

```
1
2 // COS30008, Problem Set 4, Problem 2, 2022
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     using BNode = BinaryTreeNode<T>;
19     using BTreeNode = BNode*;
20
21     BTreeNode fRoot;
22
23 public:
24     BinarySearchTree()
25     {
26         fRoot = &BNode::NIL;
27     }
28
29     ~BinarySearchTree() {
30         if (!empty())
31         {
32             if (fRoot->left != &BNode::NIL)
33             {
34                 delete fRoot->left;
35             }
36             if (fRoot->right != &BNode::NIL)
37             {
38                 delete fRoot->right;
39             }
40         }
41     }
42
43     bool empty() const
44     {
45         return fRoot->empty();
46     }
47 }
```

```
50
51     size_t height() const
52     {
53         if (fRoot == NULL)
54         {
55             return 0;
56         }
57         return fRoot->height();
58     }
59
60     bool insert(const T& aKey)
61     {
62         if (empty()) {
63             fRoot = new BNode(aKey);
64             return true;
65         }
66         return fRoot->insert(aKey);
67     }
68
69     bool remove(const T& aKey) {
70
71         //return fRoot->remove(aKey, &BTreeNode::NIL);
72         //return fRoot->remove(aKey, fRoot->NIL);
73         return false;
74
75     }
76
77     // Problem 3 methods
78
79     using Iterator = BinarySearchTreeIterator<T>;
80
81     // Allow iterator to access private member variables
82     friend class BinarySearchTreeIterator<T>;
83
84     Iterator begin() const
85     {
86         return BinarySearchTree<T>(*this);
87     }
88     Iterator end() const
89     {
90         return begin().end();
91     }
92 };
93
```