```
#include "BSTree.h"

BSTree::BSTree()
: pobjBSTNodeRoot( NULL ), uintSize( 0 )
{
    /* empty */
}

BSTree::~BSTree()
{
    this->clear();
}

bool BSTree::insert( int intInsertValue )
{
    this->insert( intInsertValue, this->pobjBSTNodeRoot );
}

bool BSTree::remove( int intRemoveValue )
{
    return remove( intRemoveValue, this->pobjBSTNodeRoot );
}

void BSTree::clear()
{
    this->clear( this->pobjBSTNodeRoot );
}

unsigned int BSTree::getSize() const
{
    return this->uintSize;
}

void BSTree::inOrder()
{
    this->inOrder( this->pobjBSTNodeRoot );
}

bool BSTree::insert( int intInsertValue, BSTNode*& pobjBSTNodeInsertNode )
{
    if( uintSize > 0 )
    {
        if( intInsertValue == pobjBSTNodeInsertNode->getContents() )
        {
            return false;
        }
        else
        {
            if( intInsertValue < pobjBSTNodeInsertNode->getContents() )
            {
```

```
                if( pobjBSTNodeInsertNode->getLeftChild() == NULL )
                {
                    pobjBSTNodeInsertNode->setLeftChild( new BSTNode( intInsertValue ) );
                    this->uintSize++;
                    return true;
                }
                else
                {
                    return this->insert( intInsertValue, pobjBSTNodeInsertNode->getLeftChild() );
                }
            }

            if( intInsertValue > pobjBSTNodeInsertNode->getContents() )
            {
                if( pobjBSTNodeInsertNode->getRightChild() == NULL )
                {
                    pobjBSTNodeInsertNode->setRightChild( new BSTNode( intInsertValue ) );
                    this->uintSize++;
                    return true;
                }
                else
                {
                    return this->insert( intInsertValue, pobjBSTNodeInsertNode->getRightChild() );
                }
            }
        }
    }
    else
    {
        pobjBSTNodeInsertNode   = new BSTNode( intInsertValue );
        this->uintSize++;
        return true;
    }
}

bool BSTree::remove( int intRemoveValue, BSTNode*& pobjBSTNodeSelectedNode )
{
    if( pobjBSTNodeSelectedNode == NULL )
    {
        return false;
    }
    else
    {
        if( intRemoveValue == pobjBSTNodeSelectedNode->getContents() )
        {
            if( pobjBSTNodeSelectedNode->getLeftChild() == NULL )
            {
                BSTNode* pobjBSTNodeMarkDelete = pobjBSTNodeSelectedNode;
                // command automatically works backwards to assign the new value of leftchild to the parent's
                pobjBSTNodeLeftChild pointer variable
```

```
                pobjBSTNodeSelectedNode     = pobjBSTNodeSelectedNode->getRightChild();
                delete pobjBSTNodeMarkDelete;
                pobjBSTNodeMarkDelete               = NULL;
                this->uintSize--;
            }
            else
            {
                this->removeMax( pobjBSTNodeSelectedNode->getContents(), pobjBSTNodeSelectedNode->getLeftChild() );
            }

            return true;
        }
        else
        {
            if( intRemoveValue < pobjBSTNodeSelectedNode->getContents() )
            {
                return this->remove( intRemoveValue, pobjBSTNodeSelectedNode->getLeftChild() );
            }
            else
            {
                return this->remove( intRemoveValue, pobjBSTNodeSelectedNode->getRightChild() );
            }
        }
    }
}

void BSTree::removeMax( int& intRemoveValue, BSTNode*& pobjBSTNodeSelectedNode )
{
    if( pobjBSTNodeSelectedNode->getRightChild() == NULL )
    {
        intRemoveValue  = pobjBSTNodeSelectedNode->getContents();

        BSTNode* pobjBSTNodeMarkDelete  = pobjBSTNodeSelectedNode;
        pobjBSTNodeSelectedNode     = pobjBSTNodeSelectedNode->getRightChild();
        delete pobjBSTNodeMarkDelete;
        pobjBSTNodeMarkDelete               = NULL;
        this->uintSize--;
    }
    else
    {
        removeMax( intRemoveValue, pobjBSTNodeSelectedNode->getRightChild() );
    }
}

void BSTree::clear( BSTNode*& pobjBSTNodeClearNode )
{
    if( uintSize > 0 )
    {
        if( pobjBSTNodeClearNode->getLeftChild() != NULL )
        {
```

```
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
```