```
 1
 2  // COS30008, Problem Set 4, Problem 2, 2025
 3
 4  #pragma once
 5
 6  #include "BinaryTreeNode.h"
 7
 8  #include <stdexcept>
 9
10  // Problem 3 requirement
11  template<typename T>
12  class BinarySearchTreeIterator;
13
14  template<typename T>
15  class BinarySearchTree
16  {
17  private:
18
19      using BNode = BinaryTreeNode<T>;
20      using BTreeNode = BNode*;
21
22      BTreeNode fRoot;
23
24  public:
25      // default constructor
26      BinarySearchTree() :
27          fRoot(&BNode::NIL)
28      { }
29
30      // destructor
31      ~BinarySearchTree()
32      {
33          // avoid deleting NIL
34          if (!empty())
35          {
36              delete fRoot;
37          }
38      }
39
40      bool empty() const
41      {
42          return fRoot->empty();
43      }
44
45      size_t height() const
46      {
47          if (empty())
48          {
49              throw std::domain_error("Empty tree has no height.");
```

```
50          }
51
52          return fRoot->height();
53      }
54
55      bool insert(const T& aKey)
56      {
57          // If tree is empty, create a new root
58          if (empty())
59          {
60              fRoot = new BNode(aKey);
61              return true;
62          }
63
64          // else, insert into the tree
65          return fRoot->insert(aKey);
66      }
67
68      bool remove(const T& aKey)
69      {
70          if (empty())
71          {
72              throw std::domain_error("Cannot remove from an empty tree.");
73          }
74
75          // If fRoot is the only node in the tree,
76          // delete it and set the root to NIL
77          if (aKey == fRoot->key && fRoot->leaf())
78          {
79              delete fRoot;
80              fRoot = &BNode::NIL;
81              return true;
82          }
83
84          return fRoot->remove(aKey, &BNode::NIL);
85      }
86
87      // Problem 3 methods
88
89      using Iterator = BinarySearchTreeIterator<T>;
90
91      // Allow iterator to access private member variables
92      friend class BinarySearchTreeIterator<T>;
93
94      Iterator begin() const
95      {
96          return Iterator(*this);
97      }
98
```

```
 99      Iterator end() const
100      {
101          return Iterator(*this).end();
102      }
103  };
104
```