```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) {
    //end starts at zero always
    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];
    }


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

```
36  }
37
38  int JCString::capacity() {
39      return this->cap;
40  }
41
42  char JCString::at(int index) {
43      if (index >= 0 && index < end) {
44          return this->str[index];
45      }
46      else {
47          return '\0';
48      }
49  }
50  //for reading streams??
51  bool JCString::read(istream& inputStrm) {
52      char inputWord[ 100 ];
53      if (inputStrm >> inputWord) {
54          for (this->end = 0; inputWord[this->end] != '\0'; ++(this->end));           //empty loop
55
56          // cap = ??;                          //TODO: needs to potentially grow for prog3
57
58          for (int i = 0; i <= this->end; ++i) {
59              this->str[i] = inputWord[i];
60          }
61          return true;
62      }
63      else
64          return false;
65  }
66
67  void JCString::write(ostream& outputStrm) {
68      outputStrm << this->str;
69  }
70
```

```
 71  bool JCString::lessThan(const JCString& argStr) {
 72      if (this->JCScompareTo(argStr) == -1)
 73      {
 74          return true;
 75      }
 76
 77      return false;
 78  }
 79
 80  bool JCString::greaterThan(const JCString& argStr) {
 81      if (this->JCScompareTo(argStr) == 1)
 82      {
 83          return true;
 84      }
 85
 86      return false;
 87  }
 88
 89  bool JCString::equals(const JCString& argStr) {
 90      if (this->JCScompareTo(argStr) == 0)
 91      {
 92          return true;
 93      }
 94      return false;
 95  }
 96
 97  int JCString::JCScompareTo(const JCString& angStr)
 98  {
 99
100          int len = 0;
101          int count = 0;
102          int result = 0;
103          // dumie char stirngs
104          JCString str1(this->str);
105
```

```
106         // lower case things
107         str1.makeLower();
108         JCString str2 = angStr.returnLower();
109
110
111
112
113         // make sure we iter through to the longest char string
114         if (str1.length() > str2.length())
115         {
116             len = str1.length();
117         }
118         else
119         {
120             len = str2.length();
121         }
122         // compares char for char, returns 1 if this-> string is larger
123
124         while (count < len)
125         {
126             if (str1.str[count] > str2.str[count])
127             {
128                 result = 1;
129                 count = len; // effectivly breaks
130             }
131             else if (str1.str[count] < str2.str[count])
132             {
133                 result = -1;
134                 count = len; // effectivly breaks
135             }
136             count++;
137         }
138         return result;//return 0 if equal
139 }
140
```

```
141 void JCString::setEqualTo(const JCString& argStr) {
142     this->end = argStr.end;
143     this->cap= argStr.cap;
144                         //TODO: needs to potentially grow for prog3
145 
146     for (int i = 0; i <= end; ++i) {
147         this->str[i] = argStr.str[i];
148     }
149 }
150 void JCString::print()
151 {
152     for (int index = 0; index < this->end; index++)
153     {
154         std::cout << this->str[index];
155     }
156 }
157 const char* JCString::c_str() {
158     return this->str;
159 }
160 
161 void JCString::makeLower()
162 {
163     for (int i = 0; i <= this->end; i++)
164     {
165         if (this->str[i] < 91 || this->str[i] > 64)
166         {
167             this->str[i] += 32;
168         }
169     }
170 }
171 JCString JCString::returnLower() const
172 {
173     JCString returnString(this->str);
174     returnString.makeLower();
175 
```

```
176     return returnString;
177 }
178
```