```
#include "Word.h"

Word::Word()
: uintCount( 1 )
{
    /* empty */
}

Word::Word( string strWord )
: strWord( strWord ), uintCount( 1 )
{
    /* empty */
}

Word::~Word()
{
    /* empty */
}

ostream& operator << ( ostream& objostreamOut, const Word*& objWordToOutput )
{
    unsigned int uintWordLength = objWordToOutput->strWord.length();

    for( unsigned int uintI; uintI < uintWordLength; uintI++ )
    {
        objostreamOut << toupper( objWordToOutput->strWord.at( uintI ) );
    }

    objostreamOut << " " << objWordToOutput->getCount();

    return objostreamOut;
}

bool Word::operator == ( Word& objWordCompare )
{
    unsigned int uintWordLength     = this->strWord.length();
    string strCompareValue          = objWordCompare.getValue();

    if( uintWordLength != strCompareValue.length() )
    {
        return false;
    }
    else
    {
        for( unsigned int uintI = 0; uintI < uintWordLength; uintI++)
        {
            if( toupper( this->strWord.at( uintI ) ) != toupper( strCompareValue.at( uintI ) ) )
            {
                return false;
            }
```

```
        }

        return true;
    }
}

bool Word::operator < ( Word& objWordCompare )
{
    unsigned int uintSmallestWordLength = 0;
    string strCompareValue          = objWordCompare.getWord();

    if( this->strWord.length() <= strCompareValue.length() )
    {
        uintSmallestWordLength  = this->strWord.length();
    }
    else
    {
        uintSmallestWordLength  = strCompareValue.length();
    }

    for( unsigned int uintI = 0; uintI < uintSmallestWordLength; uintI++ )
    {
        if( toupper( this->strWord.at( uintI ) ) < toupper( strCompareValue.at( uintI ) ) )
        {
            return true;
        }
        else if( toupper( this->strWord.at( uintI ) ) > toupper( strCompareValue.at( uintI ) ) )
        {
            return false;
        }
    }
}

bool Word::operator > ( Word& objWordCompare )
{
    unsigned int uintSmallestWordLength = 0;
    string strCompareValue          = objWordCompare.getWord();

    if( this->strWord.length() <= strCompareValue.length() )
    {
        uintSmallestWordLength  = this->strWord.length();
    }
    else
    {
        uintSmallestWordLength  = strCompareValue.length();
    }

    for( unsigned int uintI = 0; uintI < uintSmallestWordLength; uintI++ )
    {
        if( toupper( this->strWord.at( uintI ) ) > toupper( strCompareValue.at( uintI ) ) )
```

```
        {
            return true;
        }
        else if( toupper( this->strWord.at( uintI ) ) < toupper( strCompareValue.at( uintI ) ) )
        {
            return false;
        }
    }
}

string Word::getWord() const
{
    return this->strWord;
}

string Word::getValue() const
{
    return this->strWord;
}

void Word::setWord( string strWord )
{
    this->strWord   = strWord;
}

unsigned int Word::getCount() const
{
    return this->uintCount;
}

void Word::incrementCount()
{
    this->uintCount++;
}

```