```
#include "BSTree.h"

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

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

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

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

        if( pobjBSTNodeClearNode->getRightChild() != NULL )
        {
            this->clear( pobjBSTNodeClearNode->getRightChild() );
        }

        delete pobjBSTNodeClearNode;
        pobjBSTNodeClearNode    = NULL;
        this->uintSize--;
    }
}
```

```
101  void BSTree::inOrder( BSTNode* pobjBSTNodePrintNode )
102  {
103      if( uintSize > 0 )
104      {
105          if( pobjBSTNodePrintNode->getLeftChild() != NULL )
106          {
107              this->inOrder( pobjBSTNodePrintNode->getLeftChild() );
108          }
109
110          cout << pobjBSTNodePrintNode->getContents() << " ";
111
112          if( pobjBSTNodePrintNode->getRightChild() != NULL )
113          {
114              this->inOrder( pobjBSTNodePrintNode->getRightChild() );
115          }
116      }
117      else
118      {
119          cout << "";
120      }
121  }
122
```