```cpp
#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:
    BinarySearchTree() : fRoot((&BNode::NIL)) {}
    ~BinarySearchTree()
    {
        if (!fRoot->empty())
        {
            delete fRoot;
        }
    }
    bool empty() const
    {
        return fRoot->empty();
    }
    size_t height() const
    {
        if (empty())
        {
            throw domain_error("Empty tree has no height.");
        }
        return fRoot->height();
    }

    bool insert(const T& aKey)
    {
        if (empty())
        {
            fRoot = new BNode(aKey);
            return true;
        }
        return fRoot->insert(aKey);
    }
    bool remove(const T& aKey)
    {
        if (empty())
        {
            throw domain_error("Cannot remove in empty tree.");
        }
        if (fRoot->leaf())
        {
            if (fRoot->key != aKey)
            {
                return false;
            }
            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).begin();
    }
    Iterator end() const
    {
        return Iterator(*this).end();
    }
};
```