```cpp
#include "../bits/stdc++.h"
#include "../Others/xor_shift.hpp"

XorShift rnd(114514);

/**
 * merge/split ベース Treap
 * insert, erase, k-th element ができる
 */
// verified: https://atcoder.jp/contests/arc033/submissions/4342542
template <typename T>
class Treap
{
  public:
    struct Node
    {
        T val;
        T sum; // sum of subtree's values
        Node *left;
        Node *right;
        int pri;
        int sz; // size of subtree

        Node(T v, int p) : val(v), pri(p), sz(1), sum(v)
        {
            left = right = NULL;
        }
    };

    Node *root;
    Treap() { root = NULL; }
    int size(Node *t) { return !t ? 0 : t->sz; }
    int size() { return size(root); }
    T sum(Node *t) { return !t ? 0 : t->sum; }

    Node *update(Node *t)
    {
        t->sz = size(t->left) + size(t->right) + 1;
        t->sum = sum(t->left) + sum(t->right) + t->val;
        return t;
    }

    // Treap l, Treap r をマージ
    Node *merge(Node *l, Node *r)
    {
        if (!l || !r)
            return l ? l : r;

        if (l->pri > r->pri)
        {
            l->right = merge(l->right, r);
            return update(l);
        }
        r->left = merge(l, r->left);
        return update(r);
    }

    // 位置 [0,k), [k,n) で木を分割
    std::pair<Node *, Node *> split(Node *t, int k)
    {
        if (!t)
            return pair<Node *, Node *>(NULL, NULL);

        if (k <= size(t->left))
        {
            auto s = split(t->left, k);
            t->left = s.second;
            return pair<Node *, Node *>(s.first, update(t));
        }
        auto s = split(t->right, k - size(t->left) - 1);
        t->right = s.first;
        return pair<Node *, Node *>(update(t), s.second);
    }

    Node *insert(Node *t, int k, T val)
    {
        auto s = split(t, k);
        auto r = merge(s.first, new Node(val, rnd.rand()));
        r = merge(r, s.second);
        return r;
    }

    Node *erase(Node *t, int k)
    {
        auto u = split(t, k), v = split(u.second, 1);
        if (v.first)
            delete v.first;
        return merge(u.first, v.second);
    }

    Node *find(Node *t, int k)
    {
        if (!t)
            return t;
        int sz = size(t->left);
```

```
96          return k < sz ? find(t->left, k) : k == sz ? t : find(t->right, k - (sz + 1));
97      }
98
99      // val 未満の要素の個数
100     int count(Node *t, T val)
101     {
102         if (!t)
103             return 0;
104         if (t->val < val)
105             return size(t->left) + 1 + count(t->right, val);
106         if (t->val == val)
107             return size(t->left);
108         return count(t->left, val);
109     }
110
111     // 値 val を挿入
112     void insert(T val) { root = insert(root, count(root, val), val); }
113     // 位置 k のノードを削除
114     void erase(int k) { root = erase(root, k); }
115     // k 番目の値を求める
116     T find(int k)
117     {
118         auto t = find(root, k);
119         // ここ何とかして
120         if (!t)
121             return -1;
122         return t->val;
123     }
124 };
125
```