```

// COS30008, Problem Set 4, Problem 2, 2025

#pragma once

#include "BinaryTreeNode.h"

#include <stdexcept>

// Problem 3 requirement
template<typename T>
class BinarySearchTreeIterator;

template<typename T>
class BinarySearchTree
{
private:

    using BNode = BinaryTreeNode<T>;
    using BTreeNode = BNode*;

    BTreeNode fRoot;

public:
    // default constructor
    BinarySearchTree() :
        fRoot(&BNode::NIL)
    { }

    // destructor
    ~BinarySearchTree()
    {
        // avoid deleting NIL
        if (!empty())
        {
            delete fRoot;
        }
    }

    bool empty() const
    {
        return fRoot->empty();
    }

    size_t height() const
    {
        if (empty())
        {
            throw std::domain_error("Empty tree has no height.");
        }

        return fRoot->height();
    }

    bool insert(const T& aKey)
    {
        // If tree is empty, create a new root
        if (empty())
        {
            fRoot = new BNode(aKey);
            return true;
        }

        // else, insert into the tree
        return fRoot->insert(aKey);
    }

    bool remove(const T& aKey)
    {
        if (empty())
        {
            throw std::domain_error("Cannot remove from an empty tree.");
        }

        // If fRoot is the only node in the tree,
        // delete it and set the root to NIL
        if (aKey == fRoot->key && fRoot->leaf())
        {
            delete fRoot;
            fRoot = &BNode::NIL;
            return true;
        }

        return fRoot->remove(aKey, &BNode::NIL);
    }

    // Problem 3 methods

    using Iterator = BinarySearchTreeIterator<T>;

    // Allow iterator to access private member variables
    friend class BinarySearchTreeIterator<T>;

    Iterator begin() const
    {
        return Iterator(*this);
    }

    Iterator end() const
    {
        return Iterator(*this).end();
    }
};

```