

Swinburne University of Technology
Faculty of Science, Engineering and Technology

ASSIGNMENT COVER SHEET

Subject Code: COS30008
Subject Title: Data Structures and Patterns
Assignment number and title: 4, Binary Search Trees & In-Order Traversal
Due date: May 26, 2022, 14:30
Lecturer: Dr. Markus Lumpe

Your name: _____ **Your student id:** _____

Check Tutorial	Mon 10:30	Mon 14:30	Tues 08:30	Tues 10:30	Tues 12:30	Tues 14:30	Tues 16:30	Wed 08:30	Wed 10:30	Wed 12:30	Wed 14:30

Marker's comments:

Problem	Marks	Obtained
1	94	
2	42	
3	8+86=94	
Total	230	

Extension certification:

This assignment has been given an extension and is now due on _____

Signature of Convener: _____

Problem 1: BinaryTreeNode.h

... 4\Student_Project1\Student_Project1\BinaryTreeNode.h

1

```
1
2 // COS30008, Problem Set 4, Problem 1, 2022
3
4 #pragma once
5
6 #include <stdexcept>
7 #include <algorithm>
8
9 template<typename T>
10 struct BinaryTreeNode
11 {
12     using BNode = BinaryTreeNode<T>;
13     using BTreeNode = BNode*;
14
15     T key;
16     BTreeNode left;
17     BTreeNode right;
18
19     static BNode NIL;
20
21     const T& findMax() const
22     {
23         if ( empty() )
24         {
25             throw std::domain_error( "Empty tree encountered." );
26         }
27
28         return right->empty() ? key : right->findMax();
29     }
30
31     const T& findMin() const
32     {
33         if ( empty() )
34         {
35             throw std::domain_error( "Empty tree encountered." );
36         }
37
38         return left->empty() ? key : left->findMin();
39     }
40
41     bool remove( const T& aKey, BTreeNode aParent )
42     {
43         BTreeNode x = this;
44         BTreeNode y = aParent;
45
46         while ( !x->empty() )
47         {
48             if ( aKey == x->key )
49             {
```

```

50         break;
51     }
52
53     y = x;                                     // new parent
54
55     x = aKey < x->key ? x->left : x->right;
56 }
57
58 if ( x->empty() )
59 {
60     return false;                             // delete failed
61 }
62
63 if ( !x->left->empty() )
64 {
65     const T& lKey = x->left->findMax();          // find max to
66     left
67     x->key = lKey;
68     x->left->remove( lKey, x );
69 }
70 else
71 {
72     if ( !x->right->empty() )
73     {
74         const T& lKey = x->right->findMin();     // find min to
75         right
76         x->key = lKey;
77         x->right->remove( lKey, x );
78     }
79     else
80     {
81         if ( y != &NIL )                       // y can be NIL
82         {
83             if ( y->left == x )
84             {
85                 y->left = &NIL;
86             }
87             else
88             {
89                 y->right = &NIL;
90             }
91         }
92
93         delete x;                               // free deleted
94         node
95     }
96 }
97
98 return true;

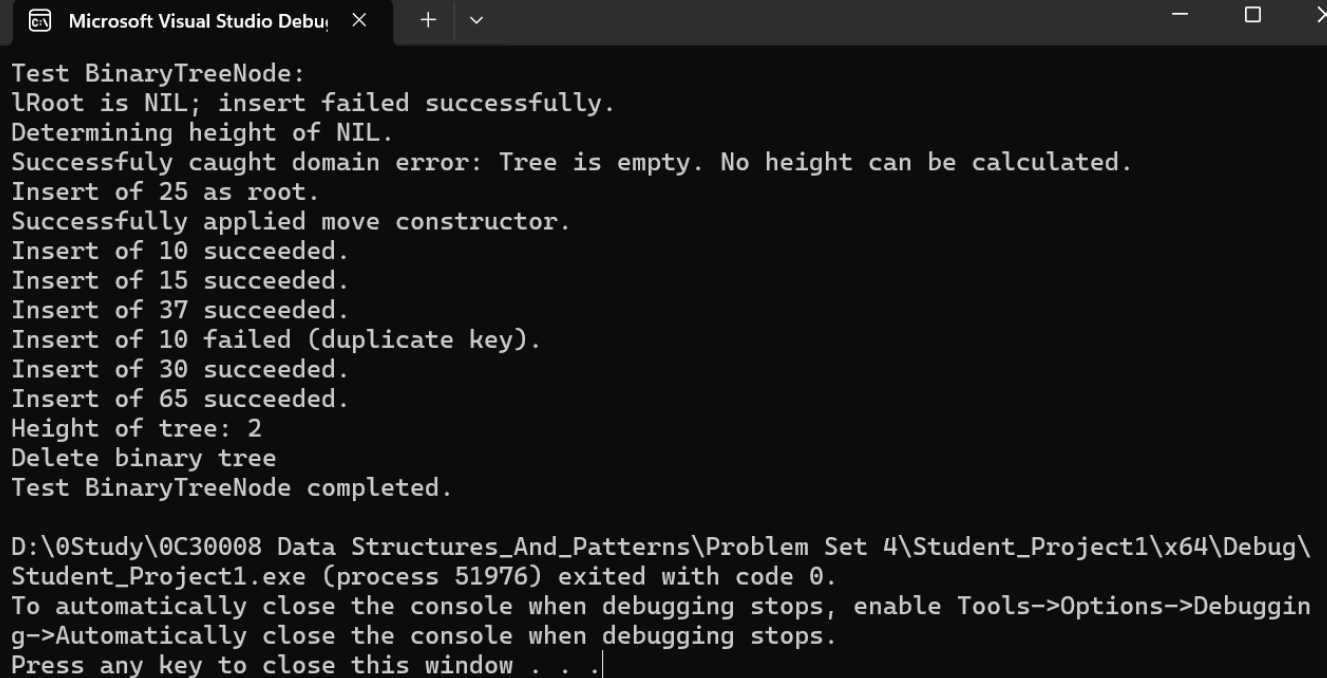
```

```
96     }
97
98     // PS4 starts here
99
100    BinaryTreeNode() : // Default constructor
101        key(T()),
102        left(&NIL),
103        right(&NIL)
104    {}
105
106    BinaryTreeNode(const T& aKey) : // Copy constructor
107        key(aKey),
108        left(&NIL),
109        right(&NIL)
110    {}
111
112    BinaryTreeNode(T&& aKey) :// Move constructor
113        key(std::move(aKey)),
114        left(&NIL),
115        right(&NIL)
116    {}
117
118    ~BinaryTreeNode() // Destructor
119    {
120        if (!left->empty())
121        {
122            delete left;
123        }
124
125        if (!right->empty())
126        {
127            delete right;
128        }
129    }
130
131    bool empty() const // Check if node is empty
132    {
133        return this == &NIL;
134    }
135
136    bool leaf() const // Check if the elements is leaf or not
137    {
138        return left->empty() && right->empty();
139    }
140
141    size_t height() const
142    {
143        if (empty())
144        {
```

```
145         throw std::domain_error("Tree is empty. No height can be  
            calculated.");  
146     }  
147  
148     if (leaf())  
149     {  
150         return 0;  
151     }  
152     else  
153     {  
154         size_t leftHeight = 0;  
155         size_t rightHeight = 0;  
156  
157         if (!left->empty())  
158         {  
159             leftHeight = left->height();    // đổi gốc  
160         }  
161  
162         if (!right->empty())  
163         {  
164             rightHeight = right->height();  // đổi gốc  
165         }  
166  
167         return max(leftHeight, rightHeight) + 1;  
168     }  
169 }  
170  
171 bool insert(const T& aKey)  
172 {  
173     BTreeNode x = this;  
174     BTreeNode y = &NIL;  
175  
176     while (!x->empty())  
177     {  
178         y = x;  
179  
180         if (aKey == x->key)  
181         {  
182             return false;    // duplicate key - error  
183         }  
184  
185         x = aKey < x->key ? x->left : x->right;  
186     }  
187  
188     BTreeNode z = new BNode(aKey);  
189  
190     if (y->empty())  
191     {  
192         return false;    // insert failed (NIL) - emptyy
```

```
193     }
194     else
195     {
196         if (aKey < y->key)
197         {
198             y->left = z;
199         }
200         else
201         {
202             y->right = z;
203         }
204     }
205
206     return true;           // insert done
207 }
208 };
209
210 template<typename T>
211 BinaryTreeNode<T> BinaryTreeNode<T>::NIL;
212
```

Output



The screenshot shows the Microsoft Visual Studio Debug Console window. The title bar reads 'Microsoft Visual Studio Debug Console'. The output text is as follows:

```
Test BinaryTreeNode:
lRoot is NIL; insert failed successfully.
Determining height of NIL.
Successfully caught domain error: Tree is empty. No height can be calculated.
Insert of 25 as root.
Successfully applied move constructor.
Insert of 10 succeeded.
Insert of 15 succeeded.
Insert of 37 succeeded.
Insert of 10 failed (duplicate key).
Insert of 30 succeeded.
Insert of 65 succeeded.
Height of tree: 2
Delete binary tree
Test BinaryTreeNode completed.

D:\0Study\0C30008 Data Structures_And_Patterns\Problem Set 4\Student_Project1\x64\Debug\
Student_Project1.exe (process 51976) exited with code 0.
To automatically close the console when debugging stops, enable Tools->Options->Debugging
g->Automatically close the console when debugging stops.
Press any key to close this window . . .|
```

Problem 2: BinarySearchTree.h

...\Student_Project1\Student_Project1\BinarySearchTree.h

1

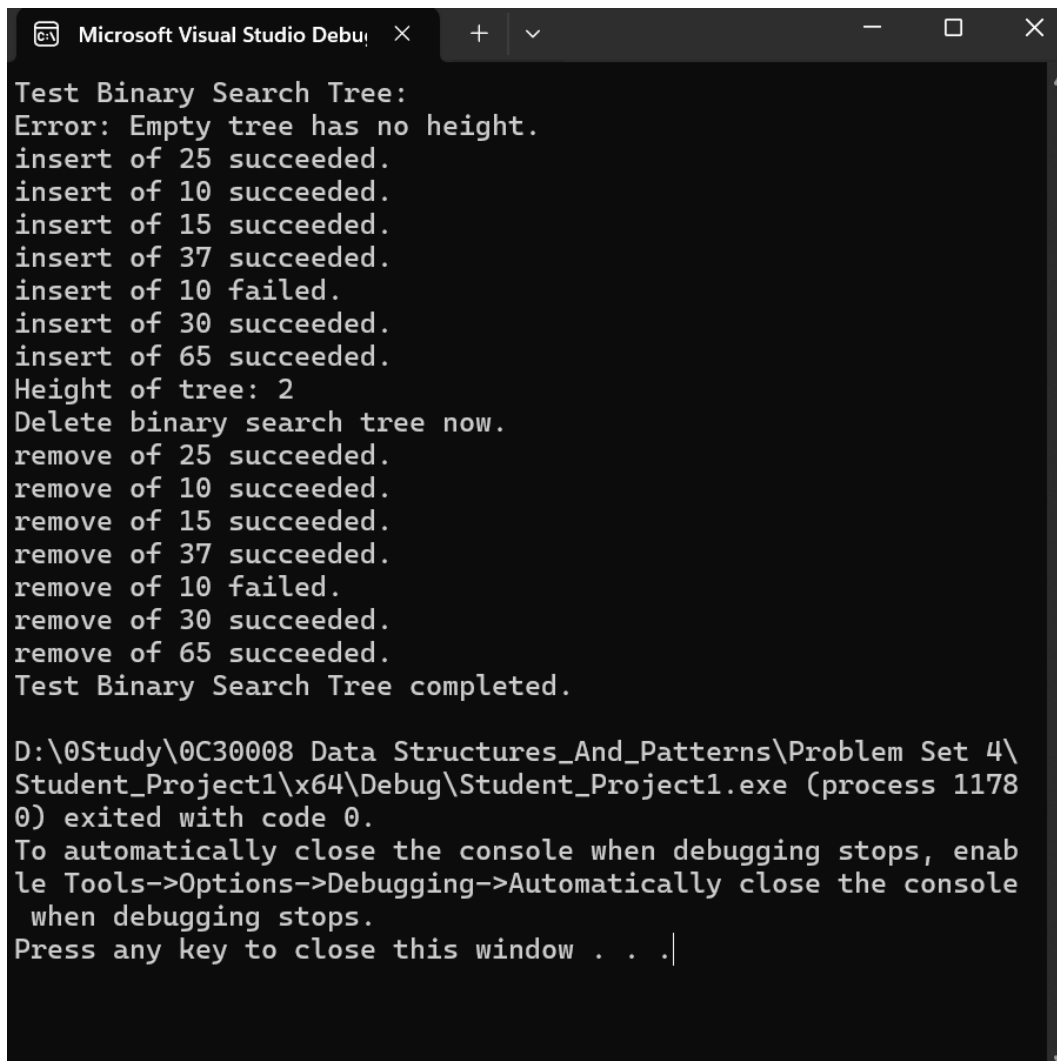
```
1
2 // COS30008, Problem Set 4, Problem 2, 2022
3
4 #pragma once
5
6 #include "BinaryTreeNode.h"
7
8 #include <stdexcept>
9
10 // Problem 3 requirement
11 template<typename T>
12 class BinarySearchTreeIterator;
13
14 template<typename T>
15 class BinarySearchTree
16 {
17 private:
18
19     using BNode = BinaryTreeNode<T>;
20     using BTreeNode = BNode*;
21
22     BTreeNode fRoot;
23
24 public:
25
26     BinarySearchTree()
27     {
28         fRoot = &BNode::NIL;
29     }
30
31     ~BinarySearchTree()
32     {
33         if ( !empty() )
34         {
35             delete fRoot;
36         }
37     }
38
39     bool empty() const
40     {
41         return fRoot == &BNode::NIL;
42     }
43
44     size_t height() const
45     {
46         // The empty tree has no height.
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 ( empty() )
58     {
59         fRoot = new BNode( aKey );
60
61         return true;
62     }
63     else
64     {
65         return fRoot->insert( aKey );
66     }
67 }
68
69 bool remove( const T& aKey )
70 {
71     if ( empty() )
72     {
73         throw std::domain_error( "remove(): NIL encountered." );
74     }
75
76     if ( fRoot->leaf() )
77     {
78         // last node
79         if ( fRoot->key == aKey )
80         {
81             delete fRoot;
82             fRoot = &BNode::NIL;
83
84             return true;
85         }
86
87         return false;
88     }
89     else
90     {
91         return fRoot->remove( aKey, &BNode::NIL );
92     }
93 }
94
95 // Problem 3 methods
96
97 using Iterator = BinarySearchTreeIterator<T>;
98
```



```
99 // Allow iterator to access private member variables
100 friend class BinarySearchTreeIterator<T>;
101
102 Iterator begin() const
103 {
104     return Iterator( *this );
105 }
106
107 Iterator end() const
108 {
109     return begin().end();
110 }
111 };
112
```

Output



```
Microsoft Visual Studio Debug Console
Test Binary Search Tree:
Error: Empty tree has no height.
insert of 25 succeeded.
insert of 10 succeeded.
insert of 15 succeeded.
insert of 37 succeeded.
insert of 10 failed.
insert of 30 succeeded.
insert of 65 succeeded.
Height of tree: 2
Delete binary search tree now.
remove of 25 succeeded.
remove of 10 succeeded.
remove of 15 succeeded.
remove of 37 succeeded.
remove of 10 failed.
remove of 30 succeeded.
remove of 65 succeeded.
Test Binary Search Tree completed.

D:\0Study\0C30008 Data Structures_And_Patterns\Problem Set 4\
Student_Project1\x64\Debug\Student_Project1.exe (process 1178
0) exited with code 0.
To automatically close the console when debugging stops, enab
le Tools->Options->Debugging->Automatically close the console
when debugging stops.
Press any key to close this window . . .|
```

Problem 3: BinarySearchTreeIterator.h

..._Project1\Student_Project1\BinarySearchTreeIterator.h

1

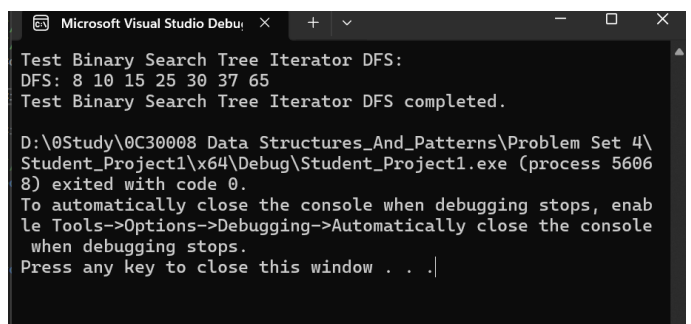
```
1
2 // COS30008, Problem Set 4, Problem 3, 2022
3
4 #pragma once
5
6 #include "BinarySearchTree.h"
7
8 #include <stack>
9
10 template<typename T>
11 class BinarySearchTreeIterator
12 {
13 private:
14
15     using BSTree = BinarySearchTree<T>;
16     using BNode = BinaryTreeNode<T>;
17     using BTreeNode = BNode*;
18     using BTNStack = std::stack<BTreeNode>;
19
20     const BSTree& fBSTree;
21     BTNStack fStack;
22
23     void pushLeft( BTreeNode aNode )
24     {
25         while ( !aNode->empty() )
26         {
27             fStack.push( aNode );
28             aNode = aNode->left;
29         }
30     }
31
32 public:
33
34     using Iterator = BinarySearchTreeIterator<T>;
35
36     BinarySearchTreeIterator( const BSTree& aBSTree ) :
37         fBSTree(aBSTree)
38     {
39         pushLeft( fBSTree.fRoot );
40     }
41
42     const T& operator*() const
43     {
44         return fStack.top()->key;
45     }
46
47     Iterator& operator++()
48     {
49         BTreeNode newNode = fStack.top()->right;
```

```

50
51     fStack.pop();
52
53     pushLeft(newNode);
54
55     return *this;
56 }
57
58 Iterator operator++(int)
59 {
60     Iterator temp = *this;
61
62     ++(*this);
63
64     return temp;
65 }
66
67 bool operator==( const Iterator& aOtherIter ) const
68 {
69     return
70         &fBSTree == &aOtherIter.fBSTree &&
71         fStack.size() == aOtherIter.fStack.size();
72 }
73
74 bool operator!=( const Iterator& aOtherIter ) const
75 {
76     return !(*this == aOtherIter);
77 }
78
79 Iterator begin() const
80 {
81     Iterator iter = *this;
82
83     iter.fStack = BTNStack();
84     iter.pushLeft( iter.fRoot );
85
86     return iter;
87 }
88
89 Iterator end() const
90 {
91     Iterator iter = *this;
92
93     iter.fStack = BTNStack();
94
95     return iter;
96 }
97 };
98

```

Output



```

Microsoft Visual Studio Debug Console
Test Binary Search Tree Iterator DFS:
DFS: 8 10 15 25 30 37 65
Test Binary Search Tree Iterator DFS completed.

D:\0Study\0C30008 Data Structures_And_Patterns\Problem Set 4\
Student_Project1\x64\Debug\Student_Project1.exe (process 5606
8) exited with code 0.
To automatically close the console when debugging stops, enab
le Tools->Options->Debugging->Automatically close the console
when debugging stops.
Press any key to close this window . . .

```