```
//  Name: Matthew Jensen
//  Assignment number: 4
//  Assignment: Binary Tree Traversals
//  File name: tree_extra.hpp
//  Date last modified: October 19, 2021
//  Honor statement: I have neither given nor received any unauthorized help on this assignment.

// Draws the binary tree rooted at t.
// Parameter link is a symbol to print in front of the node to
// which t points indicating the direction of the branch leading
// to the node.
// Parameter depth is proportional to depth of the node to which p
// points.
template <typename T>
static void draw(TreeNode<T> *t, char link, int depth) {
    if (t == nullptr)
        return;

    //print right subtree
    draw(t->right, '/', ++(++depth));

    //print node
    for (int i = 0; i < depth; i++){
        std::cout << " ";
    }
    std::cout << link << "[" << t->data << "]\n";

    //print left subtree
    draw(t->left, '\\', ++(++depth));
}


// Frees up the space held by the nodes in a binary tree
// rooted at t.
template <typename T>
void dispose(TreeNode<T> *t) {
    if (t == nullptr){
        return;
    }

    //delete right subtree
    dispose(t->right);

    //delete left subtree
    dispose(t->left);

    //delete node
    delete(t);

    //set t to nullptr
    t = nullptr;
}


// Builds a binary tree from preorder and inorder traversals.
// Parameter pre_begin is an iterator to the beginning of the
// preorder traveral sequence.
// Parameter pre_end is an iterator to the end of the preorder
// traveral sequence.
// Parameter in_begin is an iterator to the beginning of the
// inorder traveral sequence.
// Parameter in_end is an iterator to the end of the inorder
// traveral sequence.
// Returns a pointer to the root of the newly created binary tree.
template <typename T>
static TreeNode<T> *build_tree(typename std::vector<T>::const_iterator pre_begin,
                               typename std::vector<T>::const_iterator pre_end,
                               typename std::vector<T>::const_iterator in_begin,
                               typename std::vector<T>::const_iterator in_end) {

    int index = find(in_begin, in_end, *pre_begin) - in_begin;
    typename std::vector<T>::const_iterator pre_index_iter = pre_begin + index;
    typename std::vector<T>::const_iterator in_index_iter = find(in_begin, in_end, *pre_begin);

    std::vector<T> left_pre(next(pre_begin), next(pre_index_iter));
    std::vector<T> left_in(in_begin, in_index_iter);
    std::vector<T> right_pre(next(pre_index_iter), pre_end);
    std::vector<T> right_in(next(in_index_iter), in_end);

    TreeNode<T> *root = new TreeNode(*pre_begin,
        ( begin(left_pre) == end(left_pre) )   ? nullptr : (build_tree<T>(begin(left_pre), end(left_pre), begin(left_in),
end(left_in))),
        ( begin(right_pre) == end(right_pre) ) ? nullptr : (build_tree<T>(begin(right_pre), end(right_pre), begin(right_in),
end(right_in))));
```

Commented [RH1]:
See notes below

9.5/10

Commented [RH2]: This is superfluous, as it will not change the value of the variable the caller passed in

Commented [RH3]: Fails for empty vectors; should make an empty tree (`nullptr`)

```
    return root;
}
```