2.郁闷的出纳员

题目

第一行有两个整数 n 和 min。
n 表示下面有多少条命令，min 表示工资下界。
接下来的 n 行，每行一个字符 x 和一个整数 k，表示一条命令。命令可以是以下四种之一：

- `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 k` 把每位员工的工资加上 k。
- `S k` 把每位员工的工资扣除 k。
- `F k` 查询第 k 多的工资。

在初始时，可以认为公司里一个员工也没有。

题解

思路

因为初始工资 $< 1e5$，所以考虑用线段树工资区间内**员工个数的数量**，如 $tree[1,10]$ 表示工资1~10中的员工个数，以维护第 k 大的工资。

而工资变化后，维护一个变化量 Δb，表示**从无员工开始时**工资的总变化量，加入新员工后**将其工资减去 Δb 存入线段树**中。（实际工资没有变化，这样做只是为了统一工资起点，在查询时再加上**查询时的 Δb**）

根据这个定义，可以写出判断第 k 大工资的函数：

```
int kth(int u, int l, int r, int k) // 求第k大的位置（即工资大小）
{
    if (l == r)
        return l;
    int mid = (l + r) >> 1, l1 = -1, r1 = -1;
    if (mi < l)
        l1 = tree[u * 2];
    else if (l <= mi && mi <= mid)
        l1 = query(u * 2, l, mid, mi, mid);
    else if (mid < mi && mi <= r)
        r1 = query(u * 2 + 1, mid + 1, r, mi, r);
    else
        r1 = tree[u * 2 + 1];
    // 左边数量与右边数量
    if (r1 == -1)
        r1 = tree[u] - l1;
    l1 = tree[u] - r1;
    if (r1 >= k)
        return kth(u * 2 + 1, mid + 1, r, k);
    else
        return kth(u * 2, l, mid, k - r1);
}
```

在员工离职后，我们只需遍历员工所在子树，依次减一即可

```
void ql(int u, int l, int r, int x, int y) // 员工离职后清零操作
{
    if (l == r)
    {
        tree[u] = 0;
        return;
    }
    int mid = (l + r) >> 1;
    if (x <= mid && tree[u * 2])
        ql(u * 2, l, mid, x, y);
    if (y > mid && tree[u * 2 + 1])
        ql(u * 2 + 1, mid + 1, r, x, y);
    pushup(u);
}
```

部分代码

```
void pushup(int u) // 更改上移
{
    tree[u] = tree[u * 2] + tree[u * 2 + 1];
}
int query(int u, int l, int r, int x, int y) // 求人数
{
    if (x <= l && r <= y)
        return tree[u];
    int mid = (l + r) >> 1, res = 0;
    if (x <= mid)
        res += query(u * 2, l, mid, x, y);
    if (y > mid)
        res += query(u * 2 + 1, mid + 1, r, x, y);
    return res;
}

void change(int u, int l, int r, int x) // 加人
{
    if (l == r)
    {
        tree[u] += 1;
        return;
    }
    int mid = (l + r) >> 1;
    if (x <= mid)
        change(u * 2, l, mid, x);
    else
        change(u * 2 + 1, mid + 1, r, x);
    pushup(u);
}

int kth(int u, int l, int r, int k) // 求第k大
{
    if (l == r)
        return l;
    int mid = (l + r) >> 1, l1 = -1, r1 = -1;
    if (mi < l)
        l1 = tree[u * 2];
    else if (l <= mi && mi <= mid)
        l1 = query(u * 2, l, mid, mi, mid);
    else if (mid < mi && mi <= r)
        r1 = query(u * 2 + 1, mid + 1, r, mi, r);
    else
        r1 = tree[u * 2 + 1];
    // 左边数量与右边数量
    if (r1 == -1)
        r1 = tree[u] - l1;
    l1 = tree[u] - r1;
    if (r1 >= k)
        return kth(u * 2 + 1, mid + 1, r, k);
    else
        return kth(u * 2, l, mid, k - r1);
}

void ql(int u, int l, int r, int x, int y) // 员工离职后清零操作
```

```
{
    if (l == r)
    {
        tree[u] = 0;
        return;
    }
    int mid = (l + r) >> 1;
    if (x <= mid && tree[u * 2])
        ql(u * 2, l, mid, x, y);
    if (y > mid && tree[u * 2 + 1])
        ql(u * 2 + 1, mid + 1, r, x, y);
    pushup(u);
}
```