
                    i = CH[i][1];
               }
          }
          splay(i, f);
     }
     void interval(int begin, int end) {   //选中[begin, end)左闭右开的区间
          select(begin-1);
          select(end, r);
     }
     //--------------------下面开始insert和remove----------------------
     void insert(type * arr, int n) { //在root后面插入arr(似满二叉树), assume n>0 !
          int x = newNode();
          initNode(x, arr[n-1], insertDfs(arr,n-1,x), CH[r][1], r);

          if(CH[x][1])  P[ CH[x][1] ] = x;
          CH[r][1] = x;
          splay(x);
     }
     int insertDfs(type * arr, int n, int father) {
          if(n<=0) return 0;
          int idx = n/2;
          int x = newNode();
          initNode(x, arr[idx], insertDfs(arr,idx,x), insertDfs(arr+idx+1,n-idx-1,x), father);
          push_up(x);
          return x;
     }
     int cutKeyTree() { //切断keyTree和父亲的联系而已，不将节点清除回收，并返回keyTree
          if(!key) return 0;
          int res = key;
          key = P[res] = 0;  //清除
          splay( CH[r][1] );
          return res;
     }
     //--------------------下面开始push----------------------
```

```
    void push_up(int idx) {
        if(!idx) return;
        //断言此处idx没有任何标记!!!!!!!!!!
        CNT[idx] = CNT[CH[idx][0]]+CNT[CH[idx][1]]+1;
        SUM[idx] = SUM[CH[idx][0]]+SUM[CH[idx][1]]+V[idx];//[]

        LL[idx] = max(LL[CH[idx][0]], SUM[CH[idx][0]]+V[idx] +
max(LL[CH[idx][1]],0)  );
        RR[idx] = max(RR[CH[idx][1]], SUM[CH[idx][1]]+V[idx] +
max(RR[CH[idx][0]],0)  );
        MM[idx] = max(MM[CH[idx][0]], MM[CH[idx][1]]);
        MM[idx] = max(MM[idx], V[idx] + max(0,RR[CH[idx][0]]) +
max(0,LL[CH[idx][1]]) );
    }
    //------------------------一些其他函数-------------------
    void replaceKeyTree(int v) {
        //assume CNT[key]=1
        V[key] = v;
        splay(key);
    }
    //------------------以下是打印代码------------------------
    void outputKeyMM() {
        printf("%d\n", MM[key]);
    }
};

int arr[SIZE];

int ms() {
    char c;
    bool fu = false;
    while(c=getchar(), c>'9'||c<'0') if(c=='-')    fu = true;
    int res;
    for(res=c-'0'; c=getchar(), c>='0'&&c<='9'; res=res*10+c-'0');
    if(fu)    res=-res;
    return res;
}

int main() {
    int n, q, a, b;
    char c;
    n=ms();
    initTree();
    SplayTree sp(0,0);
    for(int i = 0; i < n; i ++) arr[i]=ms();
    sp.insert(arr, n);
    q = ms();
    while(q --) {
        while(c=getchar(), c>'Z'||c<'A');
        if(c == 'I') {
            a = ms();b = ms();
            sp.select(a-1);
            sp.insert(&b, 1);
        } else if(c == 'D') {
            a = ms();
            sp.interval(a, a+1);
            sp.cutKeyTree();
        } else if(c == 'R') {
            a = ms();b = ms();
            sp.interval(a, a+1);
            sp.replaceKeyTree(b);
        } else if(c == 'Q') {
            a = ms();b = ms();
            sp.interval(a, b+1);
            sp.outputKeyMM();
        } else while(1);
    }
    return 0;
}
```

GSS6_WHU

```
const int maxq = 100000 + 2000;
const int maxn = 100000 + maxq;

struct node {
   int val, size;
   int sum, ml, mr, mc;
   node * ch[2], *pre;
} *null, *root, *S[maxn], data[maxn];

int tot, top, tmp[maxq];

inline node *New_Node(int x) {
   node *p;
   if (top)
      p = S[top--];
   else
      p = &data[tot++];
   p->val = p->sum = p->ml = p->mr = p->mc = x;
   p->size = 1;
   p->ch[0] = p->ch[1] = p->pre = null;
   return p;
}

inline void updata(node *p) {
   p->size = p->ch[0]->size + p->ch[1]->size + 1;
   p->sum = p->ch[0]->sum + p->ch[1]->sum + p->val;

   p->ml = max(p->ch[0]->ml, p->ch[0]->sum + p->val + max(p->ch[1]->ml, 0));
   p->mr = max(p->ch[1]->mr, p->ch[1]->sum + p->val + max(p->ch[0]->mr, 0));

   p->mc = max(p->ch[0]->mc, p->ch[1]->mc);
   p->mc = max(p->mc, max(p->ch[0]->mr + p->ch[1]->ml, 0) + p->val);
   p->mc = max(p->mc, max(p->ch[0]->mr, p->ch[1]->ml) + p->val);
}

inline void rotate(node *x, int c) {
   node *y = x->pre;
   y->ch[!c] = x->ch[c];
   if (x->ch[c] != null)
      x->ch[c]->pre = y;
   x->pre = y->pre;
   if (y->pre != null)
```

```
        if (y->pre->ch[0] == y)
            y->pre->ch[0] = x;
        else
            y->pre->ch[1] = x;
    x->ch[c] = y;
    y->pre = x;
    if (y == root)
        root = x;
    updata(y);
}

inline void splay(node *x, node *f) {
    for (; x->pre != f;)
        if (x->pre->pre == f)
            if (x->pre->ch[0] == x)
                rotate(x, 1);
            else
                rotate(x, 0);
        else {
            node *y = x->pre;
            node *z = y->pre;
            if (z->ch[0] == y)
                if (y->ch[0] == x)
                    rotate(y, 1), rotate(x, 1);
                else
                    rotate(x, 0), rotate(x, 1);
            else
                if (y->ch[1] == x)
                rotate(y, 0), rotate(x, 0);
            else
                rotate(x, 1), rotate(x, 0);
        }
    updata(x);
}

inline void select(int k, node *f) {
    int tmp;
    node *t;
    for (t = root;;) {
        tmp = t->ch[0]->size;
        if (tmp + 1 == k)
            break;
        if (k <= tmp)
            t = t->ch[0];
        else
            k -= tmp + 1, t = t->ch[1];
    }
    splay(t, f);
}

inline void insert(int pos, int tmp) {
    node *tmp_root;
    select(pos - 1, null);
    select(pos, root);
    tmp_root = New_Node(tmp);
    root->ch[1]->ch[0] = tmp_root;
    tmp_root->pre = root->ch[1];
    splay(root->ch[1], null);
}

inline void del(int k) {
    select(k, null);
    node *old_root = root;
    root = root->ch[1];
    root->pre = null;
    select(1, null);
    root->ch[0] = old_root->ch[0];
    root->ch[0]->pre = root;
    updata(root);
    S[++top] = old_root;
}

inline void replace(int x, int y) {
    select(x, null);
    root->val = y;
    updata(root);
}

inline node *build(int l, int r) {
    if (l > r)
        return null;
    int m = (l + r) >> 1;
    node *p = New_Node(tmp[m]);

    p->ch[0] = build(l, m - 1);
    if (p->ch[0] != null)
        p->ch[0]->pre = p;
    p->ch[1] = build(m + 1, r);
    if (p->ch[1] != null)
        p->ch[1]->pre = p;
    updata(p);
    return p;
}

int main() {
    int n, m, i, x, y;
    char c;
    null = New_Node(-1000000000);
    null->sum = null->size = 0;
    root = New_Node(-1000000000);
    root->sum = 0;
    root->ch[1] = New_Node(-1000000000);
    root->ch[1]->sum = 0;
    root->ch[1]->pre = root;
    updata(root);
    scanf("%d", &n);
    for (i = 1; i <= n; i++)
        scanf("%d", tmp + i);
    node *tmp_root = build(1, n);
    root->ch[1]->ch[0] = tmp_root;
    tmp_root->pre = root->ch[1];
    updata(root->ch[1]);
```

```
    updata(root);
    splay(root->ch[1], null);

    scanf("%d", &m);
    for (i = 1; i <= m; i++) {
        scanf(" %c", &c);
        if (c == 'I') {
            scanf("%d%d", &x, &y);
            insert(++x, y);

        }
        if (c == 'D') {
            scanf("%d", &x);
            del(++x);

        }
        if (c == 'R') {
            scanf("%d%d", &x, &y);
            replace(++x, y);

        }
        if (c == 'Q') {
            scanf("%d%d", &x, &y);
            select(++x - 1, null);
            select(++y + 1, root);
            printf("%d\n", root->ch[1]->ch[0]->mc);
        }
    }
    return 0;
}
```

Play with Chain

```
/*hdu-3487
    【题意】给出两个操作，要你对 1。。。n这个序列进行操作，最后输出序列
    【算法】splay tree
          对于CUT a b c，先把【a，b】这个区间的子树砍了，然后再旋转tree，最后再补回去
          对于FLIP a b，直接把【a，b】这个区间标记翻转就行了
          这里注意一下，翻转不能每次都直接递归到叶子进行，因为操作多了，其实做了很多无用
          功，因此添加一个标记，在输出序列的时候再处理就行了
          用了：笨小孩大牛 的模板，然后加上自己乱改。。。速度还不错
*/

#define SIZE 300010                         //树的容量
typedef int type;                       //v的类型
const int inf = 0x7fffffff;             //inf
#define key        (CH[ CH[r][1] ][0])     //关键树

//-------------------------下面开始节点域声明-----------------------------
////////////////////////////
int CH[SIZE][2], P[SIZE];               //0 左 1 右、父
type V[SIZE];                           //值
int len;                                //资源使用量
////////////////////////////
int CNT[SIZE];                              //本棵树的节点个数
bool ROT[SIZE];                             //是否翻转
////////////////////////////
//-------------------------下面开始操作节点-------------------------------
int newNode() {                             //新建一个节点（先从回收站里找）
    return len ++;
}
void initNode(int x, type v, int ch0, int ch1, int p) { //初始化每个节点
    V[x] = v;
    CH[x][0] = ch0;
    CH[x][1] = ch1;
    P[x] = p;

    CNT[x] = 1;
    ROT[x] = false;

    V[x] = v;
}
//-------------------------下面开始操作伸展树-----------------------------
void initTree() {                       //只需掉一次，清资源，设NULL节点
    len = 1;                            //从 1 开始，0 代表NULL
    initNode(0, inf, 0, 0, 0);
    CNT[0] = 0;
}
struct SplayTree {
    int r;    //根
    //-------------------------下面是基本函数---------------------------
    SplayTree(type lv, type rv) {    //创建一个带哨兵的空SplayTree，lv/rv赋值成任何其
实无所谓
        int a = newNode();
        int b = newNode();
        initNode(a, lv, 0, b, 0);
        initNode(b, rv, 0, 0, a);
        push_up(b);        push_up(a);
        r = a;
    }
    void rotate(int x, bool c) {     //将节点i，c=0 左旋转zag，c=1 右旋转zig
        int y = P[x];

        push_down(CH[x][0]);
        push_down(CH[x][1]);
        push_down(CH[y][c]);

        CH[y][!c] = CH[x][c];
        if(CH[x][c])  P[CH[x][c]] = y;
        CH[x][c] = y;

        P[x] = P[y];
        P[y] = x;

        if(P[x]) CH[ P[x] ][ CH[P[x]][1]==y ] = x;
        else     r = x;
        push_up(y);
    }
    void splay(int x, int f=0) {     //表示把结点 x 转到结点 f 的下面，默认为节点x成为
根
```

```
        int y;
        bool b1, b2;
        push_down(x);
        while((y=P[x]) != f) {       //用到了P[0]=0 来处理x=0 的情况
            if(P[y] == f) {
                rotate(x, CH[y][0] == x);
            } else {
                b1 = (y==CH[P[y]][0]);
                b2 = (x==CH[  y ][0]);
                rotate((b1^b2) ? x : y, b2);
                rotate(x, b1);
            }
        }
        push_up(x);
    }
    //选择第rank个节点，并且将这个节点splay到f下面，从 0 开始
    void select(int rank, int f=0) {
        if(rank >= CNT[r] || rank<0)     return;
        int i = r;
        while(true) {
            push_down(i);
            if(CNT[ CH[i][0] ] == rank) break;
            if(CNT[CH[i][0]] > rank) {
                i = CH[i][0];
            } else {
                rank -= CNT[CH[i][0]]+1;
                i = CH[i][1];
            }
        }
        splay(i, f);
    }
    void selectAll() {                      //选中全部有效区间(1、2 节点分别为左右哨兵)
        splay(1);
        splay(2, 1);
    }
    void interval(int begin, int end) {   //选中[begin, end)左闭右开的区间
        select(begin-1);
        select(end, r);
    }
    //--------------------下面开始insert和remove----------------------
    void insert(int n) {    //在root后面插入arr(似满二叉树), assume n>0 !
        int x = newNode();
        initNode(x, n, insertDfs(0, n-1,x), CH[r][1], r);

        if(CH[x][1])  P[ CH[x][1] ] = x;
        CH[r][1] = x;
        splay(x);
    }
    int insertDfs(int base, int n, int father) {
        if(n<=0)  return 0;
        int idx = n/2;
        int x = newNode();
        initNode(x, base+idx+1, insertDfs(base,idx,x),
insertDfs(base+idx+1,n-idx-1,x), father);
        push_up(x);
        return x;
    }
    int cutKeyTree() { //切断keyTree和父亲的联系而已，不将节点清除回收，并返回keyTree
        if(!key)  return 0;
        int res = key;
        key = P[res] = 0;  //清除
        splay( CH[r][1] );
        return res;
    }
    //--------------------下面开始push----------------------
    void push_up(int idx) {
        if(!idx)  return;
        //断言此处idx没有任何标记!!!!!!!!!!
        CNT[idx] = CNT[CH[idx][0]]+CNT[CH[idx][1]]+1;
    }
    void push_down(int idx) {
        if(!idx)  return;
        if(ROT[idx]) {
            ROT[idx] = false;
            if(CH[idx][0])     ROT[ CH[idx][0] ] ^= 1;
            if(CH[idx][1])     ROT[ CH[idx][1] ] ^= 1;
            swap(CH[idx][0], CH[idx][1]);
        }
    }
    //------------------------一些其他函数--------------------
    void rotateKeyTree() {                  //将关键树翻转
        if(!key)  return;
        ROT[key] ^= 1;
        splay(key);
    }
    //将现在的[start,end)区间取下，移植到新树的before节点前面
    void trans(int start, int end, int before) {
        if(start==end || before==start)  return;   //分别代表区间为空和无需移动

        interval(start, end);
        int kt = cutKeyTree();
        interval(before, before);
        key = kt;
        P[kt] = CH[r][1];
        splay(kt);
    }
    //------------------------打印函数--------------------

    bool first;

    void outputDFS(int idx, bool rotate = false) {
        if(!idx)  return;
        rotate ^= ROT[idx];
        outputDFS(CH[idx][0^rotate], rotate);
        if(!first)    printf(" ");
        first = false;
        printf("%d", V[idx]);
        outputDFS(CH[idx][1^rotate], rotate);
    }
    void outputKey() {
```

```
          first = true;
          outputDFS(key);
          printf("\n");
     }
};

int main() {
     int n, m;
     char cmd[20];
     int a, b, c;
     while(scanf("%d%d", &n, &m), n!=-1) {
          initTree();
          SplayTree sp(0,0);
          sp.insert(n);

          while(m --) {
               scanf("%s%d%d", cmd, &a, &b);
               if(*cmd=='C') {
                    scanf("%d", &c);
                    sp.trans(a, b+1, c+1);
               } else {
                    sp.interval(a, b+1);
                    sp.rotateKeyTree();
               }
          }
          sp.selectAll();
          sp.outputKey();
     }
     return 0;
}
```

Robotic Sort

```
/**
 * hdu-1890   Robotic Sort
 * 题意：给定数字序列S，需要对其进行排序，排序方法是题目给定的。假设进行到了第S[i]个数字，该位置有序后的数字是S[j]，则对[i,j]这区间的所有数字进行翻转操作，这样就是得Swap(S[i],S[j])。如果有多个相同值，取原序列中下标最小的。输出每次操作查找的j的位置。
 */

#define SIZE 100010
typedef pair<int,int> type; //v的类型
const int inf = 0x7fffffff;
#define key        (CH[ CH[r][1] ][0])     //关键树


////////////////////////////
int CH[SIZE][2], P[SIZE];   //0 左 1 右、父
type V[SIZE];               //值
int len;                    //资源使用量
////////////////////////////
int CNT[SIZE];                                  //本棵树的节点个数
bool ROT[SIZE];                                 //是否翻转
int MIN[SIZE];                                  //子树中最小的节点号码
////////////////////////////
```

```
void initTree() {                           //只需掉一次，清资源，设NULL节点
     len = 1;                               //从 1 开始，0 代表NULL
     CH[0][0] = CH[0][1] = P[0] = CNT[0] = 0;       //保持 0 节点的左右父一直为 0
}
void initNode(int x, type v, int ch0, int ch1, int p) { //初始化每个节点
     V[x] = v;
     CH[x][0] = ch0;
     CH[x][1] = ch1;
     P[x] = p;

     CNT[x] = 1;
     ROT[x] = false;
     MIN[x] = x;
}
struct SplayTree {
     int r;    //根
     //-------------------------下面是基本函数--------------------------
     SplayTree(type lv, type rv) {    //创建一个带哨兵的空SplayTree，
          int a = len ++;
          int b = len ++;

          initNode(a, lv, 0, b, 0);
          initNode(b, rv, 0, 0, a);

          push_up(b);        push_up(a);

          r = a;
     }
     void insert(type * arr, int n) { //在root后面插入arr, assume n>0 !
          int x = len ++;
          initNode(x, arr[n-1], insertDfs(arr,n-1,x), CH[r][1], r);

          if(CH[x][1])  P[ CH[x][1] ] = x;
          CH[r][1] = x;

          push_up(x);
          push_up(r);
     }
     int insertDfs(type * arr, int n, int father) {
          if(n<=0)  return 0;
          int idx = n/2;
          int x = len ++;
          initNode(x, arr[idx], insertDfs(arr,idx,x),
insertDfs(arr+idx+1,n-idx-1,x), father);
          push_up(x);
          return x;
     }
     void rotate(int x, bool c) {               //将节点i，c=0 左旋转，c=1 右旋转
          int y = P[x];
          CH[y][!c] = CH[x][c];
          if(CH[x][c])  P[CH[x][c]] = y;
          CH[x][c] = y;

          P[x] = P[y];
          P[y] = x;
```

```
        if(P[x]) CH[ P[x] ][ CH[P[x]][1]==y ] = x;
        else     r = x;
        push_up(y);
    }
    void splay(int x, int f=0) {     //表示把结点 x 转到结点 f 的下面，默认为节点x成为根
        int y;
        bool b1, b2;
        while((y=P[x]) != f) {       //用到了P[0]=0 来处理x=0 的情况
            push_down(P[y]);
            push_down(y);
            push_down(x);
            if(P[y] == f) {
                rotate(x, CH[y][0] == x);
            } else {
                b1 = (y==CH[P[y]][0]);
                b2 = (x==CH[  y ][0]);
                rotate((b1^b2) ? x : y, b2);
                rotate(x, b1);
            }
        }
        push_up(x);
    }
    //选择第rank个节点，并且将这个节点splay到f下面，从 0 开始
    int select(int rank, int f=0) {
        if(rank >= CNT[r] || rank<0)     return 0;
        int i = r;
        while(true) {
            push_down(i);
            if(CNT[ CH[i][0] ] == rank) break;
            if(CNT[CH[i][0]] > rank) {
                i = CH[i][0];
            } else {
                rank -= CNT[CH[i][0]]+1;
                i = CH[i][1];
            }
        }
        splay(i, f);
        return i;
    }
    //--------------------下面开始push----------------------
    void push_up(int idx) {
        CNT[idx] = CNT[CH[idx][0]]+CNT[CH[idx][1]]+1;

        MIN[idx] = idx;

        if(CH[idx][1] && V[ MIN[CH[idx][1]] ] < V[ MIN[idx] ])
            MIN[idx] = MIN[CH[idx][1]];
        if(CH[idx][0] && V[ MIN[CH[idx][0]] ] < V[ MIN[idx] ])
            MIN[idx] = MIN[CH[idx][0]];
    }
    void push_down(int idx) {
        if(!idx) return;
        if(ROT[idx]) {
            ROT[idx] = false;
            if(CH[idx][0])     ROT[ CH[idx][0] ] ^= 1;
            if(CH[idx][1])     ROT[ CH[idx][1] ] ^= 1;
            swap(CH[idx][0], CH[idx][1]);
        }
    }
    //-------------------------一些其他函数--------------------
    int getRank(int x) {                    //得到节点x的排名（即在这棵树中的位置，从 0 开始）
        splay(x);
        return CNT[CH[x][0]];
    }
    void rotateKeyTree() {                  //将关键树翻转
        ROT[key] ^= 1;
    }
};

type arr[SIZE];

int main() {
    int n;
    while(scanf("%d", &n), n) {
        initTree();
        SplayTree sp(type(0,0), type(inf,inf));    //插入哨兵，右面哨兵取最值
        for(int i = 0; i < n; i ++) {
            scanf("%d", &arr[i].first);
            arr[i].second = i+1;
        }
        sp.insert(arr, n);
        for(int i = 0; i < n; i ++) {
            sp.select(i);
            int idx = MIN[ CH[sp.r][1] ];    //最小的那个节点
            int rank = sp.getRank(idx);
            sp.select(i);
            sp.select(rank+1, sp.r);                    //选择出来了所选区间的边界
            sp.rotateKeyTree();

            if(i != 0)    printf(" ");
            printf("%d", rank);
        }
        printf("\n");
    }
    return 0;
}
```

SuperMemo

```
using namespace std;

//POJ-3580    SuperMemo
//对区间进行加数、翻转、平移、插入、删除操作，并且查询区间最小元素
//详细见main函数

#define SIZE 200010                         //树的容量
typedef int type;                       //v的类型
const int inf = 0x7fffffff;             //inf
```

```
#define key        (CH[ CH[r][1] ][0])     //关键树

//-------------------------下面开始节点域声明-----------------------------
//////////////////////////
int CH[SIZE][2], P[SIZE];              //0 左 1 右、父
type V[SIZE];                          //值
int len;                               //资源使用量
//////////////////////////
int CNT[SIZE];                              //本棵树的节点个数
bool ROT[SIZE];                             //是否翻转
//////////////////////////
int ADD[SIZE];                              //标记-是否加上ADD
int MIN[SIZE];                              //本树中最小的值
//////////////////////////
int laji;                              //垃圾回收
//////////////////////////
//-------------------------下面开始操作节点-------------------------------
int newNode() {                             //新建一个节点（先从回收站里找）
    if(laji) {
        int res = laji;
        laji = P[laji];
        return res;
    } else {
        return len ++;
    }
}
void removeNode(int x) {               //将这个节点放入回收站
    P[x] = laji;
    laji = x;
}
void initNode(int x, type v, int ch0, int ch1, int p) { //初始化每个节点
    V[x] = v;
    CH[x][0] = ch0;
    CH[x][1] = ch1;
    P[x] = p;

    CNT[x] = 1;
    ROT[x] = false;

    MIN[x] = V[x] = v;
    ADD[x] = 0;
}
//-------------------------下面开始操作伸展树-----------------------------
void initTree() {                      //只需掉一次，清资源，设NULL节点
    len = 1;                           //从 1 开始，0 代表NULL
    laji = 0;
    initNode(0, inf, 0, 0, 0);
    CNT[0] = 0;
}
struct SplayTree {
    int r;    //根
    //-------------------------下面是基本函数---------------------------
    SplayTree(type lv, type rv) {    //创建一个带哨兵的空SplayTree，lv/rv赋值成任何其实无所谓
        int a = newNode();
        int b = newNode();
        initNode(a, lv, 0, b, 0);
        initNode(b, rv, 0, 0, a);
        push_up(b);        push_up(a);
        r = a;
    }
    void rotate(int x, bool c) {     //将节点i，c=0 左旋转zag，c=1 右旋转zig
        int y = P[x];

        push_down(CH[x][0]);
        push_down(CH[x][1]);
        push_down(CH[y][c]);

        CH[y][!c] = CH[x][c];
        if(CH[x][c])  P[CH[x][c]] = y;
        CH[x][c] = y;

        P[x] = P[y];
        P[y] = x;

        if(P[x]) CH[ P[x] ][ CH[P[x]][1]==y ] = x;
        else     r = x;
        push_up(y);
    }
    void splay(int x, int f=0) {     //表示把结点 x 转到结点 f 的下面，默认为节点x成为根
        int y;
        bool b1, b2;
        push_down(x);
        while((y=P[x]) != f) {       //用到了P[0]=0 来处理x=0 的情况
            if(P[y] == f) {
                rotate(x, CH[y][0] == x);
            } else {
                b1 = (y==CH[P[y]][0]);
                b2 = (x==CH[  y ][0]);
                rotate((b1^b2) ? x : y, b2);
                rotate(x, b1);
            }
        }
        push_up(x);
    }
    //选择第rank个节点，并且将这个节点splay到f下面，从 0 开始
    void select(int rank, int f=0) {
        if(rank >= CNT[r] || rank<0)     return;
        int i = r;
        while(true) {
            push_down(i);
            if(CNT[ CH[i][0] ] == rank) break;
            if(CNT[CH[i][0]] > rank) {
                i = CH[i][0];
            } else {
                rank -= CNT[CH[i][0]]+1;
                i = CH[i][1];
```

```
            }
        }
        splay(i, f);
    }
    void interval(int begin, int end) {   //选中[begin, end)左闭右开的区间
        select(begin-1);
        select(end, r);
    }
    //--------------------下面开始insert和remove----------------------
    void insert(type * arr, int n) { //在root后面插入arr(似满二叉树), assume n>0 !
        int x = newNode();
        initNode(x, arr[n-1], insertDfs(arr,n-1,x), CH[r][1], r);

        if(CH[x][1])  P[ CH[x][1] ] = x;
        CH[r][1] = x;
        splay(x);
    }
    int insertDfs(type * arr, int n, int father) {
        if(n<=0)  return 0;
        int idx = n/2;
        int x = newNode();
        initNode(x, arr[idx], insertDfs(arr,idx,x),
insertDfs(arr+idx+1,n-idx-1,x), father);
        push_up(x);
        return x;
    }
    int cutKeyTree() { //切断keyTree和父亲的联系而已，不将节点清除回收，并返回keyTree
        if(!key)  return 0;
        int res = key;
        key = P[res] = 0;  //清除
        splay( CH[r][1] );
        return res;
    }
    void remove(int x) {    //将以x为根的树全部清除回收!
        if(!x)   return;
        remove(CH[x][0]);
        remove(CH[x][1]);
        removeNode(x);
    }
    //--------------------下面开始push----------------------
    void push_up(int idx) {
        if(!idx)  return;
        //断言此处idx没有任何标记!!!!!!!!!!

        CNT[idx] = CNT[CH[idx][0]]+CNT[CH[idx][1]]+1;
        MIN[idx] = min(MIN[CH[idx][0]], MIN[CH[idx][1]]);
        MIN[idx] = min(MIN[idx], V[idx]);
    }
    void push_down(int idx) {
        if(!idx)  return;
        if(ROT[idx]) {
            ROT[idx] = false;
            if(CH[idx][0])     ROT[ CH[idx][0] ] ^= 1;
            if(CH[idx][1])     ROT[ CH[idx][1] ] ^= 1;
            swap(CH[idx][0], CH[idx][1]);
        }
        if(ADD[idx]) {
            if(CH[idx][0])     ADD[CH[idx][0]]+=ADD[idx];
            if(CH[idx][1])     ADD[CH[idx][1]]+=ADD[idx];
            V[idx] += ADD[idx];
            MIN[idx] += ADD[idx];
            ADD[idx] = 0;
        }
    }
    //------------------------一些其他函数--------------------
    void rotateKeyTree() {                //将关键树翻转
        if(!key)  return;
        ROT[key] ^= 1;
        splay(key);
    }
    void addKeyTree(int v) {              //将关键树每个节点都加上v
        if(!key)  return;
        ADD[key] += v;
        splay(key);
    }
    //将现在的[start,end)区间取下，移植到新树的before节点前面
    void trans(int start, int end, int before) {
        if(start==end || before==start)  return;   //分别代表区间为空和无需移动

        interval(start, end);
        int kt = cutKeyTree();
        interval(before, before);
        key = kt;
        P[kt] = CH[r][1];
        splay(kt);
    }
    //-----------------------打印函数--------------------
    void outputKeyMIN() {
        printf("%d\n", MIN[key]);
    }
};

int arr[SIZE];

int main() {
    int n, q;
    char cmd[20];
    int a, b, c;
    scanf("%d", &n);
    initTree();
    SplayTree sp(0,0);
    for(int i = 0; i < n; i ++) scanf("%d", arr+i);
    sp.insert(arr, n);

    scanf("%d", &q);
    while(q --) {
        scanf("%s", cmd);
        if(strcmp(cmd,"ADD")==0) {
//ADD x y D: Add D to each number in sub-sequence {Ax ... Ay}. For example, performing
"ADD 2 4 1" on {1, 2, 3, 4, 5} results in {1, 3, 4, 5, 5}
```

```
            scanf("%d %d %d", &a, &b, &c);
            sp.interval(a, b+1);
            sp.addKeyTree(c);
        } else if(strcmp(cmd,"REVERSE")==0) {
//REVERSE x y: reverse the sub-sequence {Ax ... Ay}. For example, performing "REVERSE
2 4" on {1, 2, 3, 4, 5} results in {1, 4, 3, 2, 5}
            scanf("%d %d", &a, &b);
            sp.interval(a, b+1);
            sp.rotateKeyTree();
        } else if(strcmp(cmd, "REVOLVE")==0) {
//REVOLVE x y T: rotate sub-sequence {Ax ... Ay} T times. For example, performing
"REVOLVE 2 4 2" on {1, 2, 3, 4, 5} results in {1, 3, 4, 2, 5}
            scanf("%d %d %d", &a, &b, &c);
            c %= b-a+1;
            if(c<0)  c += b-a+1;
            sp.trans(b+1-c, b+1, a);
        } else if(strcmp(cmd, "INSERT")==0) {
//INSERT x P: insert P after Ax. For example, performing "INSERT 2 4" on {1, 2, 3,
4, 5} results in {1, 2, 4, 3, 4, 5}
            scanf("%d %d", &a, &b);
            sp.select(a);
            sp.insert(&b, 1);
        } else if(strcmp(cmd, "DELETE")==0) {
//DELETE x: delete Ax. For example, performing "DELETE 2" on {1, 2, 3, 4, 5} results
in {1, 3, 4, 5}
            scanf("%d", &a);
            sp.interval(a, a+1);
            sp.remove(sp.cutKeyTree());
        } else if(strcmp(cmd, "MIN")==0) {
//MIN x y: query the participant what is the minimum number in sub-sequence {Ax ...
Ay}. For example, the correct answer to "MIN 2 4" on {1, 2, 3, 4, 5} is 2
            scanf("%d %d", &a, &b);
            sp.interval(a, b+1);
            sp.outputKeyMIN();
        }
    }
    return 0;
}
```

宠物收养所

```
using namespace std;

#define SIZE 80010

int CH[SIZE][2], P[SIZE], V[SIZE];          //0 左 1 右、父、值
int len;                                    //资源使用量


void initTree() {                           //只需掉一次，清资源，设NULL节点
    len = 1;                                //从 1 开始，0 代表NULL
    CH[0][0] = CH[0][1] = P[0] = 0;         //保持 0 节点的左右父一直为 0
}
struct SplayTree {
    int r;   //根
    SplayTree(int r = 0) {
        this->r = r;                        //0 代表NULL
    }
    void rotate(int x, bool c) {                //将节点i，c=0 左旋转，c=1 右旋转
        int y = P[x];
        CH[y][!c] = CH[x][c];
        if(CH[x][c])  P[CH[x][c]] = y;
        CH[x][c] = y;

        P[x] = P[y];
        P[y] = x;

        if(P[x]) CH[ P[x] ][ CH[P[x]][1]==y ] = x;
        else     r = x;
    }
    void splay(int x, int f=0) {     //表示把结点 x 转到结点 f 的下面，默认为节点x成为
根

        int y;
        bool b1, b2;
        for(; (y=P[x]) != f; ) {          //用到了P[0]=0 来处理x=0 的情况
            if(P[y] == f) {
                rotate(x, CH[y][0] == x);
            } else {
                b1 = (y==CH[P[y]][0]);
                b2 = (x==CH[  y ][0]);
                rotate((b1^b2) ? x : y, b2);
                rotate(x, b1);
            }
        }
    }
    int insert(int v) {     //插入一个元素，返回这个元素的编号
        int i = len ++, *j = &r;
        CH[i][0] = CH[i][1] = P[i] = 0;
        V[i] = v;
        while(*j)
            P[i]=*j ,j = CH[*j] + (v>=V[*j]);
        splay(*j=i);
        return i;
    }
    int lower_bound(int v) {
        int i = r;
        int tgt = 0;
        while(i) {
            if(V[i] < v) {
                i = CH[i][1];
            } else {
                tgt = i;
                i = CH[i][0];
            }
        }
        return tgt;
    }
    void join(SplayTree b) {    //假设我中每个元素都小于b
        if(r == 0 || b.r == 0) {
            r = r + b.r;
            return;
        }
```

```
        int i;
        for(i = r; CH[i][1] != 0; i = CH[i][1]);
        splay(i);
        CH[r][1] = b.r;    //now r=i
        P[b.r] = r;
    }
    void remove(int x) {         //删除某个节点
        splay(x);
        P[ r=CH[x][0] ] = 0;
        P[CH[x][1]] = 0;
        join(SplayTree(CH[x][1]));
    }
    int to(int idx, int dir) {  //从节点x走，一直沿着dir走到头，dir=0 或 1
        while(CH[idx][dir])     idx = CH[idx][dir];
        return idx;
    }
};
/**
 * hnoi-1208  宠物收养所
Description:
    收养所提供两种服务：收养被主人遗弃的宠物和让新的主人领养这些宠物。
    每个领养者都希望领养到自己满意的宠物，阿Q根据领养者的要求通过他自己发明的一个特殊的公式，得出该领养者希望领养的宠物的特点值a（a是一个 正整数，a<2^31），而他也给每个处在收养所的宠物一个特点值。这样他就能够很方便的处理整个领养宠物的过程了，宠物收养所总是会有两种情况发 生：被遗弃的宠物过多或者是想要收养宠物的人太多，而宠物太少。
    1. 被遗弃的宠物过多时，假若到来一个领养者，这个领养者希望领养的宠物的特点值为a，那么它将会领养一只目前未被领养的宠物中特点值最接近a的一只宠 物。(任何两只宠物的特点值都不可能是相同的，任何两个领养者的希望领养宠物的特点值也不可能是一样的）如果有两只满足要求的宠物，即存在两只宠物他们的特点值分别为a-b和a+b，那么领养者将会领养特点值为a-b的那只宠物。
    2. 收养宠物的人过多，假若到来一只被收养的宠物，那么哪个领养者能够领养它呢？能够领养它的领养者，是那个希望被领养宠物的特点值最接近该宠物特点值的领养 者，如果该宠物的特点值为a，存在两个领养者他们希望领养宠物的特点值分别为a-b和a+b，那么特点值为a-b的那个领养者将成功领养该宠物。

    一个领养者领养了一个特点值为a的宠物，而它本身希望领养的宠物的特点值为b，那么这个领养者的不满意程度为abs(a-b)。
【任务描述】
    你得到了一年当中，领养者和被收养宠物到来收养所的情况，希望你计算所有收养了宠物的领养者的不满意程度的总和。这一年初始时，收养所里面既没有宠物，也没有领养者。
Input
    第一行为一个正整数n，n<=80000，表示一年 当中来到收养所的宠物和领养者的总数。接下来的n行，按到来时间的先后顺序描述了一年当中来到收养所的宠物和领养者的情况。每行有两个正整数a, b，其中a=0表示宠物，a=1 表示领养者，b表示宠物的特点值或是领养者希望领养宠物的特点值。(同一时间呆在收养所中的，要么全是宠物，要么全是领养 者，这些宠物和领养者的个数不会超过 10000 个)
Output
    仅有一个正整数,表示一年当中所有收养了宠物的领养者的不满意程度的总和mod 1000000 以后的结果。
 */

int main() {
    int n, a, b, now;
    while(scanf("%d", &n) != EOF) {
        initTree();
        SplayTree sp;
        int ans = 0, size = 0;  //答案，树的节点个数
        while(n --) {
            scanf("%d%d", &a, &b);
            if(size==0 || now==a) {
                sp.insert(b);
                now = a;
                size ++;
            } else {
                int idx = sp.lower_bound( b ), tmp;
                if(idx == 0) {     //没有比我大的!
                    idx = sp.to(sp.r, 1);   //最大的值
                    tmp = abs(V[idx]-b);
                    sp.remove(idx);
                } else {
                    sp.splay(idx);

                    int idxL = sp.to(CH[idx][0], 1);
                    if(idxL && abs(V[idxL]-b)<=abs(V[idx]-b)) {
                        idx = idxL;
                    }
                    tmp = abs(V[idx]-b);
                    sp.remove(idx);
                }
                ans = (ans + tmp) % 1000000;
                size --;
            }
        }
        printf("%d\n", ans);
    }
    return 0;
}
```

维修数列

```
using namespace std;

//noi-2005 维护数列(hnoi-1500)
//请写一个程序，要求维护一个数列，支持以下 6 种操作：(请注意，格式栏中的下划线丶 _ ’表示实际输入文件中的空格)【见main函数】

/**
 * 注释：
        0.伸展树从的rank从 0 开始计算，但是 0 和n+1 都是哨兵，因此有效数列的rank从 1 开始计算(恰好吻合，易于编程)
        1.节点分为三种：空节点(0)、哨兵(1,2)、普通节点(>2)
        2.哨兵是无所谓的哨兵，他们有值，但无论取什么值都不会影响到结果，因为输出结果时，应该将哨兵放在左右两端，这样选出的key则为全部的值。因此，没必要特意为哨兵赋值
            另外，哨兵不该被标记，然后改变什么。受改变的应该是普通的节点!
        3.空节点是受到保护的，其P、CH、V以及其他域不该受到改变，无论是splay还是push或者其他的，都不改改变过空节点的任何值
            但是，为了简便起见，会让空节点赋给一些特殊的值，比如空节点的孩子、父亲为空。
            其他的域自己写填写，主要是在push中，其他域会被调用，充当哨兵的作用。【【空节点是唯一一个需要赋特殊值的哨兵!!!】】
            比如，在本题中，空节点的SUM赋值为 0，LL、RR、MM、V、赋值为-inf。这样在push_up中会简便代码。。。。(由于空节点的域受保护，不必担心他们以后会被更改!)
```

如果空节点不赋特殊值也可以，但是需要增加代码两。。。空节点应该在initTree中赋值

4.push_down操作会处理各种标记，push_up操作维护了变量向上传播，【在push_up的时候，该节点的各种标记应该都没了(即已经被push_down了)assume!!】

5.不要写新的函数都要调用push，如果某一个节点的域发生了改变，首先先到的应该是splay，push操作尽量都写在splay、rotate、select等函数里，其他地方尽量不要写!

*/

```
#define SIZE 500010                         //树的容量
typedef int type;                           //V的类型
const int inf = 0x3f3f3f3f;                 //inf(其实inf只要做到 1001 就行了,因为原始数列每个数字都小于等于 1000)
#define key        (CH[ CH[r][1] ][0])      //关键树

//-------------------------下面开始节点域声明-----------------------------
////////////////////////////
int CH[SIZE][2], P[SIZE];                   //0 左 1 右、父
type V[SIZE];                               //值
int len;                                    //资源使用量
////////////////////////////
int CNT[SIZE];                              //本棵树的节点个数
bool ROT[SIZE];                             //是否翻转
////////////////////////////
int SUM[SIZE];                              //本树V的和
int LL[SIZE], RR[SIZE], MM[SIZE];           //从左面连续的值的和，右面连续的值的和，连续的最大值的和

int SAME[SIZE];                             //标记-是否变为相同的
////////////////////////////
int laji;                                   //垃圾回收
////////////////////////////
//-------------------------下面开始操作节点-------------------------------
int newNode() {                             //新建一个节点（先从回收站里找）
    if(laji) {
        int res = laji;
        laji = P[laji];
        return res;
    } else {
        return len ++;
    }
}
void removeNode(int x) {                    //将这个节点放入回收站
    P[x] = laji;
    laji = x;
}
void initNode(int x, type v, int ch0, int ch1, int p) { //初始化每个节点
    V[x] = v;
    CH[x][0] = ch0;
    CH[x][1] = ch1;
    P[x] = p;

    CNT[x] = 1;
    ROT[x] = false;
    SUM[x] = V[x];
    MM[x] = LL[x] = RR[x] = V[x];//max(V[x], 0);
    SAME[x] = inf;
}
//-------------------------下面开始操作伸展树-----------------------------
void initTree() {                           //只需掉一次，清资源，设NULL节点
    len = 1;                                //从 1 开始，0 代表NULL
    laji = 0;
    initNode(0, -inf, 0, 0, 0);
    CNT[0] = SUM[0] = 0;
}
struct SplayTree {
    int r;    //根
    //-------------------------下面是基本函数-----------------------------
    SplayTree(type lv, type rv) {           //创建一个带哨兵的空SplayTree，lv/rv赋值成任何其实无所谓
        int a = newNode();
        int b = newNode();
        initNode(a, lv, 0, b, 0);
        initNode(b, rv, 0, 0, a);
        push_up(b);         push_up(a);
        r = a;
    }
    void rotate(int x, bool c) {            //将节点i，c=0 左旋转zag，c=1 右旋转zig
        int y = P[x];

        push_down(CH[x][0]);
        push_down(CH[x][1]);
        push_down(CH[y][c]);

        CH[y][!c] = CH[x][c];
        if(CH[x][c])  P[CH[x][c]] = y;
        CH[x][c] = y;

        P[x] = P[y];
        P[y] = x;

        if(P[x]) CH[ P[x] ][ CH[P[x]][1]==y ] = x;
        else     r = x;
        push_up(y);
    }
    void splay(int x, int f=0) {            //表示把结点 x 转到结点 f 的下面，默认为节点x成为根
        int y;
        bool b1, b2;
        push_down(x);
        while((y=P[x]) != f) {              //用到了P[0]=0 来处理x=0 的情况
            if(P[y] == f) {
                rotate(x, CH[y][0] == x);
            } else {
                b1 = (y==CH[P[y]][0]);
                b2 = (x==CH[  y ][0]);
                rotate((b1^b2) ? x : y, b2);
                rotate(x, b1);
```

```
            }
        }
        push_up(x);
    }
    //选择第rank个节点，并且将这个节点splay到f下面，从 0 开始
    void select(int rank, int f=0) {
        if(rank >= CNT[r] || rank<0)     return;
        int i = r;
        while(true) {
            push_down(i);
            if(CNT[ CH[i][0] ] == rank) break;
            if(CNT[CH[i][0]] > rank) {
                i = CH[i][0];
            } else {
                rank -= CNT[CH[i][0]]+1;
                i = CH[i][1];
            }
        }
        splay(i, f);
    }
    void interval(int begin, int end) {   //选中[begin, end)左闭右开的区间
        select(begin-1);
        select(end, r);
    }
    void selectAll() {                    //选中全部有效区间(1、2 节点分别为左右哨兵)
        splay(1);
        splay(2, 1);
    }
    //--------------------下面开始insert和remove----------------------
    void insert(type * arr, int n) { //在root后面插入arr(似满二叉树), assume n>0 !
        int x = newNode();
        initNode(x, arr[n-1], insertDfs(arr,n-1,x), CH[r][1], r);

        if(CH[x][1])  P[ CH[x][1] ] = x;
        CH[r][1] = x;
        splay(x);
    }
    int insertDfs(type * arr, int n, int father) {
        if(n<=0) return 0;
        int idx = n/2;
        int x = newNode();
        initNode(x, arr[idx], insertDfs(arr,idx,x),
insertDfs(arr+idx+1,n-idx-1,x), father);
        push_up(x);
        return x;
    }
    int cutKeyTree() { //切断keyTree和父亲的联系而已，不将节点清除回收，并返回keyTree
        if(!key) return 0;
        int res = key;
        key = P[res] = 0;  //清除
        splay( CH[r][1] );
        return res;
    }
    void remove(int x) {    //将以x为根的树全部清除回收!
        if(!x)    return;
        remove(CH[x][0]);
        remove(CH[x][1]);
        removeNode(x);
    }
    //--------------------下面开始push----------------------
    void push_up(int idx) {
        if(!idx) return;
        //断言此处idx没有任何标记!!!!!!!!!
        CNT[idx] = CNT[CH[idx][0]]+CNT[CH[idx][1]]+1;
        SUM[idx] = SUM[CH[idx][0]]+SUM[CH[idx][1]]+V[idx];//[]

        LL[idx] = max(LL[CH[idx][0]], SUM[CH[idx][0]]+V[idx] +
max(LL[CH[idx][1]],0)  );
        RR[idx] = max(RR[CH[idx][1]], SUM[CH[idx][1]]+V[idx] +
max(RR[CH[idx][0]],0)  );
        MM[idx] = max(MM[CH[idx][0]], MM[CH[idx][1]]);
        MM[idx] = max(MM[idx], V[idx] + max(0,RR[CH[idx][0]]) +
max(0,LL[CH[idx][1]]) );
    }
    void push_down(int idx) {
        if(!idx) return;
        if(ROT[idx]) {
            ROT[idx] = false;
            if(CH[idx][0])     ROT[ CH[idx][0] ] ^= 1;
            if(CH[idx][1])     ROT[ CH[idx][1] ] ^= 1;
            swap(CH[idx][0], CH[idx][1]);
            swap(LL[idx], RR[idx]);
        }
        if(SAME[idx] != inf) {
            V[idx] = SAME[idx];

            if(CH[idx][0])     SAME[ CH[idx][0] ]=SAME[idx];
            if(CH[idx][1])     SAME[ CH[idx][1] ]=SAME[idx];

            SUM[idx] = V[idx]*CNT[idx];

            if(SAME[idx]>0) {
                MM[idx] = LL[idx] = RR[idx] = V[idx]*CNT[idx];
            } else {
                MM[idx] = LL[idx] = RR[idx] = V[idx];
            }
            SAME[idx] = inf;
        }
    }
    //------------------------一些其他函数-------------------
    void rotateKeyTree() {                //将关键树翻转
        if(!key) return;
        ROT[key] ^= 1;
        splay(key);
    }
    void sameKeyTree(int v) {             //将关键树值都改为v
        if(!key) return;
        SAME[key] = v;
        splay(key);
    }
```

```
    //------------------以下是打印代码------------------------
    void outputKeySum() {
        printf("%d\n", SUM[key]);
    }
    void outputKeyMM() {
        printf("%d\n", MM[key]);
    }
};

int arr[SIZE];

int main() {
    int n, m;
    while(scanf("%d%d", &n, &m) != EOF) {
        initTree();
        SplayTree sp(0, 0);          //无所谓的哨兵

        for(int i = 0; i < n; i ++) scanf("%d", arr+i);
        sp.insert(arr, n);

        char cmd[100];
        int a, b, c;

        while(m --) {
            scanf("%s", cmd);
            if(*cmd=='G') {
//求和(GET-SUM_posi_tot)计算从当前数列开始的第 posi 个数字,开始的tot个数字的和并输出
                scanf("%d%d", &a, &b);
                sp.interval(a, a+b);
                sp.outputKeySum();
            } else if(cmd[2]=='X') {
//求和最大的子列(MAX-SUM)求出当前数列中和最大的一段子列，并输出最大和
                sp.selectAll();
                sp.outputKeyMM();
            } else if(*cmd == 'I') {
//插入(INSERT_posi_tot_c1_c2_..._ctot)在当前数列的第posi个数字后插入tot个数字：c1, c2, …, ctot；若在数列首插
                scanf("%d%d", &a, &b);
                for(int i = 0; i < b; i ++) scanf("%d", arr+i);
                sp.select(a);
                sp.insert(arr, b);
            } else if(*cmd == 'D') {
//删除(DELETE_posi_tot)从当前数列的第 posi 个数字开始连续删除tot个数字
                scanf("%d%d", &a, &b);
                sp.interval(a, a+b);
                sp.remove( sp.cutKeyTree() );
            } else if(cmd[2]=='K') {
//修改(MAKE-SAME_posi_tot_c)将当前数列的第 posi 个数字开始的连续tot个数字统一修改为c
                scanf("%d%d%d", &a, &b, &c);
                sp.interval(a, a+b);
                sp.sameKeyTree(c);
            } else if(*cmd=='R') {
//翻转(REVERSE_posi_tot)取出从当前数列的第 posi 个数字开始的tot个数字，翻转后放入原来的位置
                scanf("%d%d", &a, &b);
                sp.interval(a, a+b);
                sp.rotateKeyTree();
            }
        }
    }
    return 0;
}
```

维修数列_beyond the void

```
/*
 * Problem: NOI2005 sequence
 * Author: Guo Jiabao
 * Time: 2009.5.30 14:19
 * State: Solved
 * Memo: 伸展树
*/

/*
这是NOI有史以来出过的最BT的数据结构题了，不少人写的都是块状链表，我来个Splay的（比起编程难度，其实朴素才是王道）。
建立一棵Splay，每个节点上要维护一下几个信息：数值value，子树大小size，子树和sum，子数内和最大的子数列 maxsum，子树区间内由左边界能够延伸的和最大子数列mls，子树区间内由右边界能够延伸的和最大子数列mrs，以及两个标记：子树反转标记rev，子树同化标记same。为了避免判断边界条件，首先插入两个节点，作为开头和结尾，其永久排名分别为 1 和当前数列长度。
插入和删除操作和NOI2003 的文本编辑器类似。插入一段数列，把第pos+1 个元素splay到根节点，，把第pos+2 个元素 splay到根节点的右子树根节点，把要插入的这段数列建成一条链最好，然后把这个新建的链接到根节点右子树的左子树上，最后把链的尾splay到根节点。自底向上splay过程中要维护各项信息。
删除就是把第pos个元素splay到根节点，把第pos+tot+1 个元素splay到根节点的右子树根节点，然后把根节点右子树的左子树直接砍掉就行了。由于数据规模比较大，如果内存比较紧张，要考虑回收空间。
考虑到修改和翻转的区间可能很大，暴力的肯定不行。取得区间的方法和删除一样，然后我们就用标记标识代表这个区间的这棵子树，今后访问到这这棵子树时，处理标记然后标记下传。具体来说，下传rev标记时，交换两个子树以及mls和mrs的值，然后两棵子树获得标记。下传same标记时，先把当前节点值传给子节点，维护节点的sum值为当前节点的value * size。然后如果当前节点值为非负数，令maxsum，mls，mrs的值为value * size，如果当前节点值为负数，令maxsum，mls，mrs的值为p->value。要注意的是，任何时候只要访问到带标记的节点，一定要标记下传，尤其不要忘了在splay旋转的时候。
另一个比较复杂的地方为自底向上维护节点信息，我们有size，sum，mls，mrs，maxsum五个信息需要同时维护。其中size，sum比较简单。
mls的取值有三种可能，直接继承与左子树的mls值，整个左子树的sum + 当前节点value，以及左子树的sum + 当前节点value + 右子树的mls，取三者最大值。mrs类似于mls。
maxsum的维护更为复杂，可能直接为当前节点value，继承左子树的maxsum，继承右子树的maxsum，左子树的mrs + 当前节点value，右子树的mls + 当前节点value，以及左子树的mrs + 右子树的mls + 当前节点value，这五种情况，同样取最大值。
有了上述的维护，求一段区间和也就不难了。同样的方法找到待求和区间，直接读取子树根节点的sum值即可。求和最大的子数列直接读取根节点的maxsum就行了。
看了zkw神牛的 《BST拓展与伸展树(splay)一日通》，没学会自顶向下的伸展树，倒学了不少代码技巧。例如建一个null节点，方便自底向上维护节点信息，还有空树不好操作，那就插入两个节点，作为开头和结尾。于是我的代码仅仅 245 行（4.89K），比起许多块状链表简单多了。
*/

const int MAXN=500010,MAXL=500001,INF=1001;
struct SplayTree
{
    struct SplayNode
```

```
{
     SplayNode *c[2],*f;
     int value,size,sum,maxsum,mls,mrs;
     bool rev,same;
}*root,*null,*lb,*rb,SS[MAXN], *rubbish;
int SC;
SplayNode * NewNode(int value,SplayNode *f)
{
     SplayNode *e;
     if(rubbish!=NULL) {
          e = rubbish;
          rubbish = rubbish->f;
     } else {
          e = SS+ ++SC;
     }
     e->value=value;
     e->size=1;e->f=f;
     e->sum=e->maxsum=e->mls=e->mrs=value;
     e->same=e->rev=false;
     e->c[0]=e->c[1]=null;
     return e;
}

void removeNode(SplayNode * e) {
     e->f = rubbish;
     rubbish = e;
}

inline int max(int a,int b){return a>b?a:b;}
void update(SplayNode *p)
{
     if (p==null) return;
     p->size = p->c[0]->size + p->c[1]->size + 1;
     p->sum = p->c[0]->sum + p->c[1]->sum + p->value;


     p->mls = p->c[0]->mls;
     p->mls = max( p->mls , p->c[0]->sum + p->value);
     p->mls = max( p->mls , p->c[0]->sum + p->value + p->c[1]->mls );



     p->mrs = p->c[1]->mrs;
     p->mrs = max( p->mrs , p->c[1]->sum + p->value);
     p->mrs = max( p->mrs , p->c[1]->sum + p->value + p->c[0]->mrs );



     p->maxsum = p->value;
     p->maxsum = max( p->maxsum , p->c[0]->maxsum );
     p->maxsum = max( p->maxsum , p->c[1]->maxsum );

     p->maxsum = max( p->maxsum , p->c[0]->mrs + p->value );
     p->maxsum = max( p->maxsum , p->c[1]->mls + p->value );
     p->maxsum = max( p->maxsum , p->c[0]->mrs + p->c[1]->mls + p->value );
}
```

```
     void pushdown(SplayNode *p)
     //----------------------------checked !
     {
          if (p==null) return;
          if (p->rev)
          {
               p->rev=false;
               SplayNode *q=p->c[0]; p->c[0]=p->c[1]; p->c[1]=q;
               p->c[0]->rev = !p->c[0]->rev;
               p->c[1]->rev = !p->c[1]->rev;
               int t=p->mls;
               p->mls=p->mrs; p->mrs=t;
          }
          if (p->same)
          {
               p->same=false;
               p->c[0]->same=p->c[1]->same=true;



               p->c[0]->value=p->c[1]->value=p->value;



               p->sum = p->maxsum = p->mls = p->mrs = p->value * p->size;
               if (p->value < 0)
                    p->maxsum = p->mls = p->mrs = p->value;
          }
     }
     void rotate(SplayNode *x,int o)//Zig o=0 Zag o=1------------------------check
over!
     {
          SplayNode *y=x->f;
          pushdown(x->c[0]);
          pushdown(x->c[1]);
          pushdown(y->c[!o]);

          y->c[o] = x->c[!o];
          y->c[o]->f=y;
          x->f = y->f;
          if (y->f->c[0]==y)
               y->f->c[0]=x;
          else
               y->f->c[1]=x;
          y->f=x;
          x->c[!o]=y;
          update(y);
          if (root==y) root=x;
     }
     void splay(SplayNode *x,SplayNode *y)//------------------------check over!
     {
          pushdown(x);
          while (x->f!=y)
          {
               if (x->f->f==y)
               {
```

```
            if (x->f->c[0]==x)
                rotate(x,0);
            else
                rotate(x,1);
        }
        else if (x->f->f->c[0] == x->f)
        {
            if (x->f->c[0]==x)
                rotate(x->f,0),rotate(x,0);
            else
                rotate(x,1),rotate(x,0);
        }
        else
        {
            if (x->f->c[1]==x)
                rotate(x->f,1),rotate(x,1);
            else
                rotate(x,0),rotate(x,1);
        }
    }
    update(x);
}
void select(int k,SplayNode *y)//------------------------check over!
{
    SplayNode *x=root;

    while(true) {
        pushdown(x);
        if(k == x->c[0]->size + 1)  break;
        if (k <= x->c[0]->size)
            x=x->c[0];
        else
        {
            k-=x->c[0]->size + 1;
            x=x->c[1];
        }
    }
    splay(x,y);
}
void Insert(int pos,int tot,int *C)            //assume checked!
{
    SplayNode *z,*t;
    z=t=NewNode(C[1],null);
    for (int i=2;i<=tot;i++)
        z=z->c[1]=NewNode(C[i],z);
    select(pos+1,null);
    select(pos+2,root);
    root->c[1]->c[0] = t;
    t->f=root->c[1];
    splay(z,null);
}

void deleteDFS(SplayNode * e) {                //---------------check over!
    if(e == null) return;
    deleteDFS(e->c[0]);
    deleteDFS(e->c[1]);
        removeNode(e);
    }
    void Delete(int pos,int tot)                    //---------------check over!
    {
        select(pos,null);
        select(pos+tot+1,root);

        deleteDFS(root->c[1]->c[0]);

        root->c[1]->c[0] = null;
        splay(root->c[1],null);
    }
    void MakeSame(int pos,int tot,int value)  //--------------check over!
    {
        select(pos,null);
        select(pos+tot+1,root);
        root->c[1]->c[0]->same=true;
        root->c[1]->c[0]->value=value;
        splay(root->c[1]->c[0],null);
    }
    void Reverse(int pos,int tot)    //--------------------------check over!
    {
        select(pos,null);
        select(pos+tot+1,root);
        root->c[1]->c[0]->rev=!root->c[1]->c[0]->rev;
        splay(root->c[1]->c[0],null);
    }
    int GetSum(int pos,int tot)      //--------------------------check over!
    {
        select(pos,null);
        select(pos+tot+1,root);
        return root->c[1]->c[0]->sum;
    }
    int MaxSum()                     //--------------------------check over!
    {
        return root->maxsum;
    }
    void init()                            //--------------------------checked!
    {
        rubbish = NULL;
        SC=-1;
        null=0;
        null=NewNode(-INF,0);
        null->size=0;
        lb=root=NewNode(-INF,null);
        rb=root->c[1]=NewNode(-INF,root);
        null->sum = lb->sum = rb->sum=0;
        update(root);
    }
}Splay;
int N,M,C[MAXL],pos,i,j,A;
char Ctrl[20];
int main()
{
    Splay.init();
    scanf("%d%d",&N,&M);
```

```
    for (i=1;i<=N;i++)
        scanf("%d",&C[i]);
    Splay.Insert(0,N,C);
    for (i=1;i<=M;i++)
    {
        scanf("%s",Ctrl);
        switch (Ctrl[0])
        {
            case 'I':
                scanf("%d%d",&pos,&N);
                for (j=1;j<=N;j++)
                    scanf("%d",&C[j]);
                Splay.Insert(pos,N,C);
                break;
            case 'D':
                scanf("%d%d",&pos,&N);
                Splay.Delete(pos,N);
                break;
            case 'R':
                scanf("%d%d",&pos,&N);
                Splay.Reverse(pos,N);
                break;
            case 'G':
                scanf("%d%d",&pos,&N);
                A=Splay.GetSum(pos,N);
                printf("%d\n",A);
                break;
            case 'M':
                if (Ctrl[2]=='K')
                {
                    scanf("%d%d%d",&pos,&N,&C[0]);
                    Splay.MakeSame(pos,N,C[0]);
                }
                else
                    printf("%d\n",Splay.MaxSum());
                break;
        }
    }
    if(scanf("%d%d",&N,&M)!=EOF)    while(1);
    return 0;
}
```

营业额统计

```
//hnoi-1588    营业额统计
//给定一段序列，求每一个数与其前面的数的差的最小值之和。
//第一个数与前面的数的差就是第一个数字。其他数字必须和前面存在的数字做差

#define maxn 80010
typedef int type;

struct Nod {
    Nod * ch[2], *p;   //孩子/父节点
    type val;          //值
    void init(Nod * ch0, Nod * ch1, Nod * pre, type val) {
        ch[0] = ch0;  ch[1] = ch1;  this->p = pre;    this->val = val;
    }
};

Nod * null = new Nod;        //空节点
Nod buf[maxn];     int len;  //资源/使用量

void initTree() {            //初始只需调用一次
    null->init(null, null, null, 0);
    len = 0;
}

Nod * newNod() {             //新建一个节点
    return buf + len++;
}

//--------------------下面开始伸展树------------------------
struct SplayTree {
    Nod * root;
    SplayTree() {
        root = null;
    }
    inline void rotate(Nod *x, bool c) {  //c=0 左旋转，c=1 右旋转
        Nod *y = x->p;
        y->ch[!c] = x->ch[c];
        if (x->ch[c] != null)  x->ch[c]->p = y;
        x->ch[c] = y;

        x->p = y->p;
        y->p = x;

        if (x->p != null)
            x->p->ch[ x->p->ch[1]==y ] = x;
        else root = x;
    }
    inline void splay(Nod * x, Nod * f=null) {     //表示把结点 x 转到结点 f 的下面,
默认为节点x成为根
        Nod * y;
        bool b1, b2;
        while((y=x->p) != f) {
            if(y->p == f) {
                rotate(x, y->ch[0]==x);
            } else {
                b1 = (y==y->p->ch[0]);
                b2 = (x==y->ch[0]);
                rotate((b1^b2) ? x : y, b2);
                rotate(x, b1);
            }
        }
    }
    inline void insert(type v) {          //插入一个元素
        if(root == null) {
            root = newNod();
            root->init(null,null,null,v);
            return;
        }
        Nod * i = root;
```

```
          int tmp;
          while(1) {
               if(v < i->val)     tmp=0;
               else               tmp=1;
               if(i->ch[tmp] == null) {
                    Nod * x = newNod();
                    x->init(null,null,i,v);
                    i->ch[tmp] = x;

                    splay(x);
                    break;
               }
               i = i->ch[tmp];
          }
     }
     //下面是方向函数
     inline Nod * to(Nod * x, int dir) {   //从x一直向dir走到头
          while(x->ch[dir] != null)   x = x->ch[dir];
          return x;
     }
     inline Nod * next() {                 //root的后继
          return to(root->ch[1], 0);
     }
     inline Nod * prev() {                 //root的前驱
          return to(root->ch[0], 1);
     }
};

int main() {
     int n, val;
     while(scanf("%d", &n) != EOF) {
          initTree();
          SplayTree sp;
          int ans = 0;
          for(int i = 0; i < n; i ++) {
               if(scanf("%d", &val)==EOF)  val = 0; //奇怪的读入!!
               sp.insert(val);
               Nod * l = sp.prev(), * r = sp.next();
               int tmp = 0x7fffffff;
               if(l!=null)   tmp = min(tmp, abs(l->val - val));
               if(r!=null)   tmp = min(tmp, abs(r->val - val));
               if(tmp == 0x7fffffff)   tmp = val;

               ans += tmp;
          }
          printf("%d\n", ans);
     }
     return 0;
}
```

郁闷的出纳员

```
int CH[SIZE][2], P[SIZE], V[SIZE];        //0 左 1 右、父、值
int len;                                  //资源使用量

//------------------------------------------------------------
int CNT[SIZE];     //本棵树所包含的节点个数

//------------------------------------------------------------

void initTree() {                         //只需掉一次，清资源，设NULL节点
     len = 1;                             //从 1 开始，0 代表NULL
     CH[0][0] = CH[0][1] = P[0] = CNT[0] = 0;       //保持 0 节点的左右父一直为 0
}
struct SplayTree {
     int r;    //根
     SplayTree(int r = 0) {
          this->r = r;                    //0 代表NULL
     }
     void rotate(int x, bool c) {         //将节点i，c=0 左旋转，c=1 右旋转
          int y = P[x];
          CH[y][!c] = CH[x][c];
          if(CH[x][c])  P[CH[x][c]] = y;
          CH[x][c] = y;

          P[x] = P[y];
          P[y] = x;

          if(P[x]) CH[ P[x] ][ CH[P[x]][1]==y ] = x;
          else     r = x;
          push_up(y);
     }
     void splay(int x, int f=0) {     //表示把结点 x 转到结点 f 的下面，默认为节点x成为
根
          int y;
          bool b1, b2;
          for(; (y=P[x]) != f; ) {          //用到了P[0]=0 来处理x=0 的情况
               if(P[y] == f) {
                    rotate(x, CH[y][0] == x);
               } else {
                    b1 = (y==CH[P[y]][0]);
                    b2 = (x==CH[  y ][0]);
                    rotate((b1^b2) ? x : y, b2);
                    rotate(x, b1);
               }
          }
          push_up(x);
     }
     int insert(int v) {     //插入一个元素，返回这个元素的编号
          int i = len ++, *j = &r;
          CH[i][0] = CH[i][1] = P[i] = 0;
          V[i] = v;
          while(*j)
               P[i]=*j ,j = CH[*j] + (v>=V[*j]);
          splay(*j=i);
          return i;
     }
     int lower_bound(int v) {
          int i = r;
          int tgt = 0;
          while(i) {
               if(V[i] < v) {
                    i = CH[i][1];
```

```
            } else {
                tgt = i;
                i = CH[i][0];
            }
        }
        return tgt;
    }
//---------------------以下代码用于解决cnt和getK()-----------------------
    int getK(int rank) {    //从1开始
        if(rank > CNT[r] || rank<=0)     return 0;
        int i;
        for(i = r; CNT[CH[i][0]]+1 != rank; ) {
            if(CNT[CH[i][0]] >= rank) {
                i = CH[i][0];
            } else {
                rank -= CNT[CH[i][0]]+1;
                i = CH[i][1];
            }
        }
        splay(i);
        return i;
    }
//-------------------------------以上代码基本不用改
    void push_up(int idx) {
        CNT[idx] = CNT[CH[idx][0]]+CNT[CH[idx][1]]+1;
    }
    int removeLessThan(int val) {    //删除比val小的所有节点，返回删除的节点的个数
        int idx = lower_bound(val);
        int res;
        if(idx == 0) {     //没有大于等于val的节点，将整个树全部删除!
            res = CNT[r];
            r = 0;         //新建一颗树
        } else {
            splay(idx);
            res = CNT[ CH[r][0] ];
            CH[r][0] = 0;
            push_up(r);
        }
        return res;
    }
};

/**
 * hnoi-1503  郁闷的出纳员
输入:
    名称 格式 作用
    I命令 I_k 新建一个工资档案，初始工资为k。如果某员工的初始工资低于工资下界，他将立刻离开公司。
    A命令 A_k 把每位员工的工资加上k
    S命令 S_k 把每位员工的工资扣除k
    F命令 F_k 查询第k多的工资
输出:
    对于每条F命令，你的程序要输出一行，仅包含一个整数，为当前工资第k多的员工所拿的工资数，如果k大于目前员工的数目，则输出-1。
    输出文件的最后一行包含一个整数，为离开公司的员工的总数。
 */

int main() {
    int n, mi;
    while(scanf("%d%d", &n, &mi) != EOF) {
        char c;
        int val, base = 0, sum = 0;

        initTree();
        SplayTree sp;
        while(n --) {
            scanf(" %c%d", &c, &val);
            if(c == 'I') {
                if(val >= mi) sp.insert(val+base);
            } else if(c == 'A') {
                base -= val;
            } else if(c == 'S') {
                base += val;
                sum += sp.removeLessThan(mi+base);
            } else {
                int idx = sp.getK(CNT[sp.r]+1-val);
                if(idx == 0) {
                    printf("-1\n");
                } else {
                    printf("%d\n", V[idx]-base);
                }
            }
        }
        printf("%d\n", sum);
    }
    return 0;
}
```

线段树

```
/**【题目0】hdu-3621     Area K
    非线段树的区间问题，计算覆盖了K次的长方形面积并
    效率：n*sqrt(n)
*/

#define hash(x)    ((x) & 131071)       //(x&(2^n-1))
int a1[140000];
double a2[140000];
void initHash() {
   memset(a1, 255, sizeof(a1));
}
int gen(int x) {
   int z = hash(x);
   while(a1[z]!=-1 && a1[z]!=x)    z = hash(z+1);
   if(a1[z] == -1)    a1[z]=x, a2[z]=0;    //don't forget to init a2 !
   return z;
}
```

```
int get(int x) {
    int z = hash(x);
    while(a1[z]!=-1 && a1[z]!=x)    z=hash(z+1);
    if(a1[z]==-1)    return -1;
    return z;
}
//以上是hash表

#define maxn 20100
const double eps = 1E-6;
int sig(double d) {   return (d>eps) - (d<-eps);    }

double arr[maxn];                   //每个区间的刻度值
struct List {
#define pos(i, j)    ((j+1)*(num+10)+(i+1))    //表示第i块的小块们，覆盖了j次的长度
    int p[maxn], n, k;              //每个小快覆盖次数，小快个数，要统计覆盖k次(k>0)
    int block, q[210], num;          //每个大块的长度，每个大块的覆盖次数，块数
    double ans;                     //覆盖k次的总长度
    void init(int n, int k) {       //assume n>=2
        this->n = n;
        this->k = k;
        memset(p, 0, sizeof(p));
        memset(q, 0, sizeof(q));
        block = (int)sqrt((double)n);
        ans = 0;
        num = (n-1+block-1)/block;
        initHash();
        for(int i = 0; i < num-1; i ++)
            a2[ gen(pos(i,0)) ] = arr[(i+1)*block]-arr[i*block];
        a2[ gen(pos(num-1,0)) ] = arr[n-1]-arr[(num-1)*block];
    }
    void fill(int l, int r, int v) {
        int idx;
        double tmp;
        for(int j = l; j < r; j ++) {
            idx = j/block;
            tmp = arr[j+1] - arr[j];
            a2[ gen(pos(idx,p[j])) ] -= tmp;    if(p[j]+q[idx] == k)    ans -= tmp;
            p[j] += v;
            a2[ gen(pos(idx,p[j])) ] += tmp;    if(p[j]+q[idx] == k)    ans += tmp;
        }
    }
    void insert(double l, double r, int v) {
        int lw = lower_bound(arr, arr+n, l-eps)-arr;    //减去eps比较安全!
        int rw = lower_bound(arr, arr+n, r-eps)-arr;

        int L = (lw+block-1)/block, R = rw/block;
        if(R-L>0) {
            int j;
            for(int i = L; i < R; i ++) {
                if(k-q[i]>=0 && (j=get(pos(i, k-q[i])))!=-1)    ans-=a2[j];
                q[i] += v;
                if(k-q[i]>=0 && (j=get(pos(i, k-q[i])))!=-1)    ans+=a2[j];
            }
            fill(R*block, rw, v);
            fill(lw, L*block, v);
        } else {
            fill(lw, rw, v);
        }
    }
} lst;

struct Nod {
    double x, y1, y2;
    int mode;           //进入 1，退出 2
    void init(double x, double y1, double y2, int mode) {
        this->x = x;
        this->y1= y1;
        this->y2= y2;
        this->mode=mode;
    }
    bool operator < (const Nod & nod) const {
        int val = sig(x-nod.x);
        return val==0 ? mode>nod.mode : val<0;    //进入优先!
    }
};
bool cmpEq(double a, double b) {   return sig(a-b)==0;    }
double compute(Nod * nods, int n, int k) {    //n传入的是挡板的个数，不是矩形个数!
    if(n == 0)    return 0;
    int len = 0;
    for(int i = 0; i < n; i += 2) {       //assume奇偶是一样的y1 和y2
        arr[len ++] = nods[i].y1;
        arr[len ++] = nods[i].y2;
    }
    sort(arr, arr+len);
    len = unique(arr, arr+len, cmpEq) - arr;
    lst.init(len, k);

    sort(nods, nods+n);
    double ans = 0, lastX = nods[0].x;
    for(int i = 0; i < n; i ++) {
        ans += lst.ans*(nods[i].x-lastX);
        lst.insert(nods[i].y1, nods[i].y2, nods[i].mode);
        lastX = nods[i].x;
    }
    return ans;
}

int n, k;
Nod nods[maxn];
/**
 * hdu-3621
 * Area k(2010 天津网赛)
 */
int main() {
    for(int idx = 1; scanf("%d",&n), n; idx ++) {
        int x, y, z, l;
        int newN = 0;
        for(int i = 0; i < n; i ++) {
            scanf("%d%d%d%d", &x, &y, &z, &l);
            if(2*z<=l) {
                nods[newN ++].init(x-l/2.0, y-l/2.0, y+l/2.0, 1);
                nods[newN ++].init(x+l/2.0, y-l/2.0, y+l/2.0, -1);
            }
        }
        scanf("%d", &k);
        double ans = compute(nods, newN, k);
        printf("Case %d: %.3f\n", idx, ans);
    }
    return 0;
```

```
}

/**
     【题目 1】ecnu-1317  空心长方体
    在一个三维正坐标系中，存在N(N<=5000)个点，现在要求一点P(x,y,z)，使
     得O(0,  0,  0)与P(x,y,z)两个顶点构成的长方体内不包括N个点中的任何一个点(在
     长方体边缘不算包括)，并使这个长方体的体积最大。x,y,z均不得超过 1000000。
 */

#define maxn 5010
int xs[maxn], N;

#define MEM 17000
const int BOUND = 1000000;
const int inf = 0x3f3f3f3f;

struct SegTree {
     int L[MEM], R[MEM], V[MEM], MAX[MEM];
     long long AREA[MEM];
     int MAX_X[MEM];

     int n;
     void init(int size) {
          for(n=1; n<size; n<<=1);
          for(int i = n; i < 2*n; i ++) {
               L[i] = i-n;
               R[i] = i-n+1;
          }
          for(int i = n-1; i; i --) {
               L[i] = L[2*i];
               R[i] = R[2*i+1];
          }
          memset(V, 255, sizeof(V));
          memset(MAX, 255, sizeof(MAX));
          memset(AREA, 255, sizeof(AREA));
          memset(MAX_X, 255, sizeof(MAX_X));

          insert(0, size, BOUND); //Y的边界
     }
     void insert(int l, int r, int v, int idx = 1) {//我表示要覆盖!
          if(l <= L[idx] && r >= R[idx]) {
               V[idx] = v;
               MAX[idx] = V[idx];
               AREA[idx] = (long long)xs[R[idx]-1]*V[idx];
               MAX_X[idx] = xs[R[idx]-1];
          } else {
               if(V[idx] != -1) {
                    V[2*idx] = V[2*idx+1] = V[idx];
                    MAX[2*idx] = MAX[2*idx+1] = V[idx];
                    AREA[2*idx] = (long long)xs[R[2*idx]-1]*V[2*idx];
                    AREA[2*idx+1] = (long long)xs[R[2*idx+1]-1]*V[2*idx+1];
                    MAX_X[2*idx] = xs[R[2*idx]-1];
                    MAX_X[2*idx+1] = xs[R[2*idx+1]-1];
                    V[idx] = -1;
               }
               if(l < (L[idx]+R[idx])/2)   insert(l, r, v, 2*idx);
               if(r > (L[idx]+R[idx])/2)   insert(l, r, v, 2*idx+1);
               MAX[idx] = max(MAX[2*idx], MAX[2*idx+1]);
               if(AREA[2*idx]>=AREA[2*idx+1]) {
                    AREA[idx] = AREA[2*idx];
                    MAX_X[idx] = MAX_X[2*idx];
               } else {
                    AREA[idx] = AREA[2*idx+1];
                    MAX_X[idx] = MAX_X[2*idx+1];
               }
          }
     }
     int queryR(int y) {     //返回大于y的最右区间
          int idx = 1;
          if(MAX[idx] <= y)  return -1;
          while(idx < 2*n) {
               if(V[idx] != -1)        return R[idx];
               if(MAX[2*idx+1] > y)    idx = 2*idx+1;
               else                    idx = 2*idx;
          }
          while(1); //shouldn't be here!
     }
     void insert(int x, int y) {
          x = lower_bound(xs, xs+N, x) - xs + 1;
          int r = queryR(y);
          if(x<r)   insert(x, r, y);
     }
} st;

struct Point {
     int x, y, z;
     bool operator < (const Point & p) const {
          return z < p.z;
     }
};
//最大空心长方体，传入点，返回面积，并且保存P点坐标xyz，如果有冲突，按照xyz依次字典序最小
long long largestEmptyBox(Point * ps, int n, int & x, int & y, int & z) {
     for(int i = 0; i < n; i ++) xs[i] = ps[i].x;
     N = n;
     xs[N ++] = BOUND;  //x的边界，作为X的哨兵
     sort(xs, xs+N);
     st.init(N);
     sort(ps, ps+n);
     ps[n].x = ps[n].y = 1;
     ps[n ++].z = BOUND;     //Z的边界，作为Z的哨兵

     int j;
     long long area = 0;
     x = y = z = -1;

     for(int i = 0; i < n; i = j) {
          long long tmp = st.AREA[1]*ps[i].z;
          if(tmp >= area) {
               area = tmp;
               z = ps[i].z;
               x = st.MAX_X[1];
          }
          for(j = i; j < n && ps[i].z==ps[j].z; j ++) {
               st.insert(ps[j].x, ps[j].y);
          }
     }
     y = area / x / z;
     return area;
}
int n;
```

```
Point ps[5010];
int main() {
    while(scanf("%d", &n) != EOF) {
        for(int i = 0; i < n; i ++) {
            scanf("%d%d%d", &ps[i].x, &ps[i].y, &ps[i].z);
        }
        int x, y, z;
        largestEmptyBox(ps, n, x, y, z);
        printf("%d %d %d\n", x, y, z);
    }
    return 0;
}

//【题目 2】hdu－3627    Giant For
//询问矩阵中某个元素右下方最近的点
//支持如下操作：
//1) add: Mark an element in the matrix. It is guaranteed that the element has not
been marked before.
//2) remove: Delete an element's mark. It is guaranteed that the element has been
marked before.
//3) find: For a given element's row and column, return a marked element's row and
column, where the marked element's row and column are larger than the given element's
row and column respectively. If there are multiple solutions, return the element whose
row is the smallest; and if there are still multiple solutions, return the element
whose column is the smallest. If there is no solution, return -1.
#define maxn 600010

struct SegTree {
   int L[maxn], R[maxn], Y[maxn];
   int n;
   void init(int size) {
      for(n=1; n<size; n<<=1);
      for(int i = n; i < 2*n; i ++)    L[i]=i-n, R[i]=i-n+1;
      for(int i = n-1; i ; i --)       L[i]=L[2*i], R[i]=R[2*i+1];
      memset(Y, 255, sizeof(Y));
   }
   void insert(int l, int r, int y, int idx = 1) {
      if(l<=L[idx] && r>=R[idx]) {
         Y[idx] = y;
      } else {
         if(l < (L[idx]+R[idx])/2)    insert(l, r, y, 2*idx);
         if(r > (L[idx]+R[idx])/2)    insert(l, r, y, 2*idx+1);
         Y[idx] = max(Y[2*idx], Y[2*idx+1]);
      }
   }
   int query(int l, int r, int y, int idx = 1) {
      if(l<=L[idx] && r>=R[idx]) {
         if(Y[idx]<=y)    return -1;
         while(idx<n) {
            if(Y[2*idx]>y)    idx = 2*idx;
            else              idx = 2*idx+1;
         }
         return L[idx];
      } else {
         if(l<(L[idx]+R[idx])/2) {
            int tmp = query(l, r, y, 2*idx);
            if(tmp != -1)    return tmp;
         }
         if(r>(L[idx]+R[idx])/2) return query(l, r, y, 2*idx+1);
         return -1;
      }
   }
} st;

typedef pair<int,int> T;

T ts[maxn];    char cmd[maxn];    int n;
T ts2[maxn];

int main() {
   char str[10];
   for(int idx = 1; scanf("%d", &n)!=EOF && n; idx ++) {
      if(idx!=1)    printf("\n");
      printf("Case %d:\n", idx);

      st.init(n);
      for(int i = 0; i < n; i ++) {
         scanf("%s%d%d", str, &ts[i].first, &ts[i].second);       ///first is X!
firstly sort by X!
         cmd[i] = *str;
         ts2[i] = ts[i];
      }
      sort(ts2, ts2+n);
      for(int i = 0; i < n; i ++) {
         int idx = lower_bound(ts2, ts2+n, ts[i])-ts2;
         if(cmd[i]=='a') {
            st.insert(idx, idx+1, ts[i].second);
         } else if(cmd[i]=='r') {
            st.insert(idx, idx+1, -1);
         } else {
            T t(ts[i].first+1, -1);
            int j = lower_bound(ts2, ts2+n, t)-ts2;
            int k = st.query(j, n, ts[i].second);
            if(k == -1)    printf("-1\n");
            else printf("%d %d\n", ts2[k].first, ts2[k].second);
         }
      }
   }
   return 0;
}
//【题目 3】长方体体积并
//计算覆盖了 3 次及以上的体积并

#define SIZE 1100
#define MEM 2*SIZE+10

struct SegTree {
   int L[MEM], R[MEM], V[MEM], V1[MEM], V2[MEM], V3[MEM];
   int n;

   void init(int size) {
      for(n=1; n<size; n<<=1);
      for(int i = n; i < 2*n; i ++) {
         L[i] = i-n;
         R[i] = i-n+1;
      }
      for(int i = n-1; i; i --) {
         L[i] = L[2*i];
         R[i] = R[2*i+1];
      }
```

```
        memset(V, 0, sizeof(V));
        memset(V1, 0, sizeof(V1));
        memset(V2, 0, sizeof(V2));
        memset(V3, 0, sizeof(V3));
    }
    void insert(int l, int r, int v, int idx = 1) {
        if(l<=L[idx] && r>=R[idx]) {
            V[idx] += v;
        } else {
            if(l<(L[idx]+R[idx])/2)    insert(l, r, v, 2*idx);
            if(r>(L[idx]+R[idx])/2)    insert(l, r, v, 2*idx+1);
        }

        if(idx>=n) {
            V1[idx] = V2[idx] = V3[idx] = 0;
            if(V[idx] == 0) {
            } else if(V[idx]==1) {
                V1[idx] = 1;
            } else if(V[idx]==2) {
                V2[idx] = 1;
            } else {
                V3[idx] = 1;
            }
        } else {
            V1[idx] = V2[idx] = V3[idx] = 0;
            if(V[idx] == 0) {
                V1[idx] = V1[2*idx] + V1[2*idx+1];
                V2[idx] = V2[2*idx] + V2[2*idx+1];
                V3[idx] = V3[2*idx] + V3[2*idx+1];
            } else if(V[idx]==1) {
                V2[idx] = V1[2*idx] + V1[2*idx+1];
                V3[idx] = V2[2*idx] + V2[2*idx+1] + V3[2*idx] + V3[2*idx+1];
                V1[idx] = R[idx]-L[idx] - V2[idx]-V3[idx];
            } else if(V[idx]==2) {
                V3[idx] = V1[2*idx] + V1[2*idx+1] + V2[2*idx] + V2[2*idx+1] + V3[2*idx]
+ V3[2*idx+1];
                V2[idx] = R[idx]-L[idx]-V3[idx];
                V1[idx] = 0;
            } else {
                V3[idx] = R[idx]-L[idx];
                V1[idx] = V2[idx] = 0;
            }
        }
    }

    int query(int l, int r, int v = 0, int idx = 1) {
        if(l <= L[idx] && r >= R[idx]) {
            if(v == 0) {
                return V3[idx];
            } else if(v == 1) {
                return V2[idx] + V3[idx];
            } else if(v == 2) {
                return V1[idx] + V2[idx] + V3[idx];
            } else {
                return R[idx] - L[idx];
            }
        } else {
            if(l < (L[idx]+R[idx])/2)    insert(l, r, v+V[idx], 2*idx);
            if(r > (L[idx]+R[idx])/2)    insert(l, r, v+V[idx], 2*idx+1);
        }
    }
} st;

struct VerSeg {
    int y, z1, z2;
    int mode;//1 为进入线段，-1 为退出线段
    bool operator <(const VerSeg& vs) const {
        return y < vs.y;
    }
    void set(int y, int z1, int z2, int mode) {
        this->y = y;
        this->z1 = z1;
        this->z2 = z2;
        this->mode = mode;
    }
    void output() {
        printf("y = %d, z1 = %d, z2 = %d, mode = %d\n", y, z1, z2, mode);
    }
};

long long rectCal(VerSeg*segs, int n) {
//    printf("n = %d\n", n);
    long long res = 0;
    sort(segs, segs+n);
//    printf("sort over!\n");
    st.init(1024);
    int lastY= 0;
    for(int i = 0; i < n; i ++) {
//        printf("i = %d\n", i);
        res += (long long)(st.V3[1]) * (long long)(segs[i].y - lastY);
        st.insert(segs[i].z1, segs[i].z2, segs[i].mode);
        lastY = segs[i].y;
//        printf("over!\n");
    }
    return res;
}

struct Point {
    int x, y, z;
    void input() {
        scanf("%d%d%d", &x, &y, &z);
        z += 500;
    }
};

struct Box {
    Point a, b;
};

Box bs[1010];
int xs[2010];


VerSeg segs[10010];

bool inter(int a1, int b1, int a2, int b2) {
    return min(b1, b2) - max(a1, a2) > 0;
}

int main() {
```

```
    int t, n;
    int idx = 0;
    for(scanf("%d", &t); t --; ) {
        scanf("%d", &n);
        int lenX = 0;
        for(int i = 0; i < n; i ++) {
            bs[i].a.input();
            bs[i].b.input();
            xs[lenX ++] = bs[i].a.x;
            xs[lenX ++] = bs[i].b.x;
        }
        sort(xs, xs+lenX);
        lenX = unique(xs, xs+lenX) - xs;

        /*for(int i = 0; i < lenX; i ++) {
            printf("i = %d, x = %d\n", i, xs[i]);
        }*/

        long long ans = 0;

        for(int i = 0; i < lenX-1; i ++) {
            int x1 = xs[i], x2 = xs[i+1];
            int segLen = 0;
            for(int j = 0; j < n; j ++) {
                if(inter(x1, x2, bs[j].a.x, bs[j].b.x)) {
                    segs[segLen ++].set(bs[j].a.y, bs[j].a.z, bs[j].b.z, 1);
                    segs[segLen ++].set(bs[j].b.y, bs[j].a.z, bs[j].b.z, -1);
                }
            }
            /*printf("segLen = %d\n", segLen);
            for(int k = 0; k < segLen; k ++) {
                segs[k].output();
            }*/
            long long res = rectCal(segs, segLen);

            ans += res * (long long)(x2-x1);
        }
        cout << "Case " << ++ idx << ": ";
        cout << ans << endl;
    }
    return 0;
}
//【题目 4】区间K小元素(划分树)
#define LL(x) x<<1
#define RR(x) 1+(x<<1)
#define M(x)  (L[x]+R[x])>>1
#define maxn 140000
/**
    归并树，将归并排序和线段树相结合

    可以求[l, r)区间上的k小元素
    要求：元素不能相同
-----------------------------------------------------------------
    教程:
    （1）用线段树来表示区间，构造线段树O(NlogN)，这样能在O(logN)时间内确定区间的最大值。
    （2）另外保存了排序后的值后，那给定一个值，可以在区间内查找从而得出该值排序后的位置，这里可以用二分查找，复杂度又乘上了O(logN)。这儿要注意的是如果存在两个或多个相同值的时候应该输出最小的位置，自然得可以理解为并列名次。
    （3）题目要求输出区间内指定名次的数值。我们从上面可以由一个值来确定名次，显然这个名次在排序后的序列中是非递减的，所以这儿又可以二分枚举，在排序后的序列中二分枚举一个数值，用（2）的方法得出名次，和指定的名次进行比较。这里的二分和（2）的二分对相同值的处理不同，这里要取相同值的最大位置，原因是当非区间内的数比区间内的数大时，才使得名次+1。
    （4）最后的时间复杂度是O(MlogNlogN)
    听说有不需造线段树的方法，继续研究……
*/
struct MergeTree {
/**
    A: 保存初始元素的数组
    V: 归并排序的中间过程
    L,R:线段树的左右指针
*/
    int A[maxn], V[20][maxn];
    int L[2*maxn+10], R[2*maxn+10];
    /**
        归并开始
        参数:
            l: 归并的左范围（初始调用为 0）
            r: 归并的右范围（初始调用为n）
            d: 归并的深度
            i: 线段树中的线段索引
    */
    void build(int l, int r, int d = 0, int i = 1) {
        L[i] = l;    R[i] = r;
        if(r-l == 1) {
            V[d][l] = A[l];
            return;
        }
        int m = M(i), il = l, ir = m;
        build(l, m, d+1, LL(i));
        build(m, r, d+1, RR(i));

            merge(V[d+1]+l, V[d+1]+m, V[d+1]+m, V[d+1]+r, V[d]+l);
    }
    /**
        询问[l, r)区间上，值小于key的元素个数
        参数:
            l: 询问的左范围（初始调用为 0）
            r: 询问的右范围（初始调用为n）
            key:要查找的值
            d: 进行的深度
            i: 线段树中的线段索引
    */
    int l, r, key;
    int query0(int d = 0, int i = 1) {
        if(l <= L[i] && r >= R[i]) {
            return lower_bound(V[d]+L[i], V[d]+R[i], key)-V[d]-L[i];
            /**
            为了加速可以自己写二分。。
                int l = L[i]-1, r = R[i], m;
                while(r - l > 1) {
                    m = (l+r)>>1;
                    if(V[d][m] < key) {
                        l = m;
                    } else {
                        r = m;
                    }
                }
                return r - L[i];
            */
        }
```

```
        int num = 0, m = M(i);
        if(l < m)    num += query0(d+1, LL(i));
        if(r > m)    num += query0(d+1, RR(i));
        return num;
    }
    /**
        询问在[L, r)区间上，排名第rank(0开始)的元素
        参数:
            n: 数组长度（因为程序不保存，需传入）
            l: 询问的左范围
            r: 询问的右范围
            rank: 排名
    */
    int query(int n, int l, int r, int rank) {
        int ir = n, il = 0, m;
        this->l = l;     this->r = r;
        while(ir-il>1) {
            m = (il + ir) >> 1;
            this->key = V[0][m];
            if(query0()>rank)
                ir = m;
            else
                il = m;
        }
        return V[0][il];
    }
};

MergeTree mt;
int n, m;

int main() {
    int t;
   // for(scanf("%d", &t); t --; ) {
        scanf("%d%d", &n, &m);

        for(int i = 0; i < n; i ++)
            scanf("%d", mt.A+i);

        mt.build(0, n);
        int L, R, rank;
        for(int i = 0; i < m; i ++) {
            scanf("%d%d%d", &L, &R, &rank);
            printf("%d\n", mt.query(n, L-1, R, rank-1));
        }
    //}
    return 0;
}
```

```
//   【【【【NotOnlySuccess的 线段树小结】】】】】】】】
//
//   我是按照notonlysuccess大牛的线段树专辑做的
//
//   转自：http://www.notonlysuccess.com/?p=59
//
//   线段树大致分为两类：
//
//   1.插入完全覆盖以前的插入值，即只保证最上层是有效的，pku3667 是经典代表。。这种题常用一个线段一直向下压，并且经常要使用EMPTY值，用于清空。
//
//   2.插入考虑到以前的插入值。。这种题就考虑到子节点，同时更新自己就行了
//
//   考虑线段树的思路，应该考虑本条线段应该有的性质，然后考虑怎么传给父线段
//
//   线段树专辑
//
//   这几天陆陆续续更新了下边几道我所能找到得具有一些代表性的线段树题目
//   从最最简单的区间求和到对区间的各种操作都包涵在这些题目里了
//   相信对一些准备学习线段树的人有一定得帮助
//   突然发现自己对数据结构的题目非常有感觉,所以在刷下边的题的同时也生出灵感出了好几道线段树题目
//   等比赛结束后也会陆续加进里边
//
//   快半年过去代码风格也有很大的改变，感觉以前写的代码很不规范，用自己在预定义中定义的一些函数，但后来感觉作用不是很大，所以又删去了，所以现在看代码可能找不到以前我定义的一些函数，本来想更新一下的，无奈这些函数用的太多，改之太过麻烦，所以在此处申明一下
//   #define LL(x) ((x)<<1)
//   #define RR(x) ((x)<<1|1)
//   #define FF(i,n) for(int i = 0 ; i < n ; i ++)
//   若还有不清楚的地方，只管提出来便是，我一定一一改正
//
//   1.hdu1166 敌兵布阵
//   更新节点,区间求和

#define SIZE 70010
#define MEM 2*SIZE+10
#define SET(x)    memset(x, 0, sizeof(x))
struct SegTree {
    int L[MEM], R[MEM];
    int n;
    int C[MEM], SUM[MEM];
    void init(int size) {
        for(n = 1; n<size; n<<=1);

        for(int i = n; i < 2*n; i ++) {
            L[i] = i-n;
            R[i] = i-n+1;
        }
        for(int i = n-1; i >= 1; i --) {
            L[i] = L[2*i];
            R[i] = R[2*i+1];
        }
        SET(C);    SET(SUM);
    }

    void updateSUM(int idx) {
        SUM[idx] = C[idx];
```

```
        if(idx < n) {
            SUM[idx] += SUM[2*idx]+SUM[2*idx+1];
        }
    }

    void insert(int l, int r, int v, int idx=1) {
        if(l<=L[idx] && r>=R[idx]) {
            C[idx] += v;
        } else {
            if(l < (L[idx]+R[idx])/2)    insert(l, r, v, 2*idx);
            if(r > (L[idx]+R[idx])/2)    insert(l, r, v, 2*idx+1);
        }
        updateSUM(idx);
    }
    int query(int l, int r, int idx = 1) {
        int res = 0;
        if(l<=L[idx] && r>=R[idx]) {
            res += SUM[idx];
        } else {
            if(l < (L[idx]+R[idx])/2)    res += query(l, r, 2*idx);
            if(r > (L[idx]+R[idx])/2)    res += query(l, r, 2*idx+1);
        }
        return res;
    }
} segTree;

int main() {
    int t, n, tmp;
    char str[100];
    int a, b;
    scanf("%d", &t);
    for(int idx = 1; idx <= t; idx ++) {
        scanf("%d", &n);
        segTree.init(n);
        for(int i = 0; i < n; i ++) {
            scanf("%d", &tmp);
            segTree.insert(i, i+1, tmp);
        }
        printf("Case %d:\n", idx);
        while(scanf("%s", str), *str != 'E') {
            scanf("%d%d", &a, &b);
            if(*str == 'A') {
                segTree.insert(a-1, a, b);
            } else if(*str == 'S') {
                segTree.insert(a-1,a, -b);
            } else {
                printf("%d\n", segTree.query(a-1,b));
            }
        }
    }
    return 0;
}
//zkw线段树，下标只能在此范围：[1,2^n-2]，即最多装 2^n-2 个数字
#define maxn 66000
struct SegTree {
     int a[2*maxn], n;
     void init(int size) {
          for(n=1; n-2<size; n<<=1);//只能装n-2 个数字
          memset(a, 0, sizeof(a));
     }
     void insert(int i, int v) { //将i位置上加上v
          for(a[i+=n]+=v,i>>=1; i; i >>=1)
               a[i] = a[2*i]+a[2*i+1];
     }
     int query(int s, int t) {   //询问[s,t]的区间内的和
          int res = 0;
          for(s=s+n-1,t=t+n+1; s^t^1; s>>=1,t>>=1) {
               if(~s&1)  res+=a[s^1];
               if( t&1)  res+=a[t^1];
          }
          return res;
     }
} st;

int main() {
    int t, n, tmp;
    char str[100];
    int a, b;
    scanf("%d", &t);
    for(int idx = 1; idx <= t; idx ++) {
        scanf("%d", &n);
        st.init(n);
        for(int i = 1; i <= n; i ++) {
            scanf("%d", &tmp);
            st.insert(i, tmp);
        }
        printf("Case %d:\n", idx);
        while(scanf("%s", str), *str != 'E') {
            scanf("%d%d", &a, &b);
            if(*str == 'A') {
                st.insert(a, b);
            } else if(*str == 'S') {
                st.insert(a, -b);
            } else {
                printf("%d\n", st.query(a,b));
            }
        }
    }
    return 0;
}
//   2.hdu1754 I Hate It
//   更新节点,区间最值
#define SIZE 262144
#define MEM 2*SIZE+10
struct SegTree {
    int L[MEM], R[MEM], V[MEM];
    int n;
    void init(int size) {
        for(n = 1; n < size; n<<=1);
        for(int i = n; i < n<<1; i ++) {
            L[i] = i-n;
            R[i] = i-n+1;
            if(i<size+n) {
                V[i] = ms();
            } else {
                V[i] = 0;
            }
        }
        for(int i = n-1; i >= 1; i --) {
            L[i] = L[i<<1];
```

```
            R[i] = R[i<<1|1];
            V[i] = max(V[i<<1], V[i<<1|1]);
        }
    }
    void insert(int l, int r, int v, int idx=1) {
        if(l<=L[idx] && r>=R[idx]) {
            V[idx] = v;
        } else {
            if(l<(L[idx]+R[idx])>>1)    insert(l, r, v, idx<<1);
            if(r>(L[idx]+R[idx])>>1)    insert(l, r, v, idx<<1|1);
            V[idx] = max(V[idx<<1], V[idx<<1|1]);
        }
    }
    int query(int l, int r, int idx = 1) {
        if(l<=L[idx] && r>=R[idx]) {
            return V[idx];
        } else {
            int res = 0;
            if(l<(L[idx]+R[idx])>>1)    res = max(res, query(l,r,idx<<1));
            if(r>(L[idx]+R[idx])>>1)    res = max(res, query(l,r,idx<<1|1));
            return res;
        }
    }
} segTree;

int main() {
    int n, m, a, b;
    char c;
    while(n=ms(), n!=-1) {
        m = ms();
        segTree.init(n);
        for(int i = 0; i < m; i ++) {
            while(c=getchar(),c!='Q'&&c!='U');
            a=ms();   b=ms();
            if(c == 'Q') {
                printf("%d\n", segTree.query(a-1, b));
            } else {
                segTree.insert(a-1,a,b);
            }
        }
    }
    return 0;
}
//zkw线段树，下标只能在此范围：[1,2^n-2]，即最多装 2^n-2 个数字
#define maxn 270000
const int inf = 0x3f3f3f3f;
struct SegTree {
     int a[2*maxn], n;
     void init(int * arr, int size) { //由arr[1...size]进行初始化
          for(n=1; n-2<size; n<<=1);//只能装n-2 个数字
          for(int i = 0; i < n; i ++)
               a[n+i] = (i>=1&&i<=size) ? arr[i] : -inf;
          for(int i = n-1; i; i --)
               a[i] = max(a[2*i], a[2*i+1]);
     }
     void insert(int i, int v) { //将i位置变为v
          for(a[i+=n]=v,i>>=1; i; i >>=1)
               a[i] = max(a[2*i],a[2*i+1]);
     }
     int query(int s, int t) {   //询问[s,t]的区间内的最大值
          int res = -inf;
          for(s=s+n-1,t=t+n+1; s^t^1; s>>=1,t>>=1) {
               if(~s&1)  res = max(res, a[s^1]);
               if( t&1)  res = max(res, a[t^1]);
          }
          return res;
     }
} st;

int arr[maxn];

int main() {
     int n, m, a, b;
     char c;
     while(n=ms(), n!=-1) {
          m = ms();
          for(int i = 1; i <= n; i ++)     arr[i]=ms();
          st.init(arr, n);
          for(int i = 0; i < m; i ++) {
               while(c=getchar(),c!='Q'&&c!='U');
               a=ms();   b=ms();
               if(c == 'Q') {
                    printf("%d\n", st.query(a, b));
               } else {
                    st.insert(a,b);
               }
          }
     }
     return 0;
}

//   3.hdu1698 Just a Hook
//   成段更新,总区间求和
#define SIZE 131072
#define MEM 2*SIZE+10
struct SegTree {    //质保怔了最上层的正确性！下面有什么我不管！
    int L[MEM], R[MEM], V[MEM], SUM[MEM];
    int n;
    void init(int size) {
        for(n = 1; n < size; n <<= 1);
        for(int i = n; i < 2*n; i ++) {
            L[i] = i-n;
            R[i] = i-n+1;
        }
        for(int i = n-1; i >= 1; i --) {
            L[i] = L[2*i];
            R[i] = R[2*i+1];
        }
        memset(V, 0, sizeof(V));
        memset(SUM, 0, sizeof(SUM));
    }

    void insert(int l, int r, int v, int idx = 1) {
        if(l <= L[idx] && r >= R[idx]) {
            V[idx] = v;
            SUM[idx] = v*(R[idx]-L[idx]);
        } else {
            if(V[idx]) {
                V[2*idx] = V[2*idx+1] = V[idx];
                SUM[2*idx] = SUM[2*idx+1] = SUM[idx]/2;
```

```
            V[idx] = 0;
        }
        if(l < (L[idx]+R[idx])/2)    insert(l, r, v, 2*idx);
        if(r > (L[idx]+R[idx])/2)    insert(l, r, v, 2*idx+1);
        SUM[idx] = SUM[2*idx]+SUM[2*idx+1];
    }
  }
} segTree;

int main() {
    int t, n, m;
    t = ms();
    for(int idx = 1; idx <= t; idx ++) {
        n = ms();    m = ms();
        segTree.init(n);
        segTree.insert(0, n, 1);
        int a, b, v;
        while(m --) {
            a = ms();    b = ms();    v = ms();
            segTree.insert(a-1, b, v);
        }
        printf("Case %d: The total value of the hook is %d.\n", idx, segTree.SUM[1]);
    }
    return 0;
}
//   4.hdu1394 Minimum Inversion Number
//   更新节点,区间求和(实现nlog(n)的逆序数方法,和树状数组比起来实在是有点鸡肋)
#define SIZE 10010
#define MEM 2*SIZE + 10
struct SegTree {
    int L[MEM], R[MEM], NUM[MEM];
    int n;
    void init(int size) {
        for(n = 1; n < size; n <<=1);
        for(int i = n; i <2*n; i ++) {
            L[i] = i-n;
            R[i] = i-n+1;
        }
        for(int i = n-1; i >= 1; i --) {
            L[i] = L[2*i];
            R[i] = R[2*i+1];
        }
        memset(NUM, 0, sizeof(NUM));
    }
    void insert(int l, int r, int idx = 1) {
        if(l <= L[idx] && r >= R[idx]) {
            NUM[idx] = 1;
        } else {
            if(l < (L[idx]+R[idx])/2)    insert(l, r, 2*idx);
            if(r > (L[idx]+R[idx])/2)    insert(l, r, 2*idx+1);
            NUM[idx] = NUM[2*idx] + NUM[2*idx+1];
        }
    }
    int query(int l, int r, int idx = 1) {
        if(l <= L[idx] && r >= R[idx])    return NUM[idx];
        int res = 0;
        if(l<(L[idx]+R[idx])/2)    res += query(l, r, 2*idx);
        if(r>(L[idx]+R[idx])/2)    res += query(l, r, 2*idx+1);
        return res;
    }
} st;

int arr[MEM];

int main() {
    int n;
    while(scanf("%d", &n) != EOF) {
        for(int i = 0; i < n; i ++)    scanf("%d", arr+i);

        st.init(n);

        int inv = 0;
        for(int i = 0; i < n; i ++) {
            inv += st.query(arr[i], n);
            st.insert(arr[i], arr[i]+1);
        }
        int ans = inv;
        for(int i = 0; i < n; i ++) {
            inv += n-2*arr[i]-1;
            ans = min(ans, inv);
        }
        printf("%d\n", ans);
    }
    return 0;
}

//   5.hdu1779 9-Problem C暂不公开
//   成段更新,区间最值
#define SIZE 131072
#define MEM 2*SIZE+10
struct SegTree {
    int L[MEM], R[MEM], V[MEM], MAX[MEM], IDX[MEM];
    int n;
    void init(int size) {
        for(n=1; n<size; n<<=1);
        for(int i = n; i < 2*n; i ++) {
            L[i] = i-n;
            R[i] = i-n+1;
            IDX[i] = L[i]+1;
        }
        for(int i = n-1; i ; i --) {
            L[i] = L[2*i];
            R[i] = R[2*i+1];
            IDX[i] = L[i]+1;
        }
        memset(V, 0, sizeof(V));
        memset(MAX, 0, sizeof(MAX));
    }
    void insert(int l, int r, int v, int idx = 1) {
        if(l <= L[idx] && r >= R[idx]) {
            V[idx] += v;
        } else {
            if(l < (L[idx]+R[idx])/2)    insert(l, r, v, 2*idx);
            if(r > (L[idx]+R[idx])/2)    insert(l, r, v, 2*idx+1);
        }
        MAX[idx] = V[idx];
        if(idx < n) {
            if(MAX[2*idx] >= MAX[2*idx+1]) {
                MAX[idx] += MAX[2*idx];
                IDX[idx] = IDX[2*idx];
```

```
            } else {
                MAX[idx] += MAX[2*idx+1];
                IDX[idx] = IDX[2*idx+1];
            }
        }
    }
    int query(int l, int r, int & ansIDX, int idx = 1) {
        if(l <= L[idx] && r >= R[idx]) {
            ansIDX = IDX[idx];
            return MAX[idx];
        } else {
            int res = -999999999, tmp, tmpIDX;
            if(l < (L[idx]+R[idx])/2) {
                tmp = query(l, r, tmpIDX, 2*idx);
                if(tmp > res) {
                    res = tmp;
                    ansIDX = tmpIDX;
                }
            }
            if(r > (L[idx]+R[idx])/2) {
                tmp = query(l, r, tmpIDX, 2*idx+1);
                if(tmp > res) {
                    res = tmp;
                    ansIDX = tmpIDX;
                }
            }
            return res + V[idx];
        }
    }
} st;

int n, m;

int main() {
    char c;
    int a, b, v;
    while(n=ms(), m=ms(), n||m) {
        st.init(n);
        while(m --) {
            while(c=getchar(), c!='I' && c!='C');
            a=ms();b=ms();
            if(c == 'I') {
                v=ms();
                st.insert(a-1, b, v);
            } else {
                int idx, ans;
                ans = st.query(a-1, b, idx);
                printf("%d\n", ans);
                st.insert(idx-1, idx, -ans);
            }
        }
    }
    return 0;
}

//   6.pku2777 Count Color
//   成段更新,区间统计,位运算加速(我总把query里的传递给子节点的步骤忘了)
#define SIZE 131072
#define MEM 2*SIZE+10
```

```
struct SegTree {
    int L[MEM], R[MEM], COL[MEM], DIF[MEM];
    int n;
    void init(int size) {
        for(n = 1; n < size; n <<=1);
        for(int i = n; i < 2*n; i ++) {
            L[i] = i-n;
            R[i] = i-n+1;
        }
        for(int i = n-1; i; i --) {
            L[i] = L[2*i];
            R[i] = R[2*i+1];
        }
        memset(COL, 0, sizeof(COL));
        memset(DIF, 0, sizeof(DIF));
        COL[1] = 1;
        DIF[1] = 2;
    }
    void insert(int l, int r, int col, int idx = 1) {
        if(l <= L[idx] && r >= R[idx]) {
            COL[idx] = col;
            DIF[idx] = 1<<col;
        } else {
            if(COL[idx]) {
                COL[2*idx] = COL[2*idx+1] = COL[idx];
                DIF[2*idx] = DIF[2*idx+1] = DIF[idx];
            }
            COL[idx] = 0;
            DIF[idx] = 0;
            if(l < (L[idx]+R[idx])/2)   insert(l, r, col, 2*idx);
            if(r > (L[idx]+R[idx])/2)   insert(l, r, col, 2*idx+1);
            DIF[idx] = DIF[2*idx] | DIF[2*idx+1];
        }
    }
    int query(int l, int r, int idx = 1) {
        if((l <= L[idx] && r >= R[idx]) || COL[idx]) {
            return DIF[idx];
        }
        int res = 0;
        if(l < (L[idx]+R[idx])/2)   res |= query(l, r, 2*idx);
        if(r > (L[idx]+R[idx])/2)   res |= query(l, r, 2*idx+1);
        return res;
    }
} st;

int main() {
    int l, t, o;
    char c;
    int a, b, v;
    while(l=ms(), l!=-1) {
        t=ms();  o=ms();
        st.init(l);
        while(o --) {
            while(c=getchar(),c!='P'&&c!='C');
            a = ms();b=ms();
            if(a > b) swap(a, b);
            if(c == 'P') {
                int mask = st.query(a-1, b);
                int ans = 0;
                while(mask) {
```

```
                    ans += mask & 1;
                    mask >>= 1;
                }
                printf("%d\n", ans);
            } else {
                v=ms();
                st.insert(a-1, b, v);
            }
        }
    }
    return 0;
}

//   7.pku3468 A Simple Problem with Integers
//   成段更新,区间求和(中间乘法会超int..纠结了)
#define SIZE 140000
#define MEM 2*SIZE+10
struct SegTree {
    long long L[MEM], R[MEM], V[MEM], S[MEM];
    long long n;
    void init(long long size) {
        int v;
        for(n=1; n<size; n<<=1);
        for(long long i = n ; i < 2*n; i ++) {
            L[i] = i-n;
            R[i] = i-n+1;
            if(i<n+size) {
                scanf("%d", &v);
                V[i] = v;
            } else {
                V[i] = 0;
            }
            S[i] = V[i];
        }
        for(long long i = n-1; i; i --) {
            L[i] = L[2*i];
            R[i] = R[2*i+1];
            V[i] = 0;
            S[i] = S[2*i] + S[2*i+1];
        }
    }
    void insert(long long l, long long r, long long v, long long idx = 1) {
        if(l <= L[idx] && r >= R[idx]) {
            V[idx] += v;
        } else {
            if(l < (L[idx]+R[idx])/2)   insert(l, r, v, 2*idx);
            if(r > (L[idx]+R[idx])/2)   insert(l, r, v, 2*idx+1);
        }
        S[idx] = V[idx]*(R[idx]-L[idx]);
        if(idx < n) {
            S[idx] += S[2*idx] + S[2*idx+1];
        }
    }
    long long query(long long l, long long r, long long idx = 1) {
        if(l <= L[idx] && r >= R[idx])   return S[idx];
        long long res = V[idx]*(min(r, R[idx])-max(l, L[idx]));
        if(l < (L[idx]+R[idx])/2)   res += query(l, r, 2*idx);
        if(r > (L[idx]+R[idx])/2)   res += query(l, r, 2*idx+1);
        return res;
    }
} st;

int main() {
    int n, m;
    char c;
    int a, b, v;
    while(scanf("%d%d", &n, &m) != EOF) {
        st.init(n);
        while(m --) {
            scanf(" %c%d%d", &c, &a, &b);
            if(c == 'Q') {
                cout << st.query(a-1, b) << endl;
            } else {
                scanf("%d", &v);
                st.insert(a-1, b, v);
            }
        }
    }
    return 0;
}
//   8.pku2528 Mayor's posters
//   成段更新,区间统计(离散化)
#define SIZE 20010*2
#define MEM 2*SIZE+10
bool vis[MEM];
struct SegTree {
    int L[MEM], R[MEM], V[MEM];
    int n;
    void init(int size) {
        for(n=1; n<size; n<<=1);
        for(int i = n; i < 2*n; i ++) {
            L[i] = i-n;
            R[i] = i-n+1;
        }
        for(int i = n-1; i ; i --) {
            L[i] = L[2*i];
            R[i] = R[2*i+1];
        }
        memset(V, 0, sizeof(V));
    }
    void insert(int l, int r, int v, int idx = 1) {
        if(l <= L[idx] && r >= R[idx]) {
            V[idx] = v;
        } else {
            if(V[idx]) {
                V[2*idx] = V[2*idx+1] = V[idx];
                V[idx] = 0;
            }
            if(l < (L[idx]+R[idx])/2)   insert(l, r, v, 2*idx);
            if(r > (L[idx]+R[idx])/2)   insert(l, r, v, 2*idx+1);
        }
    }
    void dfs(int l, int r, int idx = 1) {
        if(V[idx] || idx>=n) {
            vis[V[idx]] = true;
            return;
        }
        if(l < (L[idx] + R[idx])/2) dfs(l, r, 2*idx);
        if(r > (L[idx] + R[idx])/2) dfs(l, r, 2*idx+1);
    }
```

```
} st;

int A[MEM], B[MEM];
map<int,int> maps;
int main() {
     int t, n;
     for(scanf("%d", &t); t --; ) {
          scanf("%d", &n);
          maps.clear();
          for(int i = 0; i < n; i ++) {
               scanf("%d%d", A+i, B+i);
               maps[A[i]];
               maps[B[i]];
          }
          int num = 1;
          for(map<int,int>::iterator i = maps.begin(); i != maps.end(); i ++) {
               i->second = num ++;
          }
          st.init(num);
          for(int i = 0; i < n; i ++) {
               st.insert(maps[A[i]]-1, maps[B[i]], i+1);
          }
          memset(vis, 0, sizeof(vis));
          st.dfs(0, num);
          int ans = 0;
          for(int i = 1; i <= n; i ++) {
               if(vis[i]) {
                    ans ++;
               }
          }
          printf("%d\n", ans);
     }
     return 0;
}
//   9.hdu2795 Billboard
//   更新节点,询问特殊(暑假的时候不会线段树被这水题狂虐)
#define SIZE 210000*2
#define MEM 2*SIZE + 10

int w, h, n;

struct SegTree {
    int L[MEM], R[MEM], V[MEM];
    int n;
    void init(int size) {
        for(n = 1; n < size; n <<= 1);
        for(int i = n; i < 2*n; i ++) {
            L[i] = i-n;
            R[i] = i-n+1;
            if(i < n+size) {
                V[i] = w;
            } else {
                V[i] = 0;
            }
        }
        for(int i = n-1; i ; i --) {
            L[i] = L[2*i];
            R[i] = R[2*i+1];
            V[i] = max(V[2*i], V[2*i+1]);
        }
    }
    void insert(int l, int r, int v, int idx = 1) {
        if(l <= L[idx] && r >= R[idx]) {
            V[idx] -= v;
        } else {
            if(l < (L[idx]+R[idx])/2)    insert(l, r, v, 2*idx);
            if(r > (L[idx]+R[idx])/2)    insert(l, r, v, 2*idx+1);
            V[idx] = max(V[2*idx], V[2*idx+1]);
        }
    }
    int query(int len) {
        int idx = 1;
        if(V[idx] < len)    return -1;
        while(idx < n) {
            if(V[2*idx]>=len) {
                idx = 2*idx;
            } else {
                idx = 2*idx+1;
            }
        }
        return idx-n+1;
    }
} st;

int main() {
    int tmp, ans;
    while(h=ms(), h!=-1) {
        w = ms();   n = ms();
        h = min(h, n);
        st.init(h);
        for(int i = 0; i < n; i ++) {
            tmp=ms();
            ans = st.query(tmp);
            printf("%d\n", ans);
            if(ans != -1) {
                st.insert(ans-1, ans, tmp);
            }
        }
    }
    return 0;
}
//   10.pku3667 Hotel
//   成段更新,寻找空间(经典类型,求一块满足条件的最左边的空间)
//寻找空间，1.2.3...n个空间中，寻找连续为k的最左面的空间。
//第i个空间对应的线段树区间为[i-1, i]
//EMPTY代表没有被占用，FULL代表被占用，0 代表此线段无意义
#define SIZE 67000
#define MEM 2*SIZE+10

const int EMPTY = -99;
const int FULL = 100;
//0 代表没有插入线段

struct SegTree {
     int L[MEM], R[MEM], V[MEM];
     int LL[MEM], RR[MEM], MAX[MEM];
     int n;
     void init(int size) {
          for(n=1; n<size; n<<=1);
          for(int i = n; i < 2*n; i ++) {
```

```
            L[i] = i-n;
            R[i] = i-n+1;
        }
        for(int i = n-1; i ; i --) {
            L[i] = L[2*i];
            R[i] = R[2*i+1];
        }
        memset(V, 0, sizeof(V));
        memset(LL, 0, sizeof(LL));
        memset(RR, 0, sizeof(RR));
        memset(MAX, 0, sizeof(MAX));
        insert(0, size, EMPTY);
        insert(size, n, FULL);
    }
    void insert(int l, int r, int v, int idx = 1) {
        if(l <= L[idx] && r >= R[idx]) {
            V[idx] = v;
            LL[idx] = RR[idx] = MAX[idx] = (v==EMPTY?R[idx] - L[idx]:0);
        } else {
            if(V[idx]) {
                V[2*idx] = V[2*idx+1] = V[idx];
                LL[2*idx] = RR[2*idx] = MAX[2*idx] = LL[idx]/2;
                LL[2*idx+1]=RR[2*idx+1]=MAX[2*idx+1]=RR[idx]/2;
                V[idx] = 0;
            }
            if(l < (L[idx]+R[idx])/2)   insert(l, r, v, 2*idx);
            if(r > (L[idx]+R[idx])/2)   insert(l, r, v, 2*idx+1);

            LL[idx] = LL[2*idx];   if(LL[idx] == (R[idx]-L[idx])/2) LL[idx] +=
LL[2*idx+1];
            RR[idx] = RR[2*idx+1]; if(RR[idx] == (R[idx]-L[idx])/2) RR[idx] +=
RR[2*idx];
            MAX[idx] = max(max(MAX[2*idx], MAX[2*idx+1]), RR[2*idx] +
LL[2*idx+1]);
        }
    }
    int query(int len) {
        int idx = 1;
        if(MAX[idx] < len) return 0;
        while(1) {
            if(V[idx] == EMPTY)                              return L[idx]+1;
            if(MAX[2*idx] >= len)                          idx = 2*idx;
            else if(RR[2*idx]+LL[2*idx+1] >= len) return R[2*idx]-RR[2*idx] + 1;
            else                                           idx = 2*idx+1;
        }
    }
} st;

int main() {
    int n, m, o, a, b;
    n = ms();m = ms();
    st.init(n);
    while(m --) {
        o=ms();  a = ms();
        if(o == 1) {
            int ans = st.query(a);
            printf("%d\n", ans);
            if(ans) {
                st.insert(ans-1, ans+a-1, FULL);
            }
        } else {
            b = ms();
            st.insert(a-1, a-1+b, EMPTY);
        }
    }
    return 0;
}

//   11.hdu1540 Tunnel Warfare
//   更新节点,询问节点所在区间(同上一道Hotel一样类型的题目)

#define SIZE 67000
#define MEM 2*SIZE+10

struct SegTree {
   int L[MEM], R[MEM];
   int V[MEM];
   int LL[MEM], RR[MEM];
   int n;
   void init(int size) {
      for(n = 1; n < size; n <<= 1);
      for(int i = n; i < 2*n; i ++) {
         L[i] = i - n;
         R[i] = i - n + 1;
         if(i < n + size) {
            V[i] = 0;
            LL[i] = RR[i] = 1;
         } else {
            V[i] = 1;
            LL[i] = RR[i] = 0;
         }
      }
      for(int i = n-1; i ; i --) {
         L[i] = L[2*i];
         R[i] = R[2*i+1];

         V[i] = 0;
         LL[i] = LL[2*i];   if(LL[2*i] == (R[i]-L[i])/2)   LL[i] += LL[2*i+1];
         RR[i] = RR[2*i+1];   if(RR[2*i+1]==(R[i]-L[i])/2)   RR[i] += RR[2*i];
      }
   }
   void insert(int l, int r, int v, int idx = 1) {
      if(l <= L[idx] && r >= R[idx]) {
         V[idx] = v;
         if(V[idx] == 0) {
            LL[idx] = RR[idx] = 1;
         } else {
            LL[idx] = RR[idx] = 0;
         }
      } else {
         if(l < (L[idx]+R[idx])/2)   insert(l, r, v, 2*idx);
         if(r > (L[idx]+R[idx])/2)   insert(l, r, v, 2*idx+1);
         LL[idx] = LL[2*idx];       if(LL[2*idx] == (R[idx]-L[idx])/2)   LL[idx]
+= LL[2*idx+1];
         RR[idx] = RR[2*idx+1];       if(RR[2*idx+1]==(R[idx]-L[idx])/2)
RR[idx] += RR[2*idx];
      }
   }
   int queryR(int l, int r, int idx = 1) {
      if(l <= L[idx] && r >= R[idx]) {
```

```
            return RR[idx];
        } else {
            int res1 = -1, res2 = -1;
            if(l < (L[idx]+R[idx])/2)    res1 = queryR(l, r, 2*idx);
            if(r > (L[idx]+R[idx])/2)    res2 = queryR(l, r, 2*idx+1);
            if(res2 != -1) {
                if(res2 == min(r, R[idx])-(L[idx]+R[idx])/2) {
                    return res2 + max(res1, 0);
                } else {
                    return res2;
                }
            } else {
                return res1;
            }
        }
    }
    int queryL(int l, int r, int idx = 1) {
//        printf("L , l = %d, r = %d, idx = %d, L = %d, R = %d\n", l, r, idx, L[idx],
R[idx]);
        if(l <= L[idx] && r >= R[idx]) {
            return LL[idx];
        } else {
            int res1 = -1, res2 = -1;
            if(l < (L[idx]+R[idx])/2)    res1 = queryL(l, r, 2*idx);
            if(r > (L[idx]+R[idx])/2)    res2 = queryL(l, r, 2*idx+1);
//            printf("L , l = %d, r = %d, idx = %d, L = %d, R = %d, res1 = %d, res2
= %d\n", l, r, idx, L[idx], R[idx], res1, res2);
            if(res1 != -1) {
                if(res1 == (L[idx]+R[idx])/2 - max(l, L[idx])) {
                    return res1 + max(res2, 0);
                } else {
                    return res1;
                }
            } else {
                return res2;
            }
        }
    }
} st;

int stk[50100], len;
int n, m;

int main() {
    while(scanf("%d%d", &n, &m) != EOF) {
        st.init(n);
        char c;    int a;
        while(m --) {
            scanf(" %c", &c);
            if(c=='D') {
                scanf("%d", &a);
                stk[len ++] = a;
                st.insert(a-1, a, 1);
            } else if(c == 'R') {
                a = stk[-- len];
                st.insert(a-1, a, 0);
            } else {
                scanf("%d", &a);
                if(st.V[st.n+a-1] == 1) {
                    printf("0\n");
                    continue;
                }
                int ans = 1;
                if(a-1 > 0) {
                    ans += st.queryR(0, a-1);
                }
                if(a < n) {
                    ans += st.queryL(a, n);
                }
                printf("%d\n", ans);
            }
        }
    }
    return 0;
}

//   12.hdu2871 Memory Control
//   hotel变形题目,三个都函数一样(vector忘记清空检查了好久)
int N;

//寻找空间，1.2.3...n个空间中，寻找连续为k的最左面的空间。
//第i个空间对应的线段树区间为[i-1, i]
//EMPTY代表没有被占用，FULL代表被占用，0 代表此线段无意义
#define SIZE 67000
#define MEM 2*SIZE+10

const int EMPTY = -99;
const int FULL = 100;
//0 代表没有插入线段

struct SegTree {
    int L[MEM], R[MEM], V[MEM];        //左，右，值
    int LL[MEM], RR[MEM], MAX[MEM];    //这段区间中左面连续最多，右面连续最多，连续最多
    int n;
    void init(int size) {
        for(n=1; n<size; n<<=1);
        for(int i = n; i < 2*n; i ++) {
            L[i] = i-n;
            R[i] = i-n+1;
        }
        for(int i = n-1; i ; i --) {
            L[i] = L[2*i];
            R[i] = R[2*i+1];
        }
        memset(V, 0, sizeof(V));
        memset(LL, 0, sizeof(LL));
        memset(RR, 0, sizeof(RR));
        memset(MAX, 0, sizeof(MAX));
        insert(0, size, EMPTY);
        insert(size, n, FULL);
    }
    void insert(int l, int r, int v, int idx = 1) {
        if(l <= L[idx] && r >= R[idx]) {
            V[idx] = v;
            LL[idx] = RR[idx] = MAX[idx] = (v==EMPTY?R[idx] - L[idx]:0);
        } else {
            if(V[idx]) {
                V[2*idx] = V[2*idx+1] = V[idx];
                LL[2*idx] = RR[2*idx] = MAX[2*idx] = MAX[idx]/2;
                LL[2*idx+1]=RR[2*idx+1]=MAX[2*idx+1]=MAX[idx]/2;
```

```
            V[idx] = 0;
        }
        if(l < (L[idx]+R[idx])/2)    insert(l, r, v, 2*idx);
        if(r > (L[idx]+R[idx])/2)    insert(l, r, v, 2*idx+1);

        LL[idx] = LL[2*idx];    if(LL[idx] == (R[idx]-L[idx])/2)    LL[idx] +=
LL[2*idx+1];
        RR[idx] = RR[2*idx+1];    if(RR[idx] == (R[idx]-L[idx])/2)    RR[idx] +=
RR[2*idx];
        MAX[idx] = max(max(MAX[2*idx], MAX[2*idx+1]), RR[2*idx] + LL[2*idx+1]);
      }
   }
   //返回可用的初始房间编号，0 为没有满足条件的
   int query(int len) {
      int idx = 1;
      if(MAX[idx] < len)    return 0;
      while(1) {
         if(V[idx] == EMPTY)                    return L[idx]+1;
         if(MAX[2*idx] >= len)                 idx = 2*idx;
         else if(RR[2*idx]+LL[2*idx+1] >= len)    return R[2*idx]-RR[2*idx] + 1;
         else                                  idx = 2*idx+1;
      }
   }
} st;

int n, m;
char cmd[100];

struct T {
   int a, b;
   bool operator < (const T & t) const {
      return a < t.a;
   }
};

vector<T> vec;

int main() {
   while(scanf("%d%d", &n, &m) != EOF) {
      N = n;
      if(N > 50000 || N <= 0)    while(1);
      st.init(n);
      vec.clear();
      int tmp;
      while(m --) {
//          printf("m = %d\n", m);
         scanf("%s", cmd);
         if(*cmd == 'R') {
            vec.clear();
            st.insert(0, n, EMPTY);
            printf("Reset Now\n");
         } else if(*cmd == 'N') {
            scanf("%d", &tmp);
            int idx = st.query(tmp);
            if(idx) {
               printf("New at %d\n", idx);
               st.insert(idx-1, idx-1+tmp, FULL);
               T ttmp;
               ttmp.a = idx;
               int i = upper_bound(vec.begin(), vec.end(), ttmp)-vec.begin();
               T tt;
               tt.a = idx;
               tt.b = idx+tmp-1;
               vec.insert(vec.begin()+i, tt);
            } else {
               printf("Reject New\n");
            }
         } else if(*cmd == 'F') {
            T tmp;
            scanf("%d", &tmp.a);
            int idx = upper_bound(vec.begin(), vec.end(), tmp)-vec.begin()-1;
            if(idx == -1 || !(vec[idx].a<= tmp.a && tmp.a <= vec[idx].b)) {
               printf("Reject Free\n");
            } else {
               printf("Free from %d to %d\n", vec[idx].a, vec[idx].b);
               st.insert(vec[idx].a-1, vec[idx].b, EMPTY);
               vec.erase(vec.begin()+idx);
            }
         } else {
            scanf("%d", &tmp);
            if(tmp > vec.size()) {
               printf("Reject Get\n");
            } else {
               printf("Get at %d\n", vec[tmp-1].a);
            }
         }
      }
      printf("\n");
   }
   return 0;
}

//   13.hdu3016 Man Down
//   成段更新,单点查询(简单线段树+简单DP)

#define SIZE 200010
#define MEM 2*SIZE+10

struct SegTree {
   int L[MEM], R[MEM], V[MEM], M[MEM];
   int n;
   void init(int size) {
      for(n = 1; n < size; n <<=1);
      for(int i = n; i < 2*n; i ++) {
         L[i] = i - n;
         R[i] = i - n + 1;
         V[i] = -1;
         M[i] = -1;
      }
      for(int i = n-1; i; i --) {
         L[i] = L[2*i];
         R[i] = R[2*i+1];
         V[i] = 0;
         M[i] = max(M[2*i], M[2*i+1]);
      }
   }
   void insert(int l, int r, int v, int idx = 1) {
      if(v <= 0)    v = -1;
      if(l <= L[idx] && r >= R[idx]) {
```

```
            V[idx] = v;
            M[idx] = v;
        } else {
            if(V[idx]) {
                V[2*idx] = V[2*idx+1] = V[idx];
                M[2*idx] = M[2*idx+1] = M[idx];
                V[idx] = 0;
            }
            if(l < (L[idx]+R[idx])/2)    insert(l, r, v, 2*idx);
            if(r > (L[idx]+R[idx])/2)    insert(l, r, v, 2*idx+1);
            M[idx] = max(M[2*idx], M[2*idx+1]);
        }
    }
    int query(int l, int r, int idx = 1) {
        if((l <= L[idx] && r >= R[idx]) || V[idx]) {
            return M[idx];
        }
        int res = -1;
        if(l < (L[idx]+R[idx])/2)    res = max(res, query(l, r, 2*idx));
        if(r > (L[idx]+R[idx])/2)    res = max(res, query(l, r, 2*idx+1));
        return res;
    }
} st;

struct T {
    int h, xl, xr, v;
    bool operator < (const T & t) const {
        return h > t.h;
    }
    void input() {
        scanf("%d%d%d%d", &h, &xl, &xr, &v);
    }
} ts[100010];

int main() {
    int n;
    while(scanf("%d", &n) != EOF) {
        st.init(100000);
        for(int i = 0; i < n; i ++) {
            ts[i].input();
        }
        sort(ts, ts+n);
        st.insert(ts[0].xl-1, ts[0].xl, 100+ts[0].v);
        st.insert(ts[0].xr-1, ts[0].xr, 100+ts[0].v);
        for(int i = 1; i < n; i ++) {
            int num = st.query(ts[i].xl-1, ts[i].xr);
            if(num == -1)    continue;
            if(ts[i].xr-1-ts[i].xl > 0)    st.insert(ts[i].xl, ts[i].xr-1, -1);
            st.insert(ts[i].xl-1, ts[i].xl, num+ts[i].v);
            st.insert(ts[i].xr-1, ts[i].xr, num+ts[i].v);
        }
        printf("%d\n", st.query(0, 100000));
    }
    return 0;
}

//   14.hdu1542 Atlantis
//   矩形面积并,扫描线法(发现我们HDU的队员的矩形面积交模板基本都是在最坏情况下更新到底,宁波惨痛的教训啊..)
#define SIZE 500
#define MEM 2*SIZE+10

double val[2*SIZE];

struct SegTree {
    int L[MEM], R[MEM], V[MEM];
    double COV[MEM];
    int n;
    void init(int size) {
        for(n = 1; n < size; n <<= 1);
        for(int i = n; i < 2*n; i ++) {
            L[i] = i - n;
            R[i] = i - n + 1;
        }
        for(int i = n-1; i ; i --) {
            L[i] = L[2*i];
            R[i] = R[2*i+1];
        }
        memset(V, 0, sizeof(V));
        memset(COV, 0, sizeof(COV));
    }
    void insert(int l, int r, int v, int idx = 1) {
        if(l <= L[idx]&&r>=R[idx]) {
            V[idx] += v;
        } else {
            if(l < (L[idx]+R[idx])/2)    insert(l, r, v, 2*idx);
            if(r > (L[idx]+R[idx])/2)    insert(l, r, v, 2*idx+1);
        }
        if(V[idx]) {
            COV[idx] = val[R[idx]] - val[L[idx]];
        } else if(idx >= n) {
            COV[idx] = 0;
        } else {
            COV[idx] = COV[2*idx] + COV[2*idx+1];
        }
    }
} st;

struct T {
    double x, y1, y2;
    int mode;    //1 进入
    bool operator < (const T & t) const {
        return x < t.x;
    }
} ts[210];
int n;

map<double,int> maps;

int main() {
    for(int idx = 1; cin >> n, n; idx ++) {
        double x1,y1,x2,y2;
        maps.clear();
        for(int i = 0; i < n; i ++) {
            cin >> x1 >> y1 >> x2 >> y2;
            if(x1 > x2)    swap(x1, x2);
            if(y1 > y2)    swap(y1, y2);
            ts[2*i].x = x1;        ts[2*i].y1 = y1;    ts[2*i].y2 = y2;    ts[2*i].mode
= 1;
            ts[2*i+1].x = x2;    ts[2*i+1].y1=y1;    ts[2*i+1].y2=y2;
```

```
ts[2*i+1].mode=-1;
          maps[y1];   maps[y2];
       }
       sort(ts, ts+2*n);
       int num = 0;
       for(map<double,int>::iterator i = maps.begin(); i != maps.end(); i ++) {
          i->second = num;
          val[num] = i->first;
          num ++;
       }
       st.init(num);
       double area = 0;
       double lastX = -1;
       for(int i = 0; i < 2*n; i ++) {
          T t = ts[i];
          area += (ts[i].x - lastX) * st.COV[1];
          if(t.mode == 1) {
             st.insert(maps[ts[i].y1], maps[ts[i].y2], 1);
          } else {
             st.insert(maps[ts[i].y1], maps[ts[i].y2], -1);
          }
          lastX = ts[i].x;
       }
       printf("Test case #%d\nTotal explored area: %.2f\n\n", idx, area);
    }
    return 0;
}

//   15.hdu1255 覆盖的面积
//   同上,扫描线法,我多加了一个系数csum,来统计覆盖两次的长度(156MS,第一次做线段树排到第一,纪念下)
#define SIZE 2100
#define MEM 2*SIZE+10

double val[2*SIZE];

struct SegTree {
    int L[MEM], R[MEM], V[MEM];
    double COV1[MEM], COV2[MEM];
    int n;
    void init(int size) {
       for(n = 1; n < size; n <<= 1);
       for(int i = n; i < 2*n; i ++) {
          L[i] = i - n;
          R[i] = i - n + 1;
       }
       for(int i = n-1; i ; i --) {
          L[i] = L[2*i];
          R[i] = R[2*i+1];
       }
       memset(V, 0, sizeof(V));
       memset(COV1, 0, sizeof(COV1));
       memset(COV2, 0, sizeof(COV2));
    }
    void insert(int l, int r, int v, int idx = 1) {
       if(l <= L[idx]&&r>=R[idx]) {
          V[idx] += v;
       } else {
          if(l < (L[idx]+R[idx])/2)    insert(l, r, v, 2*idx);
          if(r > (L[idx]+R[idx])/2)    insert(l, r, v, 2*idx+1);
       }
       if(V[idx] == 1) {
          if(idx < n) {
             COV2[idx] = COV2[idx*2]+COV2[idx*2+1]+COV1[idx*2]+COV1[idx*2+1];
             COV1[idx] = val[R[idx]] - val[L[idx]] - COV2[idx];
          } else {
             COV1[idx] = val[R[idx]] - val[L[idx]];
             COV2[idx] = 0;
          }
       } else if(V[idx]>=2) {
          COV2[idx] = val[R[idx]] - val[L[idx]];
          COV1[idx] = 0;
       } else if(idx >= n) {COV1[idx] = 0;
          COV2[idx] = 0;
       } else {
          COV1[idx] = COV1[2*idx] + COV1[2*idx+1];
          COV2[idx] = COV2[2*idx] + COV2[2*idx+1];
       }
    }
} st;

struct T {
    double x, y1, y2;
    int mode;    //1 进入
    bool operator < (const T & t) const {
       return x < t.x;
    }
    void output() {
       printf("x = %.2f, y1 = %.2f, y2 = %.2f\n", x, y1, y2);
    }
} ts[2*2100];
int n;

map<double,int> maps;

int main() {
    int t;
    for(cin >> t; t --; ) {
       cin >> n;
       double x1,y1,x2,y2;
       maps.clear();
       for(int i = 0; i < n; i ++) {
          cin >> x1 >> y1 >> x2 >> y2;
          if(x1 > x2)    swap(x1, x2);
          if(y1 > y2)    swap(y1, y2);
          ts[2*i].x = x1;       ts[2*i].y1 = y1;    ts[2*i].y2 = y2;    ts[2*i].mode
= 1;
          ts[2*i+1].x = x2;    ts[2*i+1].y1=y1;    ts[2*i+1].y2=y2;
ts[2*i+1].mode=-1;
          maps[y1];   maps[y2];
       }
       sort(ts, ts+2*n);
       int num = 0;
       for(map<double,int>::iterator i = maps.begin(); i != maps.end(); i ++) {
          i->second = num;
          val[num] = i->first;
          num ++;
       }
       st.init(num);
       double area = 0;
```

```
        double lastX = -1;
        for(int i = 0; i < 2*n; i ++) {
            T t = ts[i];


            area += (ts[i].x - lastX) * st.COV2[1];
            if(t.mode == 1) {
                st.insert(maps[ts[i].y1], maps[ts[i].y2], 1);
            } else {
                st.insert(maps[ts[i].y1], maps[ts[i].y2], -1);
            }
//            ts[i].output();
//            printf("cov1 = %.2f, cov2 = %.2f\n", st.COV1[1], st.COV2[1]);
            lastX = ts[i].x;
        }
        printf("%.2f\n", round(area*100)/100.0);
    }
    return 0;
}

//   16.hdu1828 Picture
//   扫描线,同面积统计,加了一个num_Seg统计一个扫描区域里的边
#define maxn 5010

/**
    计算矩形的面积和周长
    ----------------------------
    需要掉用线段树,
    传入:
            VerSeg (矩形的竖直线，注意y1 < y2)
            n    竖直线个数 (为矩形数的 2 倍)
*/

struct VerSeg {
    int x, y1, y2;
    int mode;//1 为进入线段，-1 为退出线段
    bool operator <(const VerSeg& vs) const {
        return x < vs.x;
    }
    void set(int x, int y1, int y2, int mode) {
        this->x = x;
        this->y1 = y1;
        this->y2 = y2;
        this->mode = mode;
    }
};
int val[2*maxn];//保存离散化后的数值

struct SegTree {
#define SIZE 2*maxn+1000
#define MEM 2*SIZE+10
#define SET(x) memset(x, 0, sizeof(x))
    int L[MEM], R[MEM], V[MEM];    //L、R:线段的区间范围，V:线段的值
    int M[MEM],C[MEM];       //M此区间包含的非零线段长度，C此区间包含的连续线段个数，S此线段的V+递归子线段的V
    bool P[MEM], Q[MEM];       //P,Q左右端点是否被覆盖，辅助C
    int n;
    void init(int size) {
        for(n = 1; n < size; n <<= 1);
        int i;
        for(i = n; i < 2*n; i ++) {
            L[i] = i-n;
            R[i] = i+1-n;
        }
        for(i = n-1; i >= 1; i --) {
            L[i] = L[2*i];
            R[i] = R[2*i+1];
        }
        SET(V);    SET(M);    SET(C);    SET(P);    SET(Q);
    }
    void update1(int i) {//设置M 和 C
        if(V[i]) {
            M[i] = val[R[i]] - val[L[i]];
        } else if(i >= n) {
            M[i] = 0;
        } else {
            M[i] = M[2*i]+M[2*i+1];
        }
    }
    void update2(int i) {//设置C
        if(V[i]) {
            P[i] = Q[i] = C[i] = 1;
        } else if(i >= n) {
            P[i] = Q[i] = C[i] = 0;
        } else {
            P[i] = P[2*i];
            Q[i] = Q[2*i+1];
            C[i] = C[2*i]+C[2*i+1] - Q[2*i]*P[2*i+1];
        }
    }
    void insert(int l, int r, int v, int i=1) {
        if(l <= L[i] && r >= R[i])    V[i] += v;
        else {
            if(l < (L[i]+R[i])/2)    insert(l, r, v, 2*i);
            if(r > (L[i]+R[i])/2)    insert(l, r, v, 2*i+1);
        }
        update1(i);
        update2(i);
    }
} st;

void rectCal(VerSeg*segs, int n, int &area, long long &perimeter) {
    map<int, int> ma;
    int i;
    int lastM, lastX;
    for(i = 0; i < n; i ++) {
        ma[segs[i].y1];       ma[segs[i].y2];
    }
    i = 0;
    memset(val, 0, sizeof(val));
    for(map<int,int>::iterator iter = ma.begin(); iter!=ma.end(); iter ++, i ++) {
        iter->second = i;
        val[i] = iter->first;
    }
    sort(segs, segs+n);
    st.init(n);
    area = perimeter = 0;
    lastX = lastM = 0;
    for(i = 0; i < n; i ++) {
        lastM = st.M[1];
```

```
        area += st.M[1] * (segs[i].x - lastX);
        perimeter += (long long)2 * st.C[1] * (segs[i].x - lastX);
        st.insert(ma[segs[i].y1], ma[segs[i].y2], segs[i].mode);
        perimeter += abs(lastM - st.M[1]);
        lastX = segs[i].x;
    }
}

VerSeg vs[2*maxn];

int main() {
    int n;
    while(scanf("%d", &n) != EOF) {
        int x1, y1, x2, y2;
        for(int i = 0; i < n; i ++) {
            cin >> x1 >> y1 >> x2 >> y2;
            if(x1 > x2)       swap(x1, x2);
            if(y1 > y2)       swap(y1, y2);
            vs[2*i].set(x1, y1, y2, 1);
            vs[2*i+1].set(x2, y1, y2, -1);
        }
        int area;
        long long peri;
        rectCal(vs, 2*n, area, peri);
        cout << peri << endl;
    }
    return 0;
}

//   17.pku1436 Horizontally Visible Segments
//   成段更新,成段询问(染色问题,坐标*2 后很简单,最后统计用暴力- -)
set<int> sets;

int vis[8010][260];

#define SIZE 17000
#define MEM 2*SIZE+10
struct SegTree {
     int L[MEM], R[MEM], V[MEM];
     int n;
     void init(int size) {
          for(n = 1; n < size; n <<= 1);
          for(int i = n; i < 2*n; i ++) {
               L[i] = i - n;
               R[i] = i - n + 1;
          }
          for(int i = n-1; i; i --) {
               L[i] = L[2*i];
               R[i] = R[2*i+1];
          }
          memset(V, 0, sizeof(V));
          insert(0, n, -1);
     }
     void insert(int l, int r, int v, int idx = 1) {
          if(l <= L[idx] && r >= R[idx]) {
               V[idx] = v;
          } else {
               if(V[idx]) {
                    V[2*idx] = V[2*idx+1] = V[idx];
                    V[idx] = 0;
               }
               if(l < (L[idx]+R[idx])/2)   insert(l, r, v, 2*idx);
               if(r > (L[idx]+R[idx])/2)   insert(l, r, v, 2*idx+1);
          }
     }
     void search(int l, int r, int idx = 1) {
          if(V[idx]) {
               if(V[idx] != -1) {
                    sets.insert(V[idx]);
               }
          } else {
               if(l < (L[idx]+R[idx])/2)   search(l, r, 2*idx);
               if(r > (L[idx]+R[idx])/2)   search(l, r, 2*idx+1);
          }
     }
} st;

struct T {
     int x1, x2, h;
     void input() {
          scanf("%d%d%d", &x1, &x2, &h);
     }
     bool operator < (const T & t) const {
          return h < t.h;
     }
     /*void output() {
          printf("x1 = %d, x2 = %d, h = %d\n", x1, x2, h);
     }*/
} ts[8010];
int n;

bool isVis(int i, int j) {
     return ((vis[i][j/32] >> (j%32)) | (vis[j][i/32] >> (i%32))) & 1;
}

void setVis(int i, int j) {
     vis[i][j/32] |= (1<<(j%32));
     vis[j][i/32] |= (1<<(i%32));
}

int main() {
     int t;
     for(scanf("%d", &t); t --; ) {
          memset(vis, 0, sizeof(vis));
          st.init(16010);
          scanf("%d", &n);
          for(int i = 1; i <= n; i ++) {
               ts[i].input();
          }
          sort(ts+1, ts+n+1);
          int ans = 0;
          for(int idx = 1; idx <= n; idx ++) {
//             printf("idx = %d\n", idx);

               sets.clear();
//             printf("clear over!\n");
               st.search(2*ts[idx].x1, 2*ts[idx].x2+1);
//             printf("search over!\n");
//             ts[idx].output();
               for(set<int>::iterator i = sets.begin(); i != sets.end(); i ++) {
```

```
                    set<int>::iterator j = i;
                    j ++;
                    for(; j != sets.end(); j ++) {
                        if(isVis(*i, *j))  ans ++;
                    }
                    setVis(idx, *i);
//                  printf("%d ", *i);
                }
//              printf("\n");
                st.insert(2*ts[idx].x1, 2*ts[idx].x2+1, idx);
            }
            printf("%d\n", ans);
        }
        return 0;
}

//   18.pku3225 Help with Intervals
//   成段更新,总询问区间(有个异或操作比较新颖)
#define SIZE 67000*2
#define MEM 2*SIZE+10

int arr[MEM];

struct SegTree {
    int L[MEM], R[MEM], V[MEM];
    int n;
    void init(int size) {
        for(n=1; n<size; n<<=1);
        for(int i = n; i < 2*n; i ++) {
            L[i] = i-n;
            R[i] = i-n+1;
        }
        for(int i = n-1; i; i --) {
            L[i] = L[2*i];
            R[i] = R[2*i+1];
        }
        memset(V, 255, sizeof(V));
    }
    void insert(int l, int r, int v, int idx = 1) {
        if(l <= L[idx] && r >= R[idx]) {
            V[idx] = v;
        } else {
            if(V[idx] == 2) {
                if(V[2*idx] == -1) {
                    V[2*idx] = 2;
                } else if(V[2*idx] == 2) {
                    V[2*idx] = -1;
                } else {
                    V[2*idx] ^= 1;
                }
                if(V[2*idx+1] == -1) {
                    V[2*idx+1] = 2;
                } else if(V[2*idx+1] == 2) {
                    V[2*idx+1] = -1;
                } else {
                    V[2*idx+1] ^= 1;
                }
            } else if(V[idx] != -1) {
                V[2*idx] = V[2*idx+1] = V[idx];
            }
            V[idx] = -1;
            if(l < (L[idx]+R[idx])/2)   insert(l, r, v, 2*idx);
            if(r > (L[idx]+R[idx])/2)   insert(l, r, v, 2*idx+1);
        }
    }

    void xxx(int l, int r, int idx = 1) {
        if(l <= L[idx] && r >= R[idx]) {
            if(V[idx] == -1)        V[idx] = 2;
            else if(V[idx] == 2)    V[idx] = -1;
            else                    V[idx] ^= 1;
        } else {
            if(V[idx] == 2) {
                if(V[2*idx] == -1) {
                    V[2*idx] = 2;
                } else if(V[2*idx] == 2) {
                    V[2*idx] = -1;
                } else {
                    V[2*idx] ^= 1;
                }
                if(V[2*idx+1] == -1) {
                    V[2*idx+1] = 2;
                } else if(V[2*idx+1] == 2) {
                    V[2*idx+1] = -1;
                } else {
                    V[2*idx+1] ^= 1;
                }
            } else if(V[idx] != -1) {
                V[2*idx] = V[2*idx+1] = V[idx];
            }
            V[idx] = -1;
            if(l < (L[idx]+R[idx])/2)   xxx(l, r, 2*idx);
            if(r > (L[idx]+R[idx])/2)   xxx(l, r, 2*idx+1);
        }
    }
    void format(int idx = 1) {
        if(V[idx] == 1 || V[idx] == 0) {
            for(int i = L[idx]; i < R[idx]; i ++) {
                arr[i] = V[idx];
            }
            return;
        }
        if(V[idx] == 2) {
            if(V[2*idx] == -1) {
                V[2*idx] = 2;
            } else if(V[2*idx] == 2) {
                V[2*idx] = -1;
            } else {
                V[2*idx] ^= 1;
            }
            if(V[2*idx+1] == -1) {
                V[2*idx+1] = 2;
            } else if(V[2*idx+1] == 2) {
                V[2*idx+1] = -1;
            } else {
                V[2*idx+1] ^= 1;
            }
        }
        format(2*idx);
        format(2*idx+1);
```

```
    }
} st;

void print(int i, int j) {
    if(i & 1) {
        printf("(");
    } else {
        printf("[");
    }
    printf("%d,%d", i/2, (j+1)/2);
    if(j & 1) {
        printf(")");
    } else {
        printf("]");
    }
    printf(" ");
}

int main() {
    char opr, l, r;
    int a, b;
    st.init(131072);
    st.insert(0, 131072, 0);
    while(scanf(" %c %c%d,%d%c", &opr, &l, &a, &b, &r) != EOF) {
        a = 2*a; b = 2*b;
        if(l == '(')  a ++;
        if(r == ']')  b ++;
        if(opr == 'U') {
            st.insert(a, b, 1);
        } else if(opr == 'I') {
            st.insert(0, a, 0);
            st.insert(b, st.n, 0);
        } else if(opr == 'D') {
            st.insert(a, b, 0);
        } else if(opr == 'C') {
            st.insert(0, a, 0);
            st.insert(b, st.n, 0);
            st.xxx(a, b);
        } else if(opr == 'S') {
            st.xxx(a, b);
        } else {
//            break;
        }
    }
    st.format();
    int num = 0;
    /*for(int i = 0; i < 20; i ++) {
        printf("%d ", arr[i]);
    }
    printf("\n");*/
    for(int i = 0; i < 131072; i ++) {
        if(arr[i] == 0)    continue;
        int j;
        for(j = i; j < 131072 && arr[j]; j ++);
        print(i, j-1);
        i = j;
        num ++;
    }
    if(num == 0)  printf("empty set");
    printf("\n");
    return 0;
}

//   19.pku2482 Stars in Your Window
//   成段更新,区间最值 + 扫描线(转化成区间最值有点巧妙,数据太TMD的恶心了,中间二分的地方会int
溢出,检查了半个小时)
#define SIZE 17384
#define MEM 2*SIZE+10

struct SegTree {
    int L[MEM], R[MEM], V[MEM], MAX[MEM];
    int n;
    void init(int size) {
        for(n = 1; n < size; n <<=1);
        for(int i = n; i < 2*n; i ++) {
            L[i] = i-n;
            R[i] = i-n+1;
        }
        for(int i = n-1; i ; i --) {
            L[i] = L[2*i];
            R[i] = R[2*i+1];
        }
        memset(V, 0, sizeof(V));
        memset(MAX, 0, sizeof(MAX));
    }
    void insert(int l, int r, int v, int i = 1) {
        /*if(i == 1) {
            printf("l = %d, r = %d, v = %d\n", l, r, v);
        }*/
        if(l <= L[i] && r >= R[i]) {
            V[i] += v;
        } else {
            if(l < (L[i]+R[i])/2)   insert(l, r, v, 2*i);
            if(r > (L[i]+R[i])/2)   insert(l, r, v, 2*i+1);
        }
        if(i >= n) {
            MAX[i] = V[i];
        } else {
            MAX[i] = max(MAX[2*i], MAX[2*i+1]) + V[i];
        }
    }
} st;

int n, w, h;

const int inf = 0x3f3f3f3f;

struct T {
    int x, y, c;
    int l, r;
    void input() {
        scanf("%d%d%d", &x, &y, &c);
    }
    void output() {
        printf("x = %d, y = %d, c = %d, l = %d, r = %d\n", x, y, c, l, r);
    }
} ts[10010];

int cmp1(const T & a, const T & b) {
    return a.x < b.x;
```

```
}
int cmp2(const T & a, const T & b) {
     return a.y < b.y;
}

int main() {
     while(scanf("%d%d%d", &n, &w, &h) != EOF) {
          for(int i = 0; i < n; i ++) {
               ts[i].input();
          }
          sort(ts, ts+n, cmp1);

          int l = 0;
          for(int i = 0; i < n; i ++) {
               while(ts[i].x-ts[l].x >= w) {
                    l ++;
               }
               ts[i].l = l;
               ts[i].r = i;
          }
          st.init(n);
          st.insert(0, st.n, -inf);

          sort(ts, ts+n, cmp2);
          int u = 0;
          int ans = 0;
          for(int i = 0; i < n; i ++) {
               while(ts[i].y - ts[u].y >= h) {
                    st.insert(ts[u].r, ts[u].r+1, -inf);
                    st.insert(ts[u].l, ts[u].r+1, -ts[u].c);
                    u ++;
               }
               st.insert(ts[i].r, ts[i].r+1, inf);
               st.insert(ts[i].l, ts[i].r+1, ts[i].c);

               if(st.MAX[1] < 0 || st.MAX[1] >= inf) {
                    while(1);
               }

               ans = max(ans, st.MAX[1]);
          }
          printf("%d\n", ans);
     }
     return 0;
}
/**
 *
4 2 1
0 0 9
0 1 10
1 0 11
1 1 12
33
 */

//   20.pku2828 Buy Tickets
//   思维很巧妙,倒过来做的话就能确定此人所在的位置
#define SIZE 270000
#define MEM 2*SIZE+10
```

```
struct SegTree {
     int L[MEM], R[MEM], V[MEM];
     int n;
     void init(int size) {
          for(n = 1; n < size; n <<= 1);
          for(int i = n; i < 2*n; i ++) {
               L[i] = i-n;
               R[i] = i-n+1;
               V[i] = 1;
          }
          for(int i = n-1; i ; i --) {
               L[i] = L[2*i];
               R[i] = R[2*i+1];
               V[i] = V[2*i]+V[2*i+1];
          }
     }
     int query(int v, int i = 1) {
          if(i >= n) {
               V[i] --;
               return L[i];
          }
          int res;
          if(v <= V[2*i]) {
               res = query(v, 2*i);
          } else {
               res = query(v-V[2*i], 2*i+1);
          }
          V[i] = V[2*i]+V[2*i+1];
          return res;
     }
} st;

int pos[SIZE], val[SIZE];
int ans[SIZE];
int n;

int main() {
     while(scanf("%d", &n) != EOF) {
          for(int i = 0; i < n; i ++) {
               scanf("%d%d", pos+i, val+i);
          }
          st.init(n);
          for(int i = n-1; i >= 0; i --) {
               ans[st.query(pos[i]+1)] = val[i];
          }
          for(int i = 0; i < n; i ++) printf("%d ", ans[i]);
          printf("\n");
     }
     return 0;
}

//   21.pku2464 Brownie Points II
//   更新节点,区间求和 + 扫描线(很好玩的一道题目,用两个线段树沿着扫描线更新,然后按"自己最多,对方最少"的方案一路统计)
//   (因为两棵树,我就直接写成类了)
struct Point {
     int x,y;
     void input() {
          scanf("%d%d", &x, &y);
     }
```

```
    bool operator < (const Point & p) const {
        return x < p.x;
    }
    void output() {
        printf("x = %d, y = %d\n", x, y);
    }
} ps[200010];
int n;

map<int,int> maps;

#define SIZE 200100

#define lowbit(k) k & -k
struct TreeArr {
    int arr[SIZE+10];
    void inc(int k, int num) {
        for(; k <= SIZE; k += lowbit(k)) {
            arr[k] += num;
        }
    }
    int get(int k) {
        int res = 0;
        for(; k; k -= lowbit(k)) {
            res += arr[k];
        }
        return res;
    }
} ta1, ta2;

int ll[200010];

int main() {
    while(scanf("%d", &n), n) {
        maps.clear();
        memset(ta1.arr, 0, sizeof(ta1.arr));
        memset(ta2.arr, 0, sizeof(ta2.arr));
        for(int i = 0; i < n; i ++) {
            ps[i].input();
            maps[ps[i].y];
        }
        int y = 0;
        for(map<int,int>::iterator i = maps.begin(); i != maps.end(); i ++) {
            i->second = ++ y;
        }
        for(int i = 0; i < n; i ++) {
            ps[i].y = maps[ps[i].y];
            ta2.inc(ps[i].y, 1);
        }
        sort(ps, ps+n);
        /*for(int i = 0; i < n; i ++) {
            ps[i].output();
        }*/
        int i, j;

        int ans = -1;
        int len = 0;

        for(i = 0; i < n; i = j) {
            int minVal = 999999999;                    //san
            int maxVal = -1;                           //ool
            for(j = i; j < n && ps[j].x==ps[i].x; j ++) {
                ta2.inc(ps[j].y, -1);
            }
            for(j = i; j < n && ps[j].x==ps[i].x; j ++) {
                minVal = min(minVal,
ta1.get(ps[j].y-1)+ta2.get(SIZE)-ta2.get(ps[j].y));
                maxVal = max(maxVal,
ta2.get(ps[j].y-1)+ta1.get(SIZE)-ta1.get(ps[j].y));
            }
            for(j = i; j < n && ps[j].x==ps[i].x; j ++) {
                ta1.inc(ps[j].y, 1);
            }
            if(minVal > ans) {
                len = 0;
                ans = minVal;
            }
            if(minVal == ans) {
                ll[len ++] = maxVal;
            }
        }
        sort(ll, ll+len);
        printf("Stan: %d; Ollie:", ans);
        for(int i = 0; i < len; i ++) {
            if(i && ll[i]==ll[i-1])continue;
            printf(" %d", ll[i]);
        }
        printf(";\n");
    }
    return 0;
}

//   22.pku3145 Harmony Forever
//   查找一个区间内最左边的数,询问的val大的话用线段树,小的话线性扫描,很囧的题目
#define SIZE 530000
#define MEM 2*SIZE+10

int addr[530000];
int arr[530000];

const int inf = 0x3f3f3f3f;

struct SegTree {
    int L[MEM], R[MEM], V[MEM];
    int n;
    void init(int size) {
        for(n = 1; n < size; n <<=1);
        for(int i = n; i < 2*n; i ++) {
            L[i] = i - n;
            R[i] = i - n + 1;
        }
        for(int i = n-1; i; i --) {
            L[i] = L[2*i];
            R[i] = R[2*i+1];
        }
        memset(V, 63, sizeof(V));
    }
    void insert(int l, int r, int idx = 1) {
        if(l <= L[idx] && r >= R[idx]) {
            V[idx] = L[idx];
```

```
          } else {
               if(l < (L[idx]+R[idx])/2)    insert(l,r,2*idx);
               if(r > (L[idx]+R[idx])/2)    insert(l,r,2*idx+1);
               V[idx] = V[2*idx] < V[2*idx+1]? V[2*idx] : V[2*idx+1];//min(V[2*idx],
V[2*idx+1]);
          }
     }
     int query(int l, int r, int idx = 1) {
          if(l <= L[idx] && r >= R[idx])   return V[idx];
          int tmp = inf;
          if(l < (L[idx]+R[idx])/2) {
               tmp = query(l,r,2*idx);
               if(tmp != inf)     return tmp;
          }
          if(r > (L[idx]+R[idx])/2)    tmp = query(l,r,2*idx+1);
          return tmp;
     }
} st;

int n;

int main() {
     for(int index = 1; scanf("%d", &n), n; index ++) {
          if(index != 1)     printf("\n");
          printf("Case %d:\n", index);
          st.init(500010);
//        printf("%d\n", st.n);
          int idx = 0;
          char c;
          int num;
          for(int i = 0; i < n; i ++) {
               scanf(" %c%d", &c, &num);
               if(c == 'B') {
                    idx ++;
                    arr[idx] = num;
                    addr[num] = idx;
                    st.insert(num,num+1);
               } else {
                    if(idx == 0) {
                         printf("-1\n");
                    } else {
                         int ans;
                         if(num < 2500) {
                              int val = inf;
                              for(int j = idx; j >= 1; j --) {
                                   if(arr[j] % num < val) {
                                        val = arr[j] % num;
                                        ans = j;
                                        if(val == 0)  break;
                                   }
                              }
                         } else {
                              int val = inf;
                              int bound, tmp;
                              for(int j = 0; j < st.n; j = bound) {
                                   bound = min(j+num, st.n);
                                   tmp = st.query(j, bound);
                                   if(tmp == inf)     continue;
                                   if(tmp-j < val || (tmp-j==val && addr[tmp]>ans))
{
                                        ans = addr[tmp];
                                        val=tmp-j;
                                   }
                              }
                         }
                         printf("%d\n", ans);
                    }
               }
          }
     }
     return 0;
}


//   23.pku2886 Who Gets the Most Candies?
//   寻找区间中的左数第N个数,约瑟夫环(学到了反素数表,可以不用把所有数字预处理出来了)
int invPrime[41] = {
          1,2,4,6,12,24,36,48,60,120,
          180,240,360,720,840,1260,1680,2520,5040,7560,
          10080,15120,20160,25200,27720,45360,50400,55440,83160,110880,

     166320,221760,277200,332640,498960,554400,665280,720720,1081080,1441440,
          2162160
};
int invCnt[41] = {
          1,2,3,4,6,8,9,10,12,16,
          18,20,24,30,32,36,40,48,60,64,
          72,80,84,90,96,100,108,120,128,144,
          160,168,180,192,200,216,224,240,256,288,
          320
};


int ms() {
     int res = 0;
     char c;
     while(c = getchar(), c>'9'||c<'0')    if(c==EOF)     return -1;
     for(res = c-'0'; c = getchar(), c>='0' && c <= '9'; res = res*10+c-'0');
     return res;
}

#define SIZE 524300
#define MEM 2*SIZE+10
struct SegTree {
     int L[MEM], R[MEM], V[MEM];
     int n;
     void init(int size) {
          for(n = 1; n < size; n <<= 1);
          for(int i = n; i <= 2*n; i ++) {
               L[i] = i-n;
               R[i] = i-n+1;
               V[i] = 1;
          }
          for(int i = n-1; i ; i --) {
               L[i] = L[2*i];
               R[i] = R[2*i+1];
               V[i] = V[2*i]+V[2*i+1];
          }
     }
     void insert(int l, int r, int v, int idx = 1) {
```

```
        if(l <= L[idx] && r >= R[idx]) {
            V[idx] += v;
        } else {
            if(l < (L[idx]+R[idx])/2)   insert(l,r,v,2*idx);
            if(r > (L[idx]+R[idx])/2)   insert(l,r,v,2*idx+1);
            V[idx] = V[2*idx]+V[2*idx+1];
        }
    }
    int query1(int l, int r, int idx = 1) {
        if(l <= L[idx] && r >= R[idx]) {
            return V[idx];
        } else {
            int res = 0;
            if(l < (L[idx]+R[idx])/2)   res += query1(l,r,2*idx);
            if(r > (L[idx]+R[idx])/2)   res += query1(l,r,2*idx+1);
            return res;
        }
    }
    int query2(int v, int idx = 1) {
        if(idx >= n) {
            if(V[idx] != 1)    while(1);
            return L[idx];
        } else {
            if(V[2*idx]>=v) {
                return query2(v, 2*idx);
            } else {
                return query2(v-V[2*idx], 2*idx+1);
            }
        }
    }
} st;

char str[500010][15];
int num[500010];

int main() {
    int n, k;
    while(scanf("%d%d", &n, &k) != EOF) {
        for(int i = 0; i < n; i ++) {
            scanf("%s%d", str[i], num+i);
        }
        st.init(n);
        k --;
        int idx = 0;
        while(invPrime[idx+1]<=n)   idx ++;
        for(int i = 1; i < invPrime[idx]; i ++) {
            st.insert(k, k+1, -1);
            int leftNum = st.query1(0, k);
            int sum;
            if(num[k] < 0)     sum = leftNum+num[k]+1;
            else          sum = leftNum+num[k];
            sum = ((sum-1)%(n-i)+n-i)%(n-i)+1;
//          printf("k = %d, leftNum = %d, sum = %d\n", k, leftNum, sum);
            k = st.query2(sum);
        }
        printf("%s %d\n", str[k], invCnt[idx]);
    }
    return 0;
}
```

```
//   24.pku2991 Crane
//   记录了线段的两端点以及转过的角度,成段的转,超有意思的题目,做了之后会对线段树理解更深刻
//   (wy教主推荐的,不过我的代码写的太搓了..没啥学习的价值)
#define M_PI 3.14159265358979323846
#define SIZE 17000
#define MEM 2*SIZE+10

int LEN[MEM];

struct Point {
    double x, y;
    void init(double x, double y) {
        this->x = x;
        this->y = y;
    }
    void output() {
        printf("%.2f %.2f\n", x, y);
    }
};

Point rotate(Point p, double radian) {
    radian = radian * M_PI / 180.0;
    double c = cos(radian), s = sin(radian);
    p.init(p.x*c-p.y*s, p.y*c+p.x*s);
    return p;
}

struct SegTree {
    int L[MEM], R[MEM], ANG[MEM];
    Point P[MEM];
    int n;
    void update(int idx) {
        Point tmp1 = rotate(P[2*idx], ANG[2*idx]);
        Point tmp2 = rotate(P[2*idx+1], ANG[2*idx+1]);
        P[idx].init(tmp1.x+tmp2.x, tmp1.y+tmp2.y);
    }
    void init(int size) {
        for(n=1; n<size; n<<=1);
        for(int i = n; i < 2*n; i ++) {
            L[i] = i - n;
            R[i] = i - n + 1;
            if(i < size+n) {
                ANG[i] = 90;
                P[i].init(LEN[i-n], 0);
            } else {
                ANG[i] = 0;
                P[i].init(0, 0);
            }
        }
        for(int i = n-1; i; i --) {
            L[i] = L[2*i];
            R[i] = R[2*i+1];
            ANG[i] = 0;
            update(i);
        }
    }
    void insert(int l, int r, int v, int idx = 1) {
        if(l <= L[idx] && r >= R[idx]) {
            ANG[idx] = (ANG[idx]+v)%360;
        } else {
```

```
            if(l < (L[idx]+R[idx])/2)   insert(l, r, v, 2*idx);
            if(r > (L[idx]+R[idx])/2)   insert(l, r, v, 2*idx+1);
            update(idx);
        }
    }
    int query(int l) {
        int res = 0;
        for(int idx = 1; idx < 2*n; ) {
            res = (res+ANG[idx])%360;
            if(l < (L[idx]+R[idx])/2)   idx = 2*idx;
            else                        idx = 2*idx+1;
        }
        return res;
    }
    void output() {
        for(int i = 1; i < 2*n; i ++) {
            printf("i = %d, L = %d, R = %d, ANG = %d\tPoint = ", i, L[i], R[i],
ANG[i]);
            P[i].output();
        }
        printf("--------------output over!!------------------\n");
    }
} st;

int n, m;
int a, b;
int main() {
    while(scanf("%d%d", &n, &m) != EOF) {
        for(int i = 0; i < n; i ++) scanf("%d", LEN+i);
        st.init(n);
        while(m --) {
            scanf("%d%d", &a, &b);
            int t1 = st.query(a-1);
            int t2 = st.query(a);
            int t22 = b+t1-180;
            double rot = t22-t2;
            st.insert(a, n, rot);
            st.P[1].output();
        }
        printf("\n");
    }
    return 0;
}

//  25.hdu1823 Luck and Love
//  二维线段树,没啥意思,代码量大了一倍而已,题目和运用范围都没一维的广,随便找了道题目练习下就好
#define SIZE1 130
#define MEM1 2*SIZE1+10

struct SubTree {
   int L[MEM1], R[MEM1], V[MEM1];
   int n2;
   void init(int size2) {
      for(n2=1; n2<size2; n2<<=1);
      for(int i = n2; i < 2*n2; i ++) {
         L[i] = i - n2;
         R[i] = i - n2 + 1;
      }
      for(int i = n2-1; i; i --) {
         L[i] = L[2*i];
         R[i] = R[2*i+1];
      }
      memset(V, 255, sizeof(V));
   }
   void insert(int l, int r, int v, int idx = 1) {
      if(l <= L[idx] && r >= R[idx]) {
         V[idx] = max(V[idx], v);
      } else {
         if(l < (L[idx]+R[idx])/2)   insert(l, r, v, 2*idx);
         if(r > (L[idx]+R[idx])/2)   insert(l, r, v, 2*idx+1);
         V[idx] = max(V[2*idx], V[2*idx+1]);
      }
   }
   int query(int l, int r, int idx = 1) {
      if(l <= L[idx] && r >= R[idx]) {
         return V[idx];
      } else {
         int res = -1;
         if(l < (L[idx]+R[idx])/2)   res = max(res, query(l, r, 2*idx));
         if(r > (L[idx]+R[idx])/2)   res = max(res, query(l, r, 2*idx+1));
         return res;
      }
   }
};

#define SIZE2 1100
#define MEM2 2*SIZE2+10

struct SegTree {
   int L[MEM2], R[MEM2];
   SubTree SUB[MEM2];
   int n1;
   void init(int size1, int size2) {
      for(n1=1; n1<size1; n1<<=1);
      for(int i = n1; i < 2*n1; i ++) {
         L[i] = i - n1;
         R[i] = i - n1 + 1;
         SUB[i].init(size2);
      }
      for(int i = n1; i; i --) {
         L[i] = L[2*i];
         R[i] = R[2*i+1];
         SUB[i].init(size2);
      }
   }
   void insert(int l1, int r1, int l2, int r2, int v, int idx = 1) {
      SUB[idx].insert(l2, r2, v);
      if(idx >= n1)   return;
      if(l1 < (L[idx]+R[idx])/2)   insert(l1, r1, l2, r2, v, 2*idx);
      if(r1 > (L[idx]+R[idx])/2)   insert(l1, r1, l2, r2, v, 2*idx+1);
   }
   int query(int l1, int r1, int l2, int r2, int idx = 1) {
      if(l1 <= L[idx] && r1 >= R[idx]) {
         return SUB[idx].query(l2, r2);
      } else {
         int res = -1;
         if(l1 < (L[idx]+R[idx])/2)   res = max(res, query(l1, r1, l2, r2, 2*idx));
         if(r1 > (L[idx]+R[idx])/2)   res = max(res, query(l1, r1, l2, r2,
2*idx+1));
         return res;
```

```
        }
    }
} st;
int n;

int main() {
    double a, b;
    int x1, y1, x2, y2, v;
    while(scanf("%d", &n), n) {
        st.init(1010, 110);
        while(n --) {
            char c;
            scanf(" %c", &c);
            if(c == 'I') {
                scanf("%d%lf%lf", &x1, &a, &b);
                x1 -= 100;
                y1 = (int)round(a*10.0);
                v = (int)round(b*10.0);
                st.insert(y1, y1+1, x1, x1+1, v);
            } else {
                scanf("%d%d%lf%lf", &x1, &x2, &a, &b);
                x1 -= 100;    x2 -= 100;
                if(x1 > x2)    swap(x1, x2);
                y1 = (int)round(a*10.0);
                y2 = (int)round(b*10.0);
                if(y1 > y2)    swap(y1, y2);
                int ans = st.query(y1, y2+1, x1, x2+1);
                if(ans == -1) {
                    printf("-1\n");
                } else {
                    printf("%.1f\n", ans/10.0);
                }
            }
        }
    }
    return 0;
}
```

后缀数组

```
//----------------------hdu 3553-----------------------------
/*Just a String
Problem Description
     A substring of a string is a successive part of the string. Given a string your
task is to find out the K-th alphabet order substring of the original string.
     For example, the string "ABC" contains six substrings:
     "A" "AB" "ABC" "B" "BC" "C" (in alphabet order)
     and string "BBC" also contains six substrings:
     "B" "B" "BB" "BBC" "BC" "C" (in alphabet order)
Input
     The input contains several test cases, the first line of input contains the number
of test cases.
     For each test case, there is only one line contains the string S of length N,
followed by a integer K. (1 <= K <= N*(N+1)/2, 1 <= N <= 100000, S contains only letters
and digits)
Output
     For each test cases, output the K-th alphabet order substring.
Sample Input
     2
     ABC 2
     BBC 3
Sample Output
     Case 1: AB
     Case 2: BB
*/

/**
报告:
Problem. Just A String (推荐度: ★★★★★)
    这题本人十分推荐,        虽然不难但是确实表达了后缀系列在字符串处理中的重要性。  字符
串后缀树是一个字典树,          对于字典树的先根遍历就是按照所有串的字典序进行的。  所以简单
地说,这题就是在后缀树中遍历一次,找到第 k 大的子串即可。
    这里我们不说后缀树的问题,          同样我们可以使用后缀数组和高度数组得到解。模拟在后
缀树中的递归过程,        每次寻找高度数组最小的一个元素,    然后分为类似左右子树的两个区间
分别递归。对于求高度数组最小的元素,最好写的办法是构造高度数组的最小根笛卡尔树。
事实上,高度数组的最小根笛卡尔树和后缀树仅有一步之遥,稍加改动也就是后缀树了。
*/

#include <cstdio>
#include <cstring>
#include <algorithm>

using namespace std;

#define M 101000

char data[M],_data[M];
int SA[M],rank[M];
int h[M],height[M];
int n,value;

#define Rank(a)((a)>n?0:rank[a])

bool cmp(int a,int b)
{
    return data[a] < data[b];
}

int c[M],a[M],_SA[M],_rank[M];

void Double()
{
    memset(c,0,sizeof(int) * (n + 1));
    int i;
    for (i=1;i<=n;i++) c[Rank(i + value)]++;
    a[0] = 1;for (i=1;i<=n;i++) a[i] = a[i - 1] + c[i - 1];
    for (i=1;i<=n;i++) _SA[a[Rank(i + value)]++] = i;
    memset(c,0,sizeof(int) * (n + 1));
    for (i=1;i<=n;i++) c[Rank(i)]++;
    a[0] = 1;for (i=1;i<=n;i++) a[i] = a[i - 1] + c[i - 1];
    for (i=1;i<=n;i++) SA[a[Rank(_SA[i])]++] = _SA[i];
    _rank[SA[1]] = 1;
    for (i=2;i<=n;i++) if (Rank(SA[i]) == Rank(SA[i - 1]) && Rank(SA[i] + value)
== Rank(SA[i - 1] + value))
            _rank[SA[i]] = _rank[SA[i - 1]]; else _rank[SA[i]] = _rank[SA[i - 1]] + 1;
    memcpy(rank,_rank,sizeof(int) * (n + 1));
    value <<= 1;
```

```
}

void make_SA()
{
     int i;
     for (i=1;i<=n;i++) SA[i] = i;
     sort(SA + 1,SA + n + 1,cmp);
     rank[SA[1]] = 1;
     for (i=2;i<=n;i++) if (data[SA[i]] == data[SA[i - 1]]) rank[SA[i]] = rank[SA[i
- 1]]; else rank[SA[i]] = rank[SA[i - 1]] + 1;
     value = 1;
     while (value < n) Double();
}

void make_Height()
{
     int i;
     memset(h,0,sizeof(int) * (n + 1));
     for (i=1;i<=n;i++)
          if (rank[i] == 1) h[i] = 0;
          else if (i == 1 || h[i - 1] <= 1) while (data[i + h[i]] == data[SA[rank[i]
- 1] + h[i]]) h[i]++;
          else
          {
               h[i] = h[i - 1] - 1;
               while (data[i + h[i]] == data[SA[rank[i] - 1] + h[i]]) h[i]++;
          }
     for (i=1;i<=n;i++) height[rank[i]] = h[i];
}

int left[M], right[M], father[M], root;

void make_Descartes()
{
     for(int i=1;i<=n;i++)
     {
          int f = i-1;
          father[i] = f;
          left[i] = right[i] = 0;
          right[f] = i;
          while(father[i]>0 && height[i]<height[father[i]])
          {
               right[f] = left[i];
               father[left[i]] = right[f];
               left[i] = f;
               f = father[f];
               father[left[i]] = i;
               right[f] = i;
               father[i] = f;
          }
     }
     for(int i=1;i<=n;i++)
          if(father[i]==0) root = i;
}

char ans[M];
long long k;
int L[M], R[M], P[M], D[M], U[M], top;

void make_Answer(int l,int r,int p,int d)
{
     top = 1; L[top] = l; R[top] = r; P[top] = p; D[top] = d; U[top] = 0;
     while(top)
     {
          if(U[top]==2) top--;
          else if(U[top]==0)
          {
               l = L[top]; r = R[top]; p = P[top]; d = D[top]; U[top]++;
               int h = (p==0?(n-SA[l]+1):height[p]);
               if(k <= (r-l+1ll) * (h - d))
               {
                    for(int i=0;i<=(k-1)/(r-l+1);i++)
                         ans[i+d] = data[SA[l]+i+d];
                    ans[(k-1)/(r-l+1)+1+d] = 0;
                    return;
               }
               k -= (r-l+1ll) * (h - d);
               if(p!=0)
               {
                    for(int i=d;i<h;i++) ans[i] = data[SA[l]+i];
                    top++; L[top] = l; R[top] = p-1; P[top] = left[p]; D[top] =
height[p]; U[top] = 0;
               }
               else U[top] = 2;
          }
          else if(U[top]==1)
          {
               l = L[top]; r = R[top]; p = P[top]; d = D[top]; U[top]++;
               top++; L[top] = p; R[top] = r; P[top] = right[p]; D[top] = height[p];
U[top] = 0;
          }
     }
//   printf("make_Answer(%d, %d, %d, %d)\n",l,r,p,d);
/*   int h = (p==0?(n-SA[l]+1):height[p]);
     if(k <= (r-l+1ll) * (h - d))
     {
          for(int i=0;i<=(k-1)/(r-l+1);i++)
               ans[i+d] = data[SA[l]+i+d];
          ans[(k-1)/(r-l+1)+1+d] = 0;
          return true;
     }
     k -= (r-l+1ll) * (h - d);
     if(p!=0)
     {
          for(int i=d;i<h;i++) ans[i] = data[SA[l]+i];
          if(make_Answer(l,p-1,left[p],height[p])) return true;
          if(make_Answer(p,r,right[p],height[p])) return true;
     }
     return false;*/
}

int main()
{
//   freopen("just.in","r",stdin);
//   freopen("just.ans","w",stdout);
     int ntest;
     scanf("%d",&ntest);
     for(int test=1;test<=ntest;test++)
     {
          scanf("%s%I64d",data+1,&k); n = strlen(data+1);
```

```
        if(!(n>=1 && n<=100000)) while(1);
        if(!(k>=1 && k<=n*(n+1ll)/2)) while(1);
        make_SA(); make_Height();
        make_Descartes();
        make_Answer(1,n,root,0);
        printf("Case %d: %s\n",test,ans);
    }
    return 0;
}
----------------------hdu3553 自己的代码-----------------------
#define maxn 100010

int rmq[maxn];
struct ST {
#define CMP  <          //大于为取大数，小于取小数
    int mm[maxn];
    int best[20][maxn];
    void init(int n) {
        int i,j,a,b;
        rmq[0] = -999999999;//让rmq[0]取最反向值
        mm[0]=-1;
        for(i=1; i<=n; i++) {
            mm[i]=((i&(i-1))==0)?mm[i-1]+1:mm[i-1];
            best[0][i]=i;
        }
        for(i=1; i<=mm[n]; i++) {
            for(j=1; j<=n+1-(1<<i); j++) {
                a=best[i-1][j];
                b=best[i-1][j+(1<<(i-1))];
                best[i][j]=rmq[a] CMP rmq[b]?a:b;
            }
        }
    }
    int query(int a, int b) {
        if(a > b)return 0;
        int t;
        t=mm[b-a+1];
        a=best[t][a];
        b=best[t][b-(1<<t)+1];
        return rmq[a] CMP rmq[b] ? a : b;
    }
} st;

//------------华丽的分隔线--------------------

int A[maxn], B[maxn], S[maxn], C[maxn];
int *rank, *height, *sa = S+1;

void sortAndRank(int *a1, int *a2, int n, int &m, int j) {
    int i;
    memset(C, 0, sizeof(C));
    for(i = 0; i < n; i ++)     C[a1[i]] ++;
    for(i = 1; i <=m; i ++)     C[i] += C[i-1];
    for(i = n-1; i >= 0; i --)      sa[--C[a1[a2[i]]]] = a2[i];
    a2[sa[0]] = m = 0;
    for(i = 1; i < n; i ++)
    a2[sa[i]] = a1[sa[i-1]]==a1[sa[i]] && a1[sa[i-1]+j]==a1[sa[i]+j] ? m : ++ m;
}

void da(char*str, int n, int m) {
    int *a1 = A, *a2 = B, *tmp;
    int i, j, p;
    for(i = 0; i < n; i ++) {
        a1[i] = i;
        a2[i] = str[i];
    }
    a1[n] = a2[n] = -1;
    sortAndRank(a2, a1, n, m, 0);
    for(j = 1; m < n-1; j <<= 1) {
        p = 0;
        for(i = n-j; i < n; i ++)   a2[p ++] = i;
        for(i= 0; i < n; i ++)      if(sa[i]>=j)  a2[p ++] = sa[i]-j;
        sortAndRank(a1, a2, n, m, j);
        tmp = a1;a1 = a2;  a2 = tmp;
    }
    rank = a1;    height = a2;
}

void calHeight(char *str, int n) {
    int i, j, k;
    sa[-1] = n;
    for(height[0] = k = i = 0; i < n; i ++) {
        for(k ? k-- : 0, j = sa[rank[i]-1]; str[i+k]==str[j+k]; k++);
        height[rank[i]] = k;
    }
}

char str[maxn];
int n;
long long sum[maxn];

long long k;
int ansL, ansR;

void compute() {
    int l = 0, r = n-1, m;
    int hgt, wid;

    long long upNum;

    while(l != r) {
        m = st.query(l+1,r);

        hgt = r - l + 1;
        wid = rmq[m];

        if (k < (long long) hgt * wid) {
            ansL = sa[l];
            ansR = ansL + k / hgt + 1;
            return;
        }
        upNum = sum[m-1] - sum[l-1];
```

```
          if(k-(long long) wid * (r - m + 1) < upNum) {
               k -=(long long) wid * (r - m + 1);
               r = m - 1;
          } else {
               k -= upNum;
               l = m;
          }
     }
     ansL = sa[l];
     ansR = sa[l] + k + 1;
}

int main() {
     int t;
     scanf("%d", &t);
     for(int idx=1; idx<=t; idx ++) {
          scanf("%s", str);
          n = strlen(str);
          cin >> k; k --;
          da(str, n, 127);
          calHeight(str, n);
          copy(height, height+n, rmq);
          st.init(n-1);

          sum[0] = n - sa[0];
          for(int i = 1; i < n; i ++) {
               sum[i] = sum[i-1] + n - sa[i];
          }

          compute();
          printf("Case %d: ", idx);
          for(int i = ansL; i < ansR; i ++) {
               putchar(str[i]);
          }
          putchar('\n');
     }
     return 0;
}


//----------------------poj 1743-----------------------------
#include <stdio.h>
#include <string.h>
#include <memory.h>
#define maxn 20010

int A[maxn], B[maxn], S[maxn], C[maxn];
int *rank, *height, *sa = S+1;

void sortAndRank(int *a1, int *a2, int n, int &m, int j) {
     int i;
     memset(C, 0, sizeof(C));
     for(i = 0; i < n; i ++)     C[a1[i]] ++;
     for(i = 1; i <=m; i ++)     C[i] += C[i-1];
     for(i = n-1; i >= 0; i --)  sa[--C[a1[a2[i]]]] = a2[i];
     for(i = 1; i < n; i ++)     a2[sa[i]] = a1[sa[i-1]]==a1[sa[i]] &&
a1[sa[i-1]+j]==a1[sa[i]+j] ? m : ++ m;
}

void da(int*str, int n, int m) {
     int *a1 = A, *a2 = B, *tmp;
     int i, j, p;
     for(i = 0; i < n; i ++) {
          a1[i] = i;
          a2[i] = str[i];
     }
     a1[n] = a2[n] = -1;
     sortAndRank(a2, a1, n, m, 0);
     for(j = 1; m < n-1; j <<= 1) {
          p = 0;
          for(i = n-j; i < n; i ++)   a2[p ++] = i;
          for(i= 0; i < n; i ++)      if(sa[i]>=j)  a2[p ++] = sa[i]-j;
          sortAndRank(a1, a2, n, m, j);
          tmp = a1; a1 = a2;  a2 = tmp;
     }
     rank = a1;    height = a2;
}

int idx;

void calHeight(int *str, int n) {
     int i, j, k;
     sa[-1] = n;
     for(height[0] = idx = k = i = 0; i < n; i ++) {
          for(k ? k-- : 0, j = sa[rank[i]-1]; str[i+k]==str[j+k]; k++);
          height[rank[i]] = k;
          if(k > height[idx] || (k==height[idx]&&rank[i]<idx))    idx = rank[i];
     }
}

int str[maxn];

bool check(int h, int n) {
     int max, min;
     for(int i = 0; i < n; i ++) {
          if(height[i] < h) {
               max = min = sa[i];
          } else {
               if(sa[i] > max)    max = sa[i];
               if(sa[i] < min)    min = sa[i];
               if(max - min > h) {
                    return true;
               }
          }
     }
     return false;
}

int ms() {
     static int num;
     static char c;
     while(c = getchar(), c>'9'||c<'0');
     for(num = c-'0'; c = getchar(), c>='0' && c<='9'; num = num*10+c-'0');
     return num;
}
```

```
int main() {
     int n, tmp1, tmp2;
     while(n = ms()) {
          tmp1 = ms();
          n --;
          for(int i = 0; i < n; i ++) {
               tmp2 = ms();
               str[i] = tmp2 - tmp1 + 100;
               tmp1 = tmp2;
          }
          da(str, n, 200);
          calHeight(str, n);
          int left = 0, right = (n+1)/2 + 1, mid;
          while(left+1 != right) {
               mid = (left + right)/2;
               if(check(mid, n))  left = mid;
               else               right = mid;
          }
          if(left < 4)  left = 0;
          else left ++;
          printf("%d\n", left);
     }
}
//--------------------------------poj 3261--------------------------------
#include <stdio.h>
#include <string.h>
#include <memory.h>
#define maxn 20010

int A[maxn], B[maxn], S[maxn], C[1000010];
int *rank, *height, *sa = S+1;

void sortAndRank(int *a1, int *a2, int n, int &m, int j) {
     int i;
     memset(C, 0, sizeof(C));
     for(i = 0; i < n; i ++)     C[a1[i]] ++;
     for(i = 1; i <=m; i ++)     C[i] += C[i-1];
     for(i = n-1; i >= 0; i --)  sa[--C[a1[a2[i]]]] = a2[i];
     a2[sa[0]] = m = 0;
     for(i = 1; i < n; i ++)     a2[sa[i]] = a1[sa[i-1]]==a1[sa[i]] &&
a1[sa[i-1]+j]==a1[sa[i]+j] ? m : ++ m;
}

void da(int*str, int n, int m) {
     int *a1 = A, *a2 = B, *tmp;
     int i, j, p;
     for(i = 0; i < n; i ++) {
          a1[i] = i;
          a2[i] = str[i];
     }
     a1[n] = a2[n] = -1;
     sortAndRank(a2, a1, n, m, 0);
     for(j = 1; m < n-1; j <<= 1) {
          p = 0;
          for(i = n-j; i < n; i ++)   a2[p ++] = i;
          for(i= 0; i < n; i ++)      if(sa[i]>=j)  a2[p ++] = sa[i]-j;
          sortAndRank(a1, a2, n, m, j);
          tmp = a1; a1 = a2;  a2 = tmp;
     }
     rank = a1;    height = a2;
}

int idx;

void calHeight(int *str, int n) {
     int i, j, k;
     sa[-1] = n;
     for(height[0] = idx = k = i = 0; i < n; i ++) {
          for(k ? k-- : 0, j = sa[rank[i]-1]; str[i+k]==str[j+k]; k++);
          height[rank[i]] = k;
          if(k > height[idx] || (k==height[idx]&&rank[i]<idx))    idx = rank[i];
     }
}

int str[maxn];

bool check(int h, int n, int k) {
     int start = 0;
     int i;
     for(i = 0; i < n; i ++) {
          if(height[i] < h) {
               if(i - start >= k) return true;
               start = i;
          }
     }
     if(i - start >= k) return true;
     return false;
}

int ms() {
     static int num;
     static char c;
     while(c = getchar(), c>'9'||c<'0');
     for(num = c-'0'; c = getchar(), c>='0' && c<='9'; num = num*10+c-'0');
     return num;
}

int main() {
     int n, k;
     while(scanf("%d%d", &n, &k) != EOF) {
          for(int i = 0; i < n; i ++) {
               str[i] = ms();
          }
          da(str, n, 1000001);
          calHeight(str, n);
          int left = 0, right = n, mid;
          while(left + 1 != right) {
               mid = (left + right) / 2;
               if(check(mid, n, k))    left = mid;
               else                    right=mid;
          }
          printf("%d\n", left);
     }
}
//---------------------spoj 694/705----------------------------
#include <stdio.h>
#include <string.h>
#include <memory.h>
#define maxn 50010
```

```
int A[maxn], B[maxn], S[maxn], C[maxn];
int *rank, *height, *sa = S+1;

void sortAndRank(int *a1, int *a2, int n, int &m, int j) {
     int i;
     memset(C, 0, sizeof(C));
     for(i = 0; i < n; i ++)     C[a1[i]] ++;
     for(i = 1; i <=m; i ++)     C[i] += C[i-1];
     for(i = n-1; i >= 0; i --)  sa[--C[a1[a2[i]]]] = a2[i];
     a2[sa[0]] = m = 0;
     for(i = 1; i < n; i ++)     a2[sa[i]] = a1[sa[i-1]]==a1[sa[i]] &&
a1[sa[i-1]+j]==a1[sa[i]+j] ? m : ++ m;
}

void da(unsigned char*str, int n, int m) {
     int *a1 = A, *a2 = B, *tmp;
     int i, j, p;
     for(i = 0; i < n; i ++) {
          a1[i] = i;
          a2[i] = str[i];
     }
     a1[n] = a2[n] = -1;
     sortAndRank(a2, a1, n, m, 0);
     for(j = 1; m < n-1; j <<= 1) {
          p = 0;
          for(i = n-j; i < n; i ++)   a2[p ++] = i;
          for(i= 0; i < n; i ++)      if(sa[i]>=j)  a2[p ++] = sa[i]-j;
          sortAndRank(a1, a2, n, m, j);
          tmp = a1; a1 = a2;  a2 = tmp;
     }
     rank = a1;    height = a2;
}

int idx;

void calHeight(char *str, int n) {
     int i, j, k;
     sa[-1] = n;
     for(height[0] = idx = k = i = 0; i < n; i ++) {
          for(k ? k-- : 0, j = sa[rank[i]-1]; str[i+k]==str[j+k]; k++);
          height[rank[i]] = k;
          if(k > height[idx] || (k==height[idx]&&rank[i]<idx))    idx = rank[i];
     }
}

char str[maxn];

int main() {
     int t, n;
     for(scanf("%d", &t); t--; ) {
          scanf("%s", str);
          n = strlen(str);
          da((unsigned char*)str, n, 256);
          calHeight(str, n);
          /*str[height[idx] + sa[idx]] = 0;
          printf("%s\n", str+sa[idx]);
          */
          int cnt = 0;
          for(int i = 0; i < n; i ++) {
               cnt += n - sa[i] - height[i];
          }
          printf("%d\n", cnt);
     }
}
//-----------------------ural 1297---------------------------
#include <stdio.h>
#include <string.h>
#include <memory.h>
#define maxn 2020

int A[maxn], B[maxn], S[maxn], C[maxn];
int *rank, *height, *sa = S+1;

void sortAndRank(int *a1, int *a2, int n, int &m, int j) {
     int i;
     memset(C, 0, sizeof(C));
     for(i = 0; i < n; i ++)     C[a1[i]] ++;
     for(i = 1; i <=m; i ++)     C[i] += C[i-1];
     for(i = n-1; i >= 0; i --)  sa[--C[a1[a2[i]]]] = a2[i];
     a2[sa[0]] = m = 0;
     for(i = 1; i < n; i ++)     a2[sa[i]] = a1[sa[i-1]]==a1[sa[i]] &&
a1[sa[i-1]+j]==a1[sa[i]+j] ? m : ++ m;
}

void da(char*str, int n, int m) {
     int *a1 = A, *a2 = B, *tmp;
     int i, j, p;
     for(i = 0; i < n; i ++) {
          a1[i] = i;
          a2[i] = str[i];
     }
     a1[n] = a2[n] = -1;
     sortAndRank(a2, a1, n, m, 0);
     for(j = 1; m < n-1; j <<= 1) {
          p = 0;
          for(i = n-j; i < n; i ++)   a2[p ++] = i;
          for(i= 0; i < n; i ++)      if(sa[i]>=j)  a2[p ++] = sa[i]-j;
          sortAndRank(a1, a2, n, m, j);
          tmp = a1; a1 = a2;  a2 = tmp;
     }
     rank = a1;    height = a2;
}

int idx;

void calHeight(char *str, int n) {
     int i, j, k;
     sa[-1] = n;
     for(height[0] = idx = k = i = 0; i < n; i ++) {
          for(k ? k-- : 0, j = sa[rank[i]-1]; str[i+k]==str[j+k]; k++);
          height[rank[i]] = k;
          if(k > height[idx] || (k==height[idx]&&rank[i]<idx))    idx = rank[i];
     }
}

#include <math.h>
struct ST {
     int n;
     int d[13][2020];
```

```
    int max(int a, int b) {
        return a<b?a:b;
    }
    void init(int*a, int n) {
        this->n = n;
        for(int i = 1; i <= n; i ++)     d[0][i] = a[i];
        int b = (int)(log((double)n)/log(2.0));
        for(int j = 1; j <= b; j ++) {
            for(int i = n-(1<<j)+1; i >=1 ; i --) {
                d[j][i] = max(d[j-1][i], d[j-1][i+(1<<(j-1))]);
            }
        }
    }
    int query(int a, int b) {   //
        int depth = (int)(log((double)(b-a+1))/log(2.0));
        return max(d[depth][a], d[depth][b+1-(1<<depth)]);
    }
};

char str[maxn];
ST st;

int query(int a, int b) {
    a = rank[a];  b = rank[b];
    int rank1 = a < b ? a : b;
    int rank2 = a > b ? a : b;
    return st.query(rank1+1, rank2);
}

int n;
void compute(int & off, int & len) {
    int n = strlen(str);
    str[2*n+1] = 0;
    str[n] = 1;
    for(int i = 0; i < n; i ++) {
        str[2*n - i] = str[i];
    }
    da(str, 2*n+1, 127);
    calHeight(str, 2*n+1);
    st.init(height, 2*n+1);

    int max = 1;
    int idx = 0, tmp;


    for(int i = 1; i < n; i ++) {
        tmp = query(i, 2*n-i);
        if(2*tmp - 1 > max) {
            max = 2*tmp - 1;
            idx = i;
        }
        tmp = query(i, 2*n-i+1);
        if(2*tmp > max) {
            max = 2*tmp;
            idx = i;
        }
    }
    idx -= max / 2;

    off = idx;
    len = max;
}

int main() {
    while(scanf("%s", str) != EOF) {

        int off, len;
        compute(off, len);

        str[off+len] = 0;
        printf("%s\n", str+off);
    }
    return 0;
}
//----------------------poj 2406----------------------------
#include <stdio.h>
#include <string.h>
int n, i, k, pi[1000010];
#define KMP_GO(X) while(k>0 && P[k] != X[i])   k = pi[k-1];  if(P[k] == X[i])   k
++
void kmp_match(char*P) {
    n = strlen(P);
    pi[0] = k = 0;
    for(i = 1; i < n; i ++) {
        KMP_GO(P);
        pi[i] = k;
    }
}

char str[1000010];

int main() {
    while(scanf("%s", str), !(str[0]=='.'&&str[1]==0)) {
        kmp_match(str);
        int k = n - pi[n-1];
        if(k && n % k == 0) {
            printf("%d\n", n/k);
        } else printf("%d\n", 1);
    }
}
//----------------------spoj 687----------------------------
#define maxn 100010

int rmq[maxn];
struct ST {
#define CMP  <          //大于为取大数，小于取小数
    int mm[maxn];
    int best[20][maxn];
    void init(int n) {
        int i,j,a,b;
        rmq[0] = -999999999;//让rmq[0]取最反向值
        mm[0]=-1;
        for(i=1; i<=n; i++) {
            mm[i]=((i&(i-1))==0)?mm[i-1]+1:mm[i-1];
            best[0][i]=i;
        }
        for(i=1; i<=mm[n]; i++) {
            for(j=1; j<=n+1-(1<<i); j++) {
```

```
                    a=best[i-1][j];
                    b=best[i-1][j+(1<<(i-1))];
                    best[i][j]=rmq[a] CMP rmq[b]?a:b;
               }
          }
     }
     int query(int a, int b) {
          if(a > b)return 0;
          int t;
          t=mm[b-a+1];
          a=best[t][a];
          b=best[t][b-(1<<t)+1];
          return rmq[a] CMP rmq[b] ? a : b;
     }
} st;

//--------------------华丽的分隔线-------------------------------

int A[maxn], B[maxn], S[maxn], C[maxn];
int *rank, *height, *sa = S+1;

void sortAndRank(int *a1, int *a2, int n, int &m, int j) {
     int i;
     memset(C, 0, sizeof(C));
     for(i = 0; i < n; i ++)     C[a1[i]] ++;
     for(i = 1; i <=m; i ++)     C[i] += C[i-1];
     for(i = n-1; i >= 0; i --)  sa[--C[a1[a2[i]]]] = a2[i];
     a2[sa[0]] = m = 0;
     for(i = 1; i < n; i ++)     a2[sa[i]] = a1[sa[i-1]]==a1[sa[i]] &&
a1[sa[i-1]+j]==a1[sa[i]+j] ? m : ++ m;
}
void da(char*str, int n, int m) {
     int *a1 = A, *a2 = B, *tmp;
     int i, j, p;
     for(i = 0; i < n; i ++) {
          a1[i] = i;
          a2[i] = str[i];
     }
     a1[n] = a2[n] = -1;
     sortAndRank(a2, a1, n, m, 0);
     for(j = 1; m < n-1; j <<= 1) {
          p = 0;
          for(i = n-j; i < n; i ++)   a2[p ++] = i;
          for(i= 0; i < n; i ++)      if(sa[i]>=j)  a2[p ++] = sa[i]-j;
          sortAndRank(a1, a2, n, m, j);
          tmp = a1; a1 = a2;  a2 = tmp;
     }
     rank = a1;    height = a2;
}
void calHeight(char *str, int n) {
     int i, j, k;
     sa[-1] = n;
     for(height[0] = k = i = 0; i < n; i ++) {
          for(k ? k-- : 0, j = sa[rank[i]-1]; str[i+k]==str[j+k]; k++);
          height[rank[i]] = k;
     }
}

int lcp(int a,int b) {  //询问后缀ab的最长公共前缀, assume a!=b
     a=rank[a];b=rank[b];
     if(a>b)   swap(a,b);
     return(height[st.query(a+1,b)]);
}

char str[maxn];

int main() {
     int ans, k, now, n, l, i, start;
     int index;
     int t;
     scanf("%d", &t);
     for(index = 1; t --; index ++) {
          scanf("%d",&n);
          for(i=0;i<n;i++)   scanf(" %c", str+i);
          da(str, n, 127);
          calHeight(str, n);
          copy(height, height+n, rmq);

          st.init(n-1);
          //init over!

          ans = 0;
          for(l = 1; l <= n/2; l ++) {
               for(i = 0; i + l < n; i += l) {
                    k = lcp(i, i+l);
                    now = k / l;
                    start = i - (l-k%l);
                    if(start >= 0 && lcp(start,start+l)>=l)   now ++;
                    ans = max(ans, now);
               }
          }
          ans ++;
          printf("%d\n", ans);
     }
}
//----------------------poj 3693(others)-------------------------
#include <iostream>
#include <cstdio>
#include <cstring>
using namespace std;

#define maxn 6100000
#define F(x) ((x)/3+((x)%3==1?0:tb))
#define G(x) ((x)<tb?(x)*3+1:((x)-tb)*3+2)
int wa[maxn], wb[maxn], wv[maxn], wss[maxn];
int c0(int *r, int a, int b) {
     return r[a] == r[b] && r[a + 1] == r[b + 1] && r[a + 2] == r[b + 2];
}
int c12(int k, int *r, int a, int b) {
     if (k == 2)
          return r[a] < r[b] || r[a] == r[b] && c12(1, r, a + 1, b + 1);
     else
          return r[a] < r[b] || r[a] == r[b] && wv[a + 1] < wv[b + 1];
```

```
}
void sort(int *r, int *a, int *b, int n, int m) {
     int i;
     for (i = 0; i < n; i++)
          wv[i] = r[a[i]];
     for (i = 0; i < m; i++)
          wss[i] = 0;
     for (i = 0; i < n; i++)
          wss[wv[i]]++;
     for (i = 1; i < m; i++)
          wss[i] += wss[i - 1];
     for (i = n - 1; i >= 0; i--)
          b[--wss[wv[i]]] = a[i];
     return;
}
void dc3(int *r, int *sa, int n, int m) {
     int i, j, *rn = r + n, *san = sa + n, ta = 0, tb = (n + 1) / 3, tbc = 0, p;
     r[n] = r[n + 1] = 0;
     for (i = 0; i < n; i++)
          if (i % 3 != 0)
               wa[tbc++] = i;
     sort(r + 2, wa, wb, tbc, m);
     sort(r + 1, wb, wa, tbc, m);
     sort(r, wa, wb, tbc, m);
     for (p = 1, rn[F(wb[0])] = 0, i = 1; i < tbc; i++)
          rn[F(wb[i])] = c0(r, wb[i - 1], wb[i]) ? p - 1 : p++;
     if (p < tbc)
          dc3(rn, san, tbc, p);
     else
          for (i = 0; i < tbc; i++)
               san[rn[i]] = i;
     for (i = 0; i < tbc; i++)
          if (san[i] < tb)
               wb[ta++] = san[i] * 3;
     if (n % 3 == 1)
          wb[ta++] = n - 1;
     sort(r, wb, wa, ta, m);
     for (i = 0; i < tbc; i++)
          wv[wb[i] = G(san[i])] = i;
     for (i = 0, j = 0, p = 0; i < ta && j < tbc; p++)
          sa[p] = c12(wb[j] % 3, r, wa[i], wb[j]) ? wa[i++] : wb[j++];
     for (; i < ta; p++)
          sa[p] = wa[i++];
     for (; j < tbc; p++)
          sa[p] = wb[j++];
     return;
}
int rank[maxn], height[maxn];
void calheight(int *r, int *sa, int n) {
     int i, j, k = 0;
     for (i = 1; i <= n; i++)
          rank[sa[i]] = i;
     for (i = 0; i < n; height[rank[i++]] = k)
          for (k ? k-- : 0, j = sa[rank[i] - 1]; r[i + k] == r[j + k]; k++)
               ;
     return;
}
int mm[maxn];
int best[20][maxn];
void initRMQ(int n, int* RMQ)//这个RMQ是从1..n的 但是height碰巧也是从1开始
{
     int i, j, a, b;
     for (mm[0] = -1, i = 1; i <= n; i++)
          mm[i] = ((i & (i - 1)) == 0) ? mm[i - 1] + 1 : mm[i - 1];
     for (i = 1; i <= n; i++)
          best[0][i] = i;
     for (i = 1; i <= mm[n]; i++)
          for (j = 1; j <= n + 1 - (1 << i); j++) {
               a = best[i - 1][j];
               b = best[i - 1][j + (1 << (i - 1))];
               if (RMQ[a] < RMQ[b])
                    best[i][j] = a;//最小RMQ
               else
                    best[i][j] = b;
          }
     return;
}
int askRMQ(int a, int b, int* RMQ) {
     int t;
     t = mm[b - a + 1];
     b -= (1 << t) - 1;
     a = best[t][a];
     b = best[t][b];
     return RMQ[a] < RMQ[b] ? a : b;//最小RMQ
}
int lcp(int a, int b) {
     int t;
     a = rank[a];
     b = rank[b];
     if (a > b) {
          t = a;
          a = b;
          b = t;
     }
     return height[askRMQ(a + 1, b, height)];
}
/////////用法
//    dc3(r,sa,n+1,128);//转int后的字符串 r[0]=0 sa数组存放 长度+1 字符集大小
//    calheight(r,sa,n);//r同上 sa为上面返回的 n就是长度
//    所有数组都要开到3倍以上 如果是拼接串 要开到总长3倍
int r[maxn], sa[maxn];
int n;
char s[maxn] = "bacbacbaba";
int maxrep, beg, len;
void work(int k) {
     int i, j, l, r;
     for (i = 0; i + k < n; i += k)
          if (s[i] == s[i + k]) {
               j = i + k;
               r = lcp(i, j);
               l = lcp(n * 2 - i, n * 2 - j);
               int lt, rt;
               int rep;
               int thl;
               lt = i - l + 1;
               rt = i + r + k;
               rep = (l + r - 1) / k + 1;
               thl = rep * k;

               if (l + r - 1 < k)
```

```
                continue;
            if (rep < maxrep)
                continue;

            for (j = lt; j + thl <= rt; j++)
                if (rep > maxrep || (rep == maxrep && rank[j] < rank[beg])
                        || (rep == maxrep && rank[j] == rank[beg] && thl < len))
{
                    maxrep = rep;
                    beg = j;
                    len = thl;
                }
        }
}

int main() {
    int i, j;
    int cas = 0;
    while (1) {
        gets(s);
        if (s[0] == '#')
            break;
        cas++;
        printf("Case %d: ", cas);
        n = strlen(s);
        for (i = 0; i < n; i++)
            r[i] = s[i] + 1;
        r[n] = 1;
        for (i = 0; i < n; i++)
            r[i + n + 1] = s[n - i - 1] + 1;
        r[n * 2 + 1] = 0;
        dc3(r, sa, n * 2 + 2, 300);
        calheight(r, sa, n * 2 + 1);
        initRMQ(n * 2 + 1, height);
        j = 0;
        for (i = 1; i < n; i++)
            if (s[i] < s[j])
                j = i;
        maxrep = 1;
        beg = j;
        len = 1;
        for (i = 1; i < n; i++)
            work(i);
        for (i = beg; i < beg + len; i++)
            printf("%c", s[i]);
        printf("\n");
    }
    return 0;
}
//-----------------------poj 2774--------------------------
#include <stdio.h>
#include <string.h>
#include <memory.h>
#include <math.h>
#define maxn 200010

int A[maxn], B[maxn], S[maxn], C[maxn];
int *rank, *height, *sa = S+1;

void sortAndRank(int *a1, int *a2, int n, int &m, int j) {
    int i;
    memset(C, 0, sizeof(C));
    for(i = 0; i < n; i ++)      C[a1[i]] ++;
    for(i = 1; i <=m; i ++)      C[i] += C[i-1];
    for(i = n-1; i >= 0; i --)  sa[--C[a1[a2[i]]]] = a2[i];
    a2[sa[0]] = m = 0;
    for(i = 1; i < n; i ++)      a2[sa[i]] = a1[sa[i-1]]==a1[sa[i]] &&
a1[sa[i-1]+j]==a1[sa[i]+j] ? m : ++ m;
}

void da(char*str, int n, int m) {
    int *a1 = A, *a2 = B, *tmp;
    int i, j, p;
    for(i = 0; i < n; i ++) {
        a1[i] = i;
        a2[i] = str[i];
    }
    a1[n] = a2[n] = -1;
    sortAndRank(a2, a1, n, m, 0);
    for(j = 1; m < n-1; j <<= 1) {
        p = 0;
        for(i = n-j; i < n; i ++)   a2[p ++] = i;
        for(i= 0; i < n; i ++)      if(sa[i]>=j)  a2[p ++] = sa[i]-j;
        sortAndRank(a1, a2, n, m, j);
        tmp = a1; a1 = a2;  a2 = tmp;
    }
    rank = a1;    height = a2;
}
char str[maxn];
int n1, n2, n;
int max;
#define isDiff(i, j)    (i < j ? i < n1 && j > n1 : j < n1 && i > n1)

void calHeight(char *str, int n) {
    int i, j, k;
    sa[-1] = n;
    for(height[0] = k = i = 0; i < n; i ++) {
        for(k ? k-- : 0, j = sa[rank[i]-1]; str[i+k]==str[j+k]; k++);
        height[rank[i]] = k;
    }
}

int main() {
    while(scanf("%s", str) != EOF) {
        n1 = strlen(str);
        scanf("%s", str+n1+1);
        n2 = strlen(str+n1+1);
        n = n1 + n2 + 1;
        da(str, n, 127);
        calHeight(str, n);
        max = 0;
        for(int i = 1; i < n; i ++) {
            if(height[i] > max && isDiff(sa[i-1], sa[i])) {
                max = height[i];
            }
        }
        printf("%d\n", max);
    }

    return 0;
```

```
}
//----------------------ural 1517----------------------
#include <stdio.h>
#include <string.h>
#include <memory.h>
#include <math.h>
#define maxn 200010

int A[maxn], B[maxn], S[maxn], C[maxn];
int *rank, *height, *sa = S+1;

void sortAndRank(int *a1, int *a2, int n, int &m, int j) {
     int i;
     memset(C, 0, sizeof(C));
     for(i = 0; i < n; i ++)     C[a1[i]] ++;
     for(i = 1; i <=m; i ++)     C[i] += C[i-1];
     for(i = n-1; i >= 0; i --)  sa[--C[a1[a2[i]]]] = a2[i];
     a2[sa[0]] = m = 0;
     for(i = 1; i < n; i ++)     a2[sa[i]] = a1[sa[i-1]]==a1[sa[i]] &&
a1[sa[i-1]+j]==a1[sa[i]+j] ? m : ++ m;
}

void da(char*str, int n, int m) {
     int *a1 = A, *a2 = B, *tmp;
     int i, j, p;
     for(i = 0; i < n; i ++) {
          a1[i] = i;
          a2[i] = str[i];
     }
     a1[n] = a2[n] = -1;
     sortAndRank(a2, a1, n, m, 0);
     for(j = 1; m < n-1; j <<= 1) {
          p = 0;
          for(i = n-j; i < n; i ++)   a2[p ++] = i;
          for(i= 0; i < n; i ++)      if(sa[i]>=j)  a2[p ++] = sa[i]-j;
          sortAndRank(a1, a2, n, m, j);
          tmp = a1; a1 = a2;  a2 = tmp;
     }
     rank = a1;    height = a2;
}
char str[maxn];
int n1, n;
int max;
#define isDiff(i, j)    (i < j ? i < n1 && j > n1 : j < n1 && i > n1)

void calHeight(char *str, int n) {
     int i, j, k;
     sa[-1] = n;
     for(height[0] = k = i = 0; i < n; i ++) {
          for(k ? k-- : 0, j = sa[rank[i]-1]; str[i+k]==str[j+k]; k++);
          height[rank[i]] = k;
     }
}

int main() {

     while(scanf("%d", &n1) != EOF) {
          scanf("%s", str);
          scanf("%s", str+n1+1);
          n = n1 + n1 + 1;
          da(str, n, 127);
          calHeight(str, n);
          max = 0;
          int idx;
          for(int i = 1; i < n; i ++) {
               if(height[i] > max && isDiff(sa[i-1], sa[i])) {
                    max = height[i];
                    idx = sa[i];
               }
          }
          str[idx + max] = 0;
          printf("%s\n", str+idx);
     }

     return 0;
}
//----------------------poj 3415----------------------
#include <stdio.h>
#include <string.h>
#include <memory.h>
#include <math.h>
#define maxn 200010

int A[maxn], B[maxn], S[maxn], C[maxn];
int *rank, *height, *sa = S+1;

void sortAndRank(int *a1, int *a2, int n, int &m, int j) {
     int i;
     memset(C, 0, sizeof(C));
     for(i = 0; i < n; i ++)     C[a1[i]] ++;
     for(i = 1; i <=m; i ++)     C[i] += C[i-1];
     for(i = n-1; i >= 0; i --)  sa[--C[a1[a2[i]]]] = a2[i];
     a2[sa[0]] = m = 0;
     for(i = 1; i < n; i ++)     a2[sa[i]] = a1[sa[i-1]]==a1[sa[i]] &&
a1[sa[i-1]+j]==a1[sa[i]+j] ? m : ++ m;
}

void da(char*str, int n, int m) {
     int *a1 = A, *a2 = B, *tmp;
     int i, j, p;
     for(i = 0; i < n; i ++) {
          a1[i] = i;
          a2[i] = str[i];
     }
     a1[n] = a2[n] = -1;
     sortAndRank(a2, a1, n, m, 0);
     for(j = 1; m < n-1; j <<= 1) {
          p = 0;
          for(i = n-j; i < n; i ++)   a2[p ++] = i;
          for(i= 0; i < n; i ++)      if(sa[i]>=j)  a2[p ++] = sa[i]-j;
          sortAndRank(a1, a2, n, m, j);
          tmp = a1; a1 = a2;  a2 = tmp;
     }
     rank = a1;    height = a2;
}


void calHeight(char *str, int n) {
     int i, j, k;
     sa[-1] = n;
```

```
     for(height[0] = k = i = 0; i < n; i ++) {
          for(k ? k-- : 0, j = sa[rank[i]-1]; str[i+k]==str[j+k]; k++);
          height[rank[i]] = k;
     }
}

char str[maxn];
int stackA[maxn];
int stackB[maxn];
int lenA;
int lenB;
int start;

int main() {
     int k, n1, n2, n;
     while(scanf("%d", &k), k) {
          scanf("%s", str);
          n1 = strlen(str);
          scanf("%s", str+n1+1);
          n2 = strlen(str+n1+1);

str[n1] = 1;
//printf("str = %s\n", str);


          n = n1 + n2 + 1;
          da(str, n, 127);
          calHeight(str, n);

          long long ans = 0;
          long long sumA, sumB;
          for(int i = 0; i < n; i ++) {

               if(height[i] < k) {
                    sumA = sumB = start = lenA = lenB = 0;
                    //stackA[lenA ++] = height
               } else {
                    for(int j = lenA-1; j >= 0 && stackA[j] > height[i]; j --) {
                         sumA -= stackA[j] - height[i];
                         stackA[j] = height[i];
                    }
                    for(int j = lenB-1; j >= 0 && stackB[j] > height[i]; j --) {
                         sumB -= stackB[j] - height[i];
                         stackB[j] = height[i];
                    }

                    if(sa[i-1] < n1) { //A中的
                         stackA[lenA ++] = height[i];
                         sumA += height[i];
                    } else {
                         stackB[lenB ++] = height[i];
                         sumB += height[i];
                    }
                    if(sa[i] < n1) {
                         ans += sumB - (k-1)*lenB;
                    } else {
                         ans += sumA - (k-1)*lenA;
                    }
               }
//printf("i = %.2d, height = %d\tans = %.2I64d type = %c\tsa = %d\ts = %s\n", i,
height[i], ans, sa[i] < sa[0] ? 'A' : 'B', sa[i], str + sa[i]);
          }
          printf("%I64d\n", ans);
     }
}
//------------------------------poj 3294--------------------
#include <stdio.h>
#include <string.h>
#include <memory.h>
#include <math.h>
#define maxn 110000

int A[maxn], B[maxn], S[maxn], C[maxn];
int *rank, *height, *sa = S+1;

void sortAndRank(int *a1, int *a2, int n, int &m, int j) {
     int i;
     memset(C, 0, sizeof(C));
     for(i = 0; i < n; i ++)     C[a1[i]] ++;
     for(i = 1; i <=m; i ++)     C[i] += C[i-1];
     for(i = n-1; i >= 0; i --)  sa[--C[a1[a2[i]]]] = a2[i];
     a2[sa[0]] = m = 0;
     for(i = 1; i < n; i ++)     a2[sa[i]] = a1[sa[i-1]]==a1[sa[i]] &&
a1[sa[i-1]+j]==a1[sa[i]+j] ? m : ++ m;
}

void da(char*str, int n, int m) {
     int *a1 = A, *a2 = B, *tmp;
     int i, j, p;
     for(i = 0; i < n; i ++) {
          a1[i] = i;
          a2[i] = str[i];
     }
     a1[n] = a2[n] = -1;
     sortAndRank(a2, a1, n, m, 0);
     for(j = 1; m < n-1; j <<= 1) {
          p = 0;
          for(i = n-j; i < n; i ++)   a2[p ++] = i;
          for(i= 0; i < n; i ++)      if(sa[i]>=j)  a2[p ++] = sa[i]-j;
          sortAndRank(a1, a2, n, m, j);
          tmp = a1; a1 = a2;  a2 = tmp;
     }
     rank = a1;    height = a2;
}


void calHeight(char *str, int n) {
     int i, j, k;
     sa[-1] = n;
     for(height[0] = k = i = 0; i < n; i ++) {
          for(k ? k-- : 0, j = sa[rank[i]-1]; str[i+k]==str[j+k]; k++);
          height[rank[i]] = k;
     }
}

int n;
char str[110000];
//int start[110];
int st, len;
int idx[110000];
```

```
int L;
bool has[110];
int min;

bool judge(int l) {
     int i, id;
     int cnt;
     for(i = 0; i < len; i ++) {
//printf("height[%d] = %d\n", i, height[i]);
          id = idx[sa[i]];
          if(height[i] < l || id == -1 || (id != idx[sa[i]+l-1])) {    //初始化
               memset(has, 0, sizeof(has));
               cnt = 0;
//printf("-----clear-----\n");
          }

          if(id == -1|| (id != idx[sa[i]+l-1]))continue;
          if(!has[id]) {
//printf("i = %d, id = %d\n", i, id);
               has[id] = true;
               cnt ++;
               if(cnt > n/2) {
//printf("cnt = %d\n", cnt);
//printf("ok!!!\n");
                    return true;
               }
          }
     }
     return false;
}

bool judge2(int l) {
     int i, id;
     int cnt;
     char c;
     int stt;
     for(i = 0; i < len; i ++) {
          id = idx[sa[i]];
          if(height[i] < l || id == -1 || (id != idx[sa[i]+l-1])) {    //初始化
               memset(has, 0, sizeof(has));
               cnt = 0;
          }

          if(id == -1|| (id != idx[sa[i]+l-1]))continue;
          if(!has[id]) {
               has[id] = true;
               cnt ++;
               if(cnt > n/2) {
//printf("cnt = %d\n", cnt);
                    stt = sa[i];
                    c = str[stt + l];
                    str[stt + l] = 0;
                    printf("%s\n", str+stt);
                    str[stt + l] = c;


                    cnt = 1<<sizeof(int)*8-1;
               }
          }
     }
}

void bsearch() {
     int left = 0;      //left永远ok
     int right = min+1; //right永远不ok
     int mid;
     while(left+1 != right) {

          mid = (left + right) / 2;
//printf("left = %d, right = %d, mid = %d\n", left, right, mid);
          if(judge(mid)) {
               left = mid;
          } else {
               right = mid;
          }
     }
     L = left;
}

int main() {

     while(scanf("%d", &n), n) {
          st = 0;
          min = -1;
          memset(idx, 255, sizeof(idx));
          for(int i = 0; i < n; i ++) {
               //start[i] = st;
               scanf("%s", str + st);
               len = strlen(str+st);
               if(len > min)  min = len;
               for(int j = st; j < st+len; j ++) {
                    idx[j] = i;
               }
               idx[st+len] = -1;
               str[st+len] = 1;
               st = st + len + 1;


               //for()
          }
          /*for(int i = 0; i < st; i ++) {
               printf("%d ", idx[i]);
          }
          printf("\n");
          */
          len = st-1;
          str[len] = 0;
          //printf("st = %d\n", st);
          //printf("len = %d %d\n", strlen(str), len);
          //printf("str = [%s]\n", str);
          /*for(int i = 0; i < n; i ++) {
               printf("start[%d] = %d\n", i, start[i]);
          }*/

          da(str, len, 127);
          calHeight(str, len);
          /*
          for(int i = 0; i < len; i ++) {
               printf("height[%d] = %d, sa = %d, s = %s\n", i, height[i], sa[i], str
```

```
+ sa[i]);
          }*/

          bsearch();
//printf("L = %d\n", L);
          if(L == 0)    printf("?\n");
          else {
               judge2(L);
          }
          printf("\n");
     }
     return 0;
}
//-----------------------------spoj 220-----------------------------
#include <stdio.h>
#include <string.h>
#include <memory.h>
#include <math.h>
#define maxn 110000

int A[maxn], B[maxn], S[maxn], C[maxn];
int *rank, *height, *sa = S+1;

void sortAndRank(int *a1, int *a2, int n, int &m, int j) {
     int i;
     memset(C, 0, sizeof(C));
     for(i = 0; i < n; i ++)     C[a1[i]] ++;
     for(i = 1; i <=m; i ++)     C[i] += C[i-1];
     for(i = n-1; i >= 0; i --)  sa[--C[a1[a2[i]]]] = a2[i];
     a2[sa[0]] = m = 0;
     for(i = 1; i < n; i ++)     a2[sa[i]] = a1[sa[i-1]]==a1[sa[i]] &&
a1[sa[i-1]+j]==a1[sa[i]+j] ? m : ++ m;
}

void da(char*str, int n, int m) {
     int *a1 = A, *a2 = B, *tmp;
     int i, j, p;
     for(i = 0; i < n; i ++) {
          a1[i] = i;
          a2[i] = str[i];
     }
     a1[n] = a2[n] = -1;
     sortAndRank(a2, a1, n, m, 0);
     for(j = 1; m < n-1; j <<= 1) {
          p = 0;
          for(i = n-j; i < n; i ++)   a2[p ++] = i;
          for(i= 0; i < n; i ++)      if(sa[i]>=j)  a2[p ++] = sa[i]-j;
          sortAndRank(a1, a2, n, m, j);
          tmp = a1; a1 = a2;  a2 = tmp;
     }
     rank = a1;    height = a2;
}


void calHeight(char *str, int n) {
     int i, j, k;
     sa[-1] = n;
     for(height[0] = k = i = 0; i < n; i ++) {
          for(k ? k-- : 0, j = sa[rank[i]-1]; str[i+k]==str[j+k]; k++);
          height[rank[i]] = k;
     }
}

     }
}

int n;
char str[maxn];
int idx[maxn];
int len;
int ans;
int start;
int minLen;

int max[11], min[11];
bool ok[11];
int MAX = ~(1<<sizeof(int)*8-1);
int MIN = 1<<sizeof(int)*8-1;


bool check(int l) {
     int id;
     int cnt;
//printf("len = %d\n", len);
     for(int i = 0; i < len; i ++) {
          if(height[i] < l) {
               for(int i = 0; i < n; i ++) {
                    max[i] = MIN;
                    min[i] = MAX;
                    ok[i] = 0;
                    cnt = 0;
               }
          }
          id = idx[sa[i]];
//printf("i = %d, sa = %d, l = %d, id = %d, id2 = %d\n", i, sa[i], l, idx[sa[i]],
idx[sa[i]+l-1]);
          if(id != idx[sa[i] + l - 1])     continue;
//printf("cnt = %d\n", cnt);
          if(!ok[id]) {
               if(sa[i] < min[id])     min[id] = sa[i];
               if(sa[i] > max[id])     max[id] = sa[i];
               if(max[id] - min[id] >= l) {
                    ok[id] = true;
                    cnt ++;

                    if(cnt == n)  return true;
               }
          }
     }
     return false;
}

void bsearch() {
     int left = 0, right = minLen/2 + 1;   //left永远ok, right永远不ok
     int mid;
     while(left +1 != right) {
          mid = (left + right) / 2;
//printf("left = %d, right = %d, mid = %d\n", left, right, mid);
          if(check(mid)) {
               left = mid;
          } else    right = mid;
     }
     ans = left;
```

```
}

int main() {
     int t;
     for(scanf("%d", &t); t--; ) {

          minLen = 888888888;
          scanf("%d", &n);
          memset(idx, 255, sizeof(idx));
          start = 0;
          for(int i = 0; i < n; i ++) {
               scanf("%s", str+start);
               len = strlen(str + start);
               if(len < minLen)   minLen = len;
               for(int j = start; j < start + len; j ++) {
                    idx[j] = i;
               }
               str[start + len] = 1;
               start = start + len + 1;
          }
          len = start - 1;
          str[len] = 0;
          /*
          printf("str = [%s]\n", str);
          for(int i = 0; i <= len; i ++)   printf("%d ", idx[i]);
          printf("\n");
          */
          da(str, len, 127);
          calHeight(str, len);
          bsearch();
          printf("%d\n", ans);
     }
     return 0;
}
//-------------------poj 1226-------------------------
//You are given a number of case-sensitive strings of alphabetic characters, find
the largest string X, such that either X, or its inverse can be found as a substring
of any of the given strings.
#include <stdio.h>
#include <string.h>
#include <memory.h>
#include <math.h>
#define maxn 21000

int A[maxn], B[maxn], S[maxn], C[maxn];
int *rank, *height, *sa = S+1;

void sortAndRank(int *a1, int *a2, int n, int &m, int j) {
     int i;
     memset(C, 0, sizeof(C));
     for(i = 0; i < n; i ++)     C[a1[i]] ++;
     for(i = 1; i <=m; i ++)     C[i] += C[i-1];
     for(i = n-1; i >= 0; i --)  sa[--C[a1[a2[i]]]] = a2[i];
     a2[sa[0]] = m = 0;
     for(i = 1; i < n; i ++)     a2[sa[i]] = a1[sa[i-1]]==a1[sa[i]] &&
a1[sa[i-1]+j]==a1[sa[i]+j] ? m : ++ m;
}

void da(char*str, int n, int m) {
     int *a1 = A, *a2 = B, *tmp;
     int i, j, p;
     for(i = 0; i < n; i ++) {
          a1[i] = i;
          a2[i] = str[i];
     }
     a1[n] = a2[n] = -1;
     sortAndRank(a2, a1, n, m, 0);
     for(j = 1; m < n-1; j <<= 1) {
          p = 0;
          for(i = n-j; i < n; i ++)    a2[p ++] = i;
          for(i= 0; i < n; i ++)       if(sa[i]>=j)  a2[p ++] = sa[i]-j;
          sortAndRank(a1, a2, n, m, j);
          tmp = a1; a1 = a2;  a2 = tmp;
     }
     rank = a1;     height = a2;
}


void calHeight(char *str, int n) {
     int i, j, k;
     sa[-1] = n;
     for(height[0] = k = i = 0; i < n; i ++) {
          for(k ? k-- : 0, j = sa[rank[i]-1]; str[i+k]==str[j+k]; k++);
          height[rank[i]] = k;
     }
}

char str[maxn];
int idx[maxn];
int n;
int len;
int ans;
int minLen;

int ok[110];


bool check(int l) {
     int cnt, id;
     for(int i = 0; i < len; i ++) {
          if(height[i] < l) {
               memset(ok, 0, sizeof(ok));
               cnt = 0;
          }
          id = idx[sa[i]];
          if(id != idx[sa[i]+l-1])     continue;
          //if();
          if(!ok[id/2]) {
               ok[id/2] = true;
               cnt ++;
               if(cnt == n)  return true;
          }
     }
     return false;
}

void bsearch() {
     int left = 0;
     int right = minLen + 1;
     int mid;
```

```
    while(left + 1 != right) {
        mid = (left+right)/2;
        if(check(mid))     left = mid;
        else           right = mid;
    }
    ans = left;
}

int main() {
    int t;
    int start;
    for(scanf("%d", &t); t--; ) {
        memset(idx, 255, sizeof(idx));

        scanf("%d", &n);
        start = 0;
        minLen = 99999999;
        for(int i = 0; i < n; i ++) {
            scanf("%s", str + start);
            len = strlen(str + start);
            if(len < minLen)   minLen = len;
            str[start+len] = str[start+2*len+1]= 1;
            for(int j = 0; j < len; j ++) {
                idx[start + j] = 2 * i;
                idx[start+2*len-j] = 2*i+1;
                str[start+2*len-j] = str[start+j];
            }
            start = start + 2*len+2;
        }
        len = start - 1;
        str[len] = 0;
        /*printf("s = [%s]\n", str);
        for(int i = 0; i <= len; i ++)   printf("%d ", idx[i]); printf("\n");
        */
        da(str, len, 127);
        calHeight(str, len);
        bsearch();
        printf("%d\n", ans);
    }
    return 0;
}
//------------------poj-3581------------------------
//把给出的字符串分成三段不空的字串，然后反转各个串组成的最小串。

int str[maxn];
int n;
int arr[maxn], len;
void deal() {
    copy(str, str+n, arr);
    sort(arr, arr+n);
    len = unique(arr, arr+n) - arr;
    for(int i = 0; i < n; i ++) {
        str[i] = lower_bound(arr,arr+len,str[i]) - arr + 1;
    }
}

int main() {
    scanf("%d", &n);
    for(int i = n-1; i >= 0; i --) {
        scanf("%d", str+i);
    }
    deal();
    da(str, n, n);
    int last, mid;
    int i;
    for(i=0; sa[i]<2; i++);
    last = sa[i];
    for(i = last; i < n; i ++) {
        printf("%d\n", arr[str[i]-1]);
    }
    copy(str,str+last,str+last);
    da(str,2*last,n);

    for(i=0; sa[i]==0 || sa[i]>=last; i++);
    mid = sa[i];

    for(int i = mid; i < mid+last; i ++) {
        printf("%d\n", arr[str[i]-1]);
    }
    return 0;
}
//----------------------zoj-3296----------------------------
//   给定一个串，要用最少的回文串将这个串覆盖，回文串可以重叠


#define maxn 100010

int rmq[maxn];
struct ST {
#define CMP  <          //大于为取大数，小于取小数
    int mm[maxn];
    int best[20][maxn];
    void init(int n) {
        int i,j,a,b;
        rmq[0] = 999999999;//让rmq[0]取最反向值
        mm[0]=-1;
        for(i=1; i<=n; i++) {
            mm[i]=((i&(i-1))==0)?mm[i-1]+1:mm[i-1];
            best[0][i]=i;
        }
        for(i=1; i<=mm[n]; i++) {
            for(j=1; j<=n+1-(1<<i); j++) {
                a=best[i-1][j];
                b=best[i-1][j+(1<<(i-1))];
                best[i][j]=rmq[a] CMP rmq[b]?a:b;
            }
        }
    }
    int query(int a, int b) {
        if(a > b)return 0;
        int t;
        t=mm[b-a+1];
        a=best[t][a];
        b=best[t][b-(1<<t)+1];
```

```
          return rmq[a] CMP rmq[b] ? a : b;
     }
} st;

int minString(int * s, int n) {
     int i, j, x, y, u, v;
     for(x=0, y=1; y<n; y++) if(s[y] <= s[x]) {
          i=u=x;    j=v=y;
          while(s[i]==s[j]) {
               ++u; if(++i==n)     i=0;++v;
               if(++j==n)     j=0; if(i==x)  break;
          }
          if(s[i] <= s[j])   y=v;
          else {
               x = y;
               if(u>y)   y=u;
          }
     }
     return x;
}

int A[maxn], B[maxn], S[maxn], C[maxn];
int *rank, *height, *sa = S+1;

void sortAndRank(int *a1, int *a2, int n, int &m, int j) {
     int i;
     memset(C, 0, sizeof(C));
     for(i = 0; i < n; i ++)      C[a1[i]] ++;
     for(i = 1; i <=m; i ++)      C[i] += C[i-1];
     for(i = n-1; i >= 0; i --)  sa[--C[a1[a2[i]]]] = a2[i];
     a2[sa[0]] = m = 0;
     for(i = 1; i < n; i ++)      a2[sa[i]] = a1[sa[i-1]]==a1[sa[i]] &&
a1[sa[i-1]+j]==a1[sa[i]+j] ? m : ++ m;
}

void da(char*str, int n, int m) {
     int *a1 = A, *a2 = B, *tmp;
     int i, j, p;
     for(i = 0; i < n; i ++) {
          a1[i] = i;
          a2[i] = str[i];
     }
     a1[n] = a2[n] = -1;
     sortAndRank(a2, a1, n, m, 0);
     for(j = 1; m < n-1; j <<= 1) {
          p = 0;
          for(i = n-j; i < n; i ++)   a2[p ++] = i;
          for(i= 0; i < n; i ++)      if(sa[i]>=j)  a2[p ++] = sa[i]-j;
          sortAndRank(a1, a2, n, m, j);
          tmp = a1; a1 = a2;  a2 = tmp;
     }
     rank = a1;    height = a2;
}

void calHeight(char *str, int n) {
     int i, j, k;
     sa[-1] = n;
     for(height[0] = k = i = 0; i < n; i ++) {
          for(k ? k-- : 0, j = sa[rank[i]-1]; str[i+k]==str[j+k]; k++);
          height[rank[i]] = k;
     }
}
int lcp(int a, int b) {
     a = rank[a];  b = rank[b];
     if(a > b) swap(a, b);
     return rmq[ st.query(a+1, b) ];
}

char str[maxn];
int n;
typedef pair<int,int> T;
T ts[maxn];

int compute() {
     int len = 0;
     int tmp;

     for(int i = 0; i < n; i ++) {
          tmp = lcp( i, 2*n-i );  //以i为中心
          ts[len ++] = T(i-tmp+1, i+tmp-1);     //闭区间

          if(i != 0 && str[i]==str[i-1]) {
               tmp = lcp( i, 2*n-i+1);
               ts[len ++] = T( i-tmp,i+tmp-1 );
          }
     }
     //区间要覆盖[0,n)!!

     sort(ts, ts+len);

     int left = 0; //左面第一个没有被覆盖的区间
     int i, j;
     int res = 0;
     int right;
     for(i=0; i<len; i=j) {
          right = -1;
          for(j=i; j<len && ts[j].first<=left; j++) right = max(right,
ts[j].second);
          left = right + 1;
          res ++;

          if(left == n) break;
     }
     return res;
}

int main() {
     while(scanf("%s", str) != EOF) {
          n = strlen(str);
          str[n] = 1;
          reverse_copy(str, str+n, str+n+1);
```

```
        str[2*n+1] = 0;

        da(str, 2*n+1, 127);
        calHeight(str, 2*n+1);
        copy(height, height+2*n+1, rmq);
        st.init(2*n);
        //build over!

        int ans = compute();
        printf("%d\n", ans-1);
    }
    return 0;
}

//------------------poj 2758-----------------------
//题意：对原字符串插入若干字符后，动态查询某两个位置的lcp
//RMQ用Square Table作O(nlogn)的预处理和O(1)的询问
//suffix array rmq etc..

#include <stdio.h>
#include <string.h>
#define Min(x,y) ((x)<(y)?(x):(y))
#define maxn 51000
#define INF  0x7FFFFFFF

char s[maxn];
int count[maxn],lcp[17][maxn],smem[3][maxn];
int *SA,*nSA,*Rank,*nRank;
int Ip[300];
char Ic[300];
int Itop,L;

void init(){
    scanf("%s",s);
    L=strlen(s);
    s[L++]=1; s[L]=0;
    Ip[0]=INF; Ic[0]=1; Itop=1;
}

void suffix_array(){
    int i,j,k;
    SA=smem[0]; nSA=smem[1]; Rank=smem[2];
    memset(count,0,sizeof(int)*200);
    for (i=0;i<L;i++) count[s[i]]++;
    for (i=1;i<200;i++) count[i]+=count[i-1];
    for (i=0;i<L;i++) SA[--count[s[i]]]=i;
    for (Rank[SA[0]]=0,i=1;i<L;i++){
        Rank[SA[i]]=Rank[SA[i-1]];
        if (s[SA[i]]!=s[SA[i-1]]) Rank[SA[i]]++;
    }
    for (k=1;k<L&&Rank[SA[L-1]]<L-1;k*=2){
        for (i=0;i<L;i++) count[Rank[SA[i]]]=i+1;
        for (i=L-1;i>=0;i--) if (SA[i]>=k)
            nSA[--count[Rank[SA[i]-k]]]=SA[i]-k;
        for (i=L-k;i<L;i++) nSA[--count[Rank[i]]]=i;
        nRank=SA; SA=nSA;
        for (nRank[SA[0]]=0,i=1;i<L;i++){
            nRank[SA[i]]=nRank[SA[i-1]];
            if (Rank[SA[i]]!=Rank[SA[i-1]]
                    ||Rank[SA[i]+k]!=Rank[SA[i-1]+k])
                nRank[SA[i]]++;
        }
        nSA=Rank; Rank=nRank;
    }
}

void get_lcp_rmq(){
    int i,j,k;
    for (i=0,k=0;i<L;i++){
        if (Rank[i]==L-1) lcp[0][Rank[i]]=k=0;
        else{
            if (k>0) k--;
            j=SA[Rank[i]+1];
            for (;s[i+k]==s[j+k];k++);
            lcp[0][Rank[i]]=k;
        }
    }
    for (i=0,k=1;k<L;i++,k*=2){
        for (j=0;j+k<L;j++){
            lcp[i+1][j]=Min(lcp[i][j],lcp[i][j+k]);
        }
    }
}

int rmq(int a,int b){
    int i,k,t;
    a=Rank[a]; b=Rank[b];
    if (a>b) t=a,a=b,b=t;
    t=b-a;
    for (k=1,i=0;k*2<t;i++,k*=2);
    return Min(lcp[i][a],lcp[i][b-k]);
}

int _Q(int a,int b){
    int na,nb;
    int lr,la,lb,ia,ib,l,res;
    na=a; nb=b; res=0;
    for (ia=0;Ip[ia]<=a;ia++);
    for (ib=0;Ip[ib]<=b;ib++);
    if (a==b) return L-a+Itop-ia-2;
    for (;;){
        lr=rmq(na,nb);
        la=Ip[ia]-na;
        lb=Ip[ib]-nb;
        l=Min(lr,Min(la,lb));
        res+=l; na+=l; nb+=l;
        if (l==la||l==lb){
            while (Ip[ia]==na&&Ip[ib]==nb){
                if (Ic[ia]==Ic[ib]) ia++, ib++, res++;
                else return res;
            }
            while (Ip[ia]==na){
```

```
            if (Ic[ia]==s[nb]) ia++, nb++, res++;
            else return res;
        }
        while (Ip[ib]==nb){
            if (s[na]==Ic[ib]) na++, ib++, res++;
            else return res;
        }
    }else return res;
  }
  return res;
}

void _I(char c,int a){
  int i,j;
  for (i=0;i<Itop;i++){
    if (Ip[i]<a) a--;
    else break;
  }
  if (a>=L) a=L-1;
  for (j=Itop;j>i;j--)
    Ip[j]=Ip[j-1], Ic[j]=Ic[j-1];
  Ip[i]=a; Ic[i]=c; Itop++;
}

void solve(){
  int cas,i,j;
  char str[3];
  scanf("%d",&cas);
  while (cas--){
    scanf("%s",str);
    if (str[0]=='Q'){
      scanf("%d%d",&i,&j);
      printf("%d\n",_Q(i-1,j-1));
    }else{
      scanf("%s%d",str,&j);
      _I(str[0],j-1);
    }
  }
}

int main(){
  init();
  suffix_array();
  get_lcp_rmq();
  solve();
  return 0;
}
```

----------天津网赛_Covering Points.cpp----------

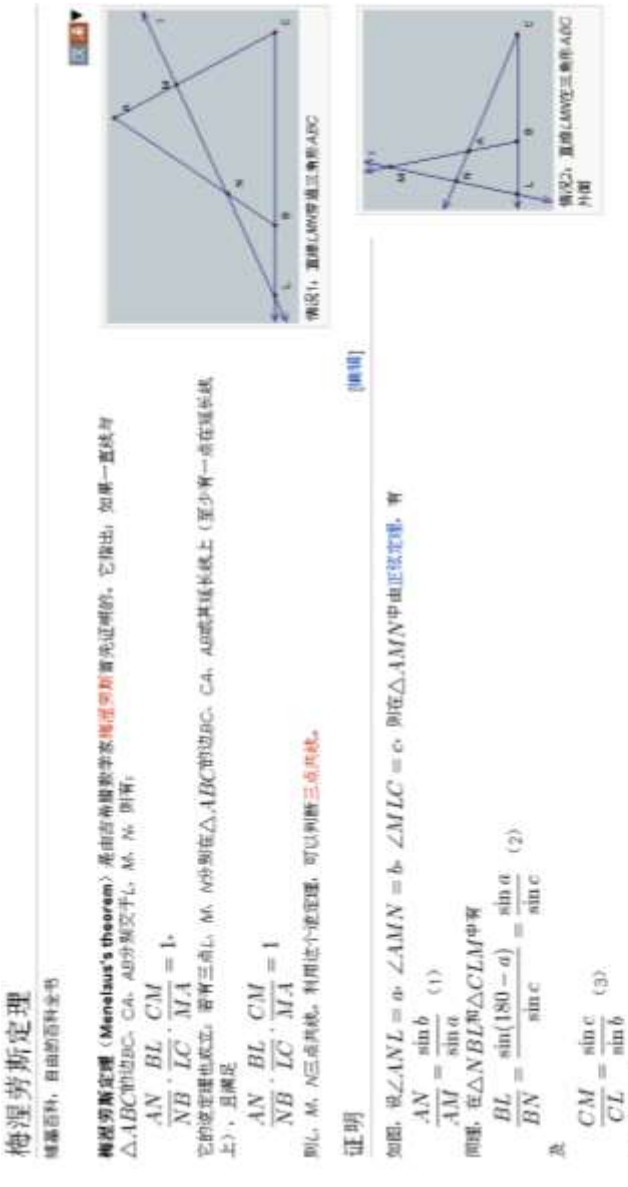
```
//http://acm.tju.edu.cn/toj/showp3740.html

#define maxn 10010
#define M_PI       3.14159265358979323846  /* pi */

int times = 100;

const double eps = 1E-12;

int sig(double d) {
    return (d>eps) - (d<-eps);
}
struct Point {
    double x, y;
    Point(double x, double y) : x(x), y(y) {}
    Point() {}
    Point rotate(double radian) {    //逆时针转
        double c = cos(radian), s = sin(radian);
        return Point(x*c-y*s, y*c+x*s);
    }
    void output() {
        if(sig(x)==0) x = 0;
        if(sig(y)==0) y = 0;
        printf("%.8f %.8f\n", x, y);
    }
};

Point ps[10010];
int n;

double minX, maxX, minY, maxY;

double get(double ang) {
    minX = 1E30, maxX = -1E30;
    minY = 1E30, maxY = -1E30;
    Point p;
    for(int i = 0; i < n; i ++) {
        p = ps[i].rotate(ang);
        minX = min(minX, p.x);
        minY = min(minY, p.y);
        maxX = max(maxX, p.x);
        maxY = max(maxY, p.y);
    }
    return (maxX-minX) - (maxY-minY);
}

double bs(double l, double r) {
    int sigL = sig(get(l));
    double m;
    while(r-l>1E-10) {
        m = (l+r)/2;
        if(sig(get(m))==sigL) {
            l = m;
        } else {
            r = m;
        }
    }
    return l;
}

int main() {
    for(int idx=1; scanf("%d",&n),n; idx ++) {
        for(int i = 0; i < n; i ++) {
            scanf("%lf%lf", &ps[i].x, &ps[i].y);
        }
```

```
        double nowVal, nowAng, nxtVal, nxtAng;
        nowAng=0, nowVal=get(0);

        if(sig(nowVal)==0) goto out;

        int i;
        for(i = 0; i <= times; i ++) {
            nxtAng = M_PI/2.0 * i / times;
            nxtVal = get(nxtAng);

            if(sig(nxtVal)==0) {
                nowAng = nxtAng;
                goto out;
            } else if(sig(nowVal) != sig(nxtVal)) {
                nowAng = bs( nowAng,nxtAng );
                goto out;
            }
            nowAng = nxtAng;
            nowVal = nxtVal;
        }
        while(1);
        out:;

        get(nowAng);
        Point a, b, c, d;
        a = Point(maxX, minY).rotate(-nowAng);
        b = Point(maxX, maxY).rotate(-nowAng);
        c = Point(minX, maxY).rotate(-nowAng);
        d = Point(minX, minY).rotate(-nowAng);

        if(idx!=1)    printf("\n");
        printf("Case %d:\n", idx);

        a.output();
        b.output();
        c.output();
        d.output();
    }
    return 0;
}
```

----------超级圆检测.cpp----------

```
#include <cstdio>
#include <cstring>
#include <cmath>
#include <vector>
#include <cstdlib>
#include <algorithm>
#include <set>
using namespace std;

/**
    poj-2932
        n个圆，互不相交，如果一个圆没有被其他圆包含，则这个圆为super圆
        问那些圆是super圆。
    思路：扫描线
    trick：此题指出了圆互不相交，如果圆有相交的话不知能否这样解，有待测试
 */

#define sqr(v) ((double)(v) * (v))
const int maxn = 40010;
const double eps = 1e-8;

int sig(double d) {
    return (d > eps) - (d < -eps);
}

bool super[maxn];                       //判断某个圆是否为超级圆
double X[maxn], Y[maxn], R[maxn];       //原始圆
int n;                                      //圆的个数
int LL[maxn], RR[maxn];                 //按照左边界和右边界排序的圆下标，这两个是事件点
int tmp[maxn],num[maxn];                //tmp[i]表示纵坐标排序第i名的为tmp[i],num是tmp
的反函数

set<int> tree;                              //树，保存信息

bool in(int a, int b) {     //排名为a的是否在排名为b的圆内
    a = tmp[a];
    b = tmp[b];
    if(sig(R[a]-R[b])>0)    return false;
    if(sig(sqr(X[a]-X[b])+sqr(Y[a]-Y[b])-sqr(R[a]-R[b]))<=0) {
        return true;
    }
    return false;
}
void insert(int v) {    //插入排名为v的点
    set<int>::iterator it = tree.lower_bound(v);
    if (it != tree.end()) {
        if(in(v, *it)) {
            super[tmp[v]] = false;
            return;
        }
    }
    if (it != tree.begin()) {
        if(in(v, *--it)) {
            super[tmp[v]] = false;
            return;
        }
    }
    tree.insert(v);
}
void remove(int v) {    //删除排名为v的点
    /*set<int>::iterator it = tree.find(v);
    if (it == tree.end())   return;
    if (it != tree.begin() && it != --tree.end()) {
        int a = *--it;
        ++it;
        int b = *++it;
        if(in(a,b))   super[tmp[a]]=false;
        if(in(b,a))   super[tmp[b]]=false;
    }
```

```
    */
    //不用加入上面的代码，因为圆在tree中是不重合的!
    tree.erase(v);
}
bool cmpy(int a, int b) {              //按照y坐标关键字排序
    return sig(Y[a]-Y[b])!=0 ? Y[a]<Y[b] : sig(X[a]-X[b])<0;
}
bool cmpl(int a, int b) {              //进入圆时的事件点
    return X[a] - R[a] < X[b] - R[b];
}
bool cmpr(int a, int b) {              //离开圆时的事件点
    return X[a] + R[a] < X[b] + R[b];
}
void solve() {
    for (int i = 0; i < n; ++i) {
        tmp[i] = LL[i] = RR[i] = i;
    }
    sort(tmp, tmp + n, cmpy);
    for (int i = 0; i < n; ++i) {
        num[tmp[i]] = i;
    }
    sort(LL, LL + n, cmpl);
    sort(RR, RR + n, cmpr);

    tree.clear();
    memset(super, 1, sizeof(super));
    int l = 0, r = 0;
    while (l < n || r < n) {
        if (l == n) {
            remove(num[RR[r++]]);
        } else if (r == n) {
            insert(num[LL[l++]]);
        } else if (sig(X[LL[l]] - R[LL[l]] - X[RR[r]] - R[RR[r]]) <= 0) {
            insert(num[LL[l++]]);
        } else {
            remove(num[RR[r++]]);
        }
    }
}

int main() {
    while(scanf("%d", &n) != EOF) {
        for(int i = 0; i < n; i ++) {
            scanf("%lf%lf%lf", R+i, X+i, Y+i);
        }
        solve();
        int ans = 0;
        for(int i = 0; i < n; i ++) {
            if(super[i])  ans ++;
        }
        printf("%d\n", ans);
        for(int i = 0; i < n; i ++) {
            if(super[i])  printf("%d ", i+1);
        }
        printf("\n");
    }
    return 0;
}
```



梅涅劳斯定理

维基百科，自由的百科全书

梅涅劳斯定理（**Menelaus's theorem**）是由古希腊数学家梅涅劳斯首先证明的，它指出：如果一直线与$\triangle ABC$的边BC、CA、AB分别交于L、M、N，则有：

$$\frac{AN}{NB}\cdot\frac{BL}{LC}\cdot\frac{CM}{MA}=1.$$

它的逆定理也成立：若有三点L、M、N分别在$\triangle ABC$的边BC、CA、AB或其延长线上（至少有一点在延长线上），且满足

$$\frac{AN}{NB}\cdot\frac{BL}{LC}\cdot\frac{CM}{MA}=1$$

则L、M、N三点共线。利用这个定理，可以判断三点共线。

[编辑]

证明

如图，设$\angle ANL=a$，$\angle AMN=b$，$\angle MLC=c$，则在$\triangle AMN$中由正弦定理，有

$$\frac{AN}{AM}=\frac{\sin b}{\sin a}\quad(1)$$

同理，在$\triangle NBL$和$\triangle CLM$中有

$$\frac{BL}{BN}=\frac{\sin(180-a)}{\sin c}=\frac{\sin a}{\sin c}\quad(2)$$

及

$$\frac{CM}{CL}=\frac{\sin c}{\sin b}\quad(3)$$

三式相乘，得

$$\frac{AN}{AM}\cdot\frac{BL}{BN}\cdot\frac{CM}{CL}=\frac{\sin b}{\sin a}\cdot\frac{\sin a}{\sin c}\cdot\frac{\sin c}{\sin b}=1.$$

即

$$\frac{AN}{NB}\cdot\frac{BL}{LC}\cdot\frac{CM}{MA}=1.$$