```cpp
#include <bits/stdc++.h>

using namespace std;

template<typename T>
class AVL;

template<typename T>
void delete_avl(AVL<T> *p);

// f(l, r) -> l < r
template<typename T>
class AVL {
  T data;
  function<bool(T, T)> f;
  int size, height, balance;
  AVL<T> *left, *right;
public:
  AVL<T>(T data, function<bool(T, T)> f) : data(data), f(f) {
    size = 1;
    height = 1;
    balance = 0;
    left = right = nullptr;
  }

  void update() {
    int l_height = 0, r_height = 0, l_size = 0, r_size;
    if (left != nullptr) {
      l_height = left->height;
      l_size = left->size;
    }
    if (right != nullptr) {
      r_height = right->height;
      r_size = right->size;
    }
    height = max(l_height, r_height) + 1;
    size = 1 + l_size + r_size;
    balance = l_height - r_height;
  }

  void print() {
    cerr << "(";
    if (left != nullptr) left->print();
    cerr << data;
    if (right != nullptr) right->print();
    cerr << ")";
  }

  AVL<T> *rotate_r() {
    AVL<T> *l = left, *lr = l->right;
    l->right = this;
    left = lr;
    update();
    l->update();
    return l;
  }

  AVL<T> *rotate_l() {
    AVL<T> *r = right, *rl = r->left;
    r->left = this;
    right = rl;
    update();
    r->update();
    return r;
  }

  AVL<T> *insert(T val) {
    if (f(val, data)) {
      if (left == nullptr) {
        left = new AVL<T>(val, f);
      } else {
        left = left->insert(val);
      }
    } else {
      if (right == nullptr) {
        right = new AVL<T>(val, f);
      } else {
        right = right->insert(val);
      }
    }
    update();
    if (balance < -1) {
      if (right->balance == 1) right = right->rotate_r();
      return rotate_l();
    } else if (balance > 1) {
      if (left->balance == -1) left = left->rotate_l();
      return rotate_r();
    }
    return this;
  }

  // バグってる
  AVL<T> *erase(T val) {
    if (f(val, data)) {
      if (left != nullptr) {
        left = left->erase(val);
      }
    } else if (f(data, val)) {
      if (right != nullptr) {
        right = right->erase(val);
```

```cpp
      }
    } else {
      // delete all
      // delete_avl(this);

      if (left != nullptr) {
        left = left->erase(val);
      } else if (right != nullptr) {
        right = right->erase(val);
      } else {
        delete this;
        return nullptr;
      }
    }
    update();
    if (balance < -1) {
      if (right->balance == -1) right = right->rotate_l();
      return rotate_l();
    } else if (balance > 1) {
      if (left->balance == 1) left = left->rotate_r();
      return rotate_r();
    }
    return this;
  }
};

template<typename T>
void delete_avl(AVL<T> *p) {
  if (p == nullptr) return;
  delete_avl<T>(p->left);
  delete_avl<T>(p->right);
  delete p;
}

using avl = AVL<int>;

int main() {
  avl *root = new avl(5, [](int l, int r) {
    return l < r;
  });
  for (int i = 0; i < 10; i++) {
    root = root->insert(i);
    root->print();
    cerr << endl;
  }
  for (int i = 0; i < 10; i++) {
    root = root->erase(i);
    root->print();
    cerr << endl;
  }
```

```
    return 0;
}
```