

```

1  #include "BSTree.h"
2
3  BSTree::BSTree()
4  : pObjBSTNodeRoot( NULL ), uintSize( 0 )
5  {
6      /* empty */
7  }
8
9  BSTree::~~BSTree()
10 {
11     this->clear();
12 }
13
14 bool BSTree::insert( int intInsertValue )
15 {
16     this->insert( intInsertValue, this->pObjBSTNodeRoot );
17 }
18
19 bool BSTree::remove( int intRemoveValue )
20 {
21     return remove( intRemoveValue, this->pObjBSTNodeRoot );
22 }
23
24 void BSTree::clear()
25 {
26     this->clear( this->pObjBSTNodeRoot );
27 }
28
29 unsigned int BSTree::getSize() const
30 {
31     return this->uintSize;
32 }
33
34 void BSTree::inOrder()
35 {
36     this->inOrder( this->pObjBSTNodeRoot );
37 }
38
39 bool BSTree::insert( int intInsertValue, BSTNode*& pObjBSTNodeInsertNode )
40 {
41     if( uintSize > 0 )
42     {
43         if( intInsertValue == pObjBSTNodeInsertNode->getContents() )
44         {
45             return false;
46         }
47         else
48         {
49             if( intInsertValue < pObjBSTNodeInsertNode->getContents() )
50             {

```

```

51         if( pObjBSTNodeInsertNode->getLeftChild() == NULL )
52         {
53             pObjBSTNodeInsertNode->setLeftChild( new BSTNode( intInsertValue ) );
54             this->uintSize++;
55             return true;
56         }
57         else
58         {
59             return this->insert( intInsertValue, pObjBSTNodeInsertNode->getLeftChild() );
60         }
61     }
62
63     if( intInsertValue > pObjBSTNodeInsertNode->getContents() )
64     {
65         if( pObjBSTNodeInsertNode->getRightChild() == NULL )
66         {
67             pObjBSTNodeInsertNode->setRightChild( new BSTNode( intInsertValue ) );
68             this->uintSize++;
69             return true;
70         }
71         else
72         {
73             return this->insert( intInsertValue, pObjBSTNodeInsertNode->getRightChild() );
74         }
75     }
76 }
77
78 else
79 {
80     pObjBSTNodeInsertNode = new BSTNode( intInsertValue );
81     this->uintSize++;
82     return true;
83 }
84 }
85
86 bool BSTree::remove( int intRemoveValue, BSTNode*& pObjBSTNodeSelectedNode )
87 {
88     if( pObjBSTNodeSelectedNode == NULL )
89     {
90         return false;
91     }
92     else
93     {
94         if( intRemoveValue == pObjBSTNodeSelectedNode->getContents() )
95         {
96             if( pObjBSTNodeSelectedNode->getLeftChild() == NULL )
97             {
98                 BSTNode* pObjBSTNodeMarkDelete = pObjBSTNodeSelectedNode;
99                 // command automatically works backwards to assign the new value of leftchild to the parent's
100                 pObjBSTNodeLeftChild pointer variable

```

```

100         pObjBSTNodeSelectedNode = pObjBSTNodeSelectedNode->getRightChild();
101         delete pObjBSTNodeMarkDelete;
102         pObjBSTNodeMarkDelete = NULL;
103         this->uintSize--;
104     }
105     else
106     {
107         this->removeMax( pObjBSTNodeSelectedNode->getContents(), pObjBSTNodeSelectedNode->getLeftChild() );
108     }
109
110     return true;
111 }
112 else
113 {
114     if( intRemoveValue < pObjBSTNodeSelectedNode->getContents() )
115     {
116         return this->remove( intRemoveValue, pObjBSTNodeSelectedNode->getLeftChild() );
117     }
118     else
119     {
120         return this->remove( intRemoveValue, pObjBSTNodeSelectedNode->getRightChild() );
121     }
122 }
123 }
124 }
125
126 void BSTree::removeMax( int& intRemoveValue, BSTNode*& pObjBSTNodeSelectedNode )
127 {
128     if( pObjBSTNodeSelectedNode->getRightChild() == NULL )
129     {
130         intRemoveValue = pObjBSTNodeSelectedNode->getContents();
131
132         BSTNode* pObjBSTNodeMarkDelete = pObjBSTNodeSelectedNode;
133         pObjBSTNodeSelectedNode = pObjBSTNodeSelectedNode->getRightChild();
134         delete pObjBSTNodeMarkDelete;
135         pObjBSTNodeMarkDelete = NULL;
136         this->uintSize--;
137     }
138     else
139     {
140         removeMax( intRemoveValue, pObjBSTNodeSelectedNode->getRightChild() );
141     }
142 }
143
144 void BSTree::clear( BSTNode*& pObjBSTNodeClearNode )
145 {
146     if( uintSize > 0 )
147     {
148         if( pObjBSTNodeClearNode->getLeftChild() != NULL )
149         {

```

```

150         this->clear( pObjBSTNodeClearNode->getLeftChild() );
151     }
152
153     if( pObjBSTNodeClearNode->getRightChild() != NULL )
154     {
155         this->clear( pObjBSTNodeClearNode->getRightChild() );
156     }
157
158     delete pObjBSTNodeClearNode;
159     pObjBSTNodeClearNode = NULL;
160     this->uintSize--;
161 }
162
163
164 void BSTree::inOrder( BSTNode* pObjBSTNodePrintNode )
165 {
166     if( uintSize > 0 )
167     {
168         if( pObjBSTNodePrintNode->getLeftChild() != NULL )
169         {
170             this->inOrder( pObjBSTNodePrintNode->getLeftChild() );
171         }
172
173         cout << pObjBSTNodePrintNode->getContents() << " ";
174
175         if( pObjBSTNodePrintNode->getRightChild() != NULL )
176         {
177             this->inOrder( pObjBSTNodePrintNode->getRightChild() );
178         }
179     }
180     else
181     {
182         cout << "";
183     }
184 }
185

```