```cpp
class Treap {
    static mt19937_64 generator;

    struct Node {
        int key, priority;
        int value, max_val, add = 0;
        Node *l = nullptr, *r = nullptr;
        Node (int key, int value): key(key), priority(generator()), value(value), max_val(value) {}
    } *root = nullptr;

    static int getMaxValue(Node *n){
        return n ? n -> max_val + n -> add : -1e9;
    }

    static void push(Node *&n){
        if (n && n -> add){
            n -> value += n -> add;
            n -> max_val += n -> add;
            if (n -> l){
                n -> l -> add += n -> add;
            }
            if (n -> r){
                n -> r -> add += n -> add;
            }
            n -> add = 0;
        }
    }

    static void update(Node *&n){
        if (n){
            n -> max_val = max(max(getMaxValue(n -> l), n -> value), getMaxValue(n -> r));
        }
    }

    static Node *merge(Node *a, Node *b){
        push(a);
        push(b);
        if (!a || !b){
            return a ? a : b;
        }
        if (a->priority > b->priority){
            a->r = merge(a->r, b);
            update(a);
            return a;
        }
        else {
            b->l = merge(a, b->l);
            update(b);
            return b;
        }
    }

    static void split(Node *n, int key, Node *&a, Node *&b){
        push(n);
        if (!n){
            a = b = nullptr;
            return ;
        }
        if (n -> key < key){
            //      a = n
            // n->l        a' b'
            split(n->r, key, n->r, b);
            a = n;
        }
        else {
            split(n->l, key, a, n->l);
            b = n;
        }
        update(a);
        update(b);
    }

public:

    bool find(int key){
        Node *greater, *equal, *less;
        split(root, key, less, greater);
        split(greater, key + 1, equal, greater);
        bool result = equal;
        root = merge(merge(less, equal), greater);
        return result;
    }

    void insert(int key, int value){
        Node *greater, *less;
        split(root, key, less, greater);
        less = merge(less, new Node(key, value));
        root = merge(less, greater);
    }

    void erase(int key){
        Node *greater, *equal, *less;
        split(root, key, less, greater);
        split(greater, key + 1, equal, greater);
        root = merge(less, greater);
    }

    long long getMax(int l, int r){
        Node *left, *middle, *right;
                split(root, l, left, middle);
                split(middle, r + 1, middle, right);
                long long ans = getMaxValue(middle);
                root = merge(merge(left, middle), right);
                return ans;
            }

            void rangeAdd(int l, int r, int value){
                Node *left, *middle, *right;
                split(root, l, left, middle);
                split(middle, r + 1, middle, right);
                if (middle){
                    middle -> add += value;
                }
                root = merge(merge(left, middle), right);
            }

        };

        mt19937_64 Treap::generator;
```