```cpp
#include "JCString.h"
#include <iostream>


// initialzer with basic values
// creates a char array with 20 memory


JCString::JCString() {
    this->cap = 20; //size of memory
    this->end = 0;//index of the end of the string
    this->str = new char[cap]; // creates the an array of size 20 chars
    this->str[end] = '\0'; // terminates the char array
}

// construntor for dumping arrays
JCString::JCString(const char* cstr) {
    //while loop counts chars and stores int
    while (cstr[this->end] != '\0')
    {
        ++this->end;
    }
    // will count until the value right before '\0'
    // if char* has 3 elem then end will return 3


    this->cap = 20;             // max size for now
    this->str = new char[cap]; // creates char arr a holding array
    //fills a char array
    // stores in the variable
    for (int i = 0; i <= this->end; ++i) {
        this->str[i] = cstr[i];
    }


```

```cpp
}
int JCString::length() {
    return this->end;
}

int JCString::capacity() {
    return this->cap;
}

char JCString::at(int index) {
    if (index >= 0 && index < end) {
        return this->str[index];
    }
    else {
        return '\0';
    }
}
//for reading streams??
bool JCString::read(istream& inputStrm) {
    char inputWord[ 100 ];
    if (inputStrm >> inputWord) { // reads in the word with the extractor ">>"
        for (this->end = 0; inputWord[this->end] != '\0'; ++(this->end));           //empty loop

        // cap = ??;                        //TODO: needs to potentially grow for prog3

        for (int i = 0; i <= this->end; ++i) {
            this->str[i] = inputWord[i];
        }
        return true;
    }
    else
        return false;
}

void JCString::write(ostream& outputStrm) {
    outputStrm << this->str;
}

bool JCString::lessThan(const JCString& argStr) {
    if (this->JCScompareTo(argStr) == -1)
    {
        return true;
    }

    return false;
}

bool JCString::greaterThan(const JCString& argStr) {
    if (this->JCScompareTo(argStr) == 1)
    {
        return true;
    }

    return false;
}

bool JCString::equals(const JCString& argStr) {
    if (this->JCScompareTo(argStr) == 0)
    {
        return true;
    }
    return false;
}

int JCString::JCScompareTo(const JCString& angStr)
{

        int len = 0;
        int count = 0;
        int result = 0;
        // dummie char stirngs
        JCString thisString(this->str);

        // lower case things for when we implement that later
        // uncomment when ready
        //JCString str2 = angStr.returnLower();
        //str1.makeLower();

        JCString str2(angStr.str);
        // make sure we iter through to the shortest char string
        if (thisString.length() < str2.length())
        {
            len = thisString.length();
        }
        else
        {
            len = str2.length();
        }
        // compares char for char, returns 1 if this-> string is larger

        while (count < len)
        {
            // if the char arr is shorter than the other but equal otherwise
            // the shorter one wins by defult
            // makes sure we don't go out of bounds
            if(thisString.str[count] > str2.str[count])
            {
                //str1 is greater or comes later return 1
                result = 1;
                count = len; // effectivly breaks

            }
            else if (thisString.str[count] < str2.str[count])
            {
                result = -1;
```

```
                    //str2 is greater or comes later return 1
                    count = len; // effectivly breaks
                }//then they must be the same char , if one is terminated here it comes first (is smaller)
                else if (thisString.str[count+1] == '\0')
                {
                    result = -1;//compare string is larger comes later in alpha
                    count = len;

                }
                else if (str2.str[count+1] == '\0')
                {
                    result = 1; //this string is larger, compare string comes first
                    count = len;
                }// if not done continue comparing char for char

                count++;//move to the next char
            }
            return result;//return 0 if equal
    }

    void JCString::setEqualTo(const JCString& argStr) {
        this->end = argStr.end;
        this->cap= argStr.cap;
                            //TODO: needs to potentially grow for prog3

        for (int i = 0; i <= end; ++i) {
            this->str[i] = argStr.str[i];
        }
    }
    const char* JCString::c_str() {
        return this->str;
    }

    //  modifies the JCString such that it is all lower case
    void JCString::makeLower()
```

```cpp
{
    for (int i = 0; i <= this->end; i++)
    {
        if (this->str[i] < 91 || this->str[i] > 64)
        {
            this->str[i] += 32;
        }
    }
}
// returns an instance of the JCString that is lower case
JCString JCString::returnLower() const
{
    JCString returnString(this->str);
    returnString.makeLower();

    return returnString;
}

```