```cpp
/*
Filename: p3.cpp
Author(s): Zachary Rea and Parker Ross
Date: 5 February 2023
Description: The cpp for circular lists
*/
#include <iostream>  //allows for usage of cin, cout, and cerr

#include "p3.h"

using namespace std;

//*************************************************************************
//Constructors and De-constructors
//*************************************************************************
//Constructor
//Written by Zach

cStringList::cStringList(int capacity) {
    this->listCapacity = listCapacity;
    a = new string[listCapacity];
    listCount = 0;
    first = 0;
    last = 0;
}

//*************************************************************************
//De-constructor
//Written by Zach

cStringList::~cStringList() {
    delete [] a;
}

//*************************************************************************
//Private functions
//*************************************************************************
//Function for decrementing passed value
//Written by Parker

void cStringList::decVal(int &value){
    //prevents decVal from going beyond list capacity
    if (value > 0) {
        value = value - 1;
    } else {
        value = listCapacity - 1;
    }
}

//*************************************************************************
//Function for Incrementing passed value
//Written by Parker

void cStringList::incVal(int &value){
    //prevents incVal from going beyond list capacity
    if (value < listCapacity) {
        value = value + 1;
    } else {
        value = 0;
    }
}

//*************************************************************************
//Public functions
//*************************************************************************
//Function for inserting a string at first position
//Written by Zach

bool cStringList::insert(string text) {
```

```
70      bool rc = false;
71      //Ensure there is space
72      if (listCount < listCapacity) {
73          if (listCount == 0) {
74              last = first;
75          } else {
76              //move the first place
77              decVal(first);
78          }
79          a[first] = text;
80          listCount++;
81          rc = true;
82      }
83      return rc;
84  }
85
86  //*************************************************************************
87  //Function for adding a string to the last position
88  //Written by Zach
89
90  bool cStringList::add(string text) {
91      bool rc = false;
92      if (listCount < listCapacity) {
93          //move the last place
94          incVal(last);
95          a[last] = text;
96          if (listCount == 0) {
97              first = last;
98          }
99          listCount++;
100         rc = true;
101     }
102     return rc;
103 }
104
105 //*************************************************************************
106 //Function for inserting a string at the given index
107 //Written by Zach
108
109 bool cStringList::insertAt(int index, string text) {
110     bool rc = false;
111     //Check for valid index
112     if ((index < listCount) && (listCount < listCapacity) && (index >= 0)) {
113         incVal(last);
114         int temp = last;
115         //walk through the list
116         for (int i = listCount; i > index; i--) {
117             int move = temp;
118             decVal (move);
119             //copy the contents of the list down
120             a[temp] = a[move];
121             decVal(temp);
122         }
123         a[index] = text;
124         listCount++;
125         rc = true;
126     }
127     return rc;
128 }
129
130 //*************************************************************************
131 //Funtion for deleting the contents of a given index
132 //Written by Zach
133
134 bool cStringList::deleteAt(int index, string &text) {
135     bool rc = false;
136     //check for valid index
137     if ((index >= 0) && (index < listCount) && (listCount > 0)) {
138         int i = first;
```

```
139         //walk through the list
140         while (i <= index) {
141             incVal(i);
142         }
143         //set the text to the string that is being deleted
144         text = a[i];
145         //overwrite the rest of the contents
146         while (i != last) {
147             int temp = i;
148             incVal(i);
149             a[temp] = a[i];
150         }
151         decVal(last);
152         listCount--;
153         rc = true;
154     }
155     return rc;
156 }
157 
158 //*****************************************************************************
159 //Function for reading the value at given index
160 //written by Parker
161 
162 bool cStringList::readAt(int index, string &text) {
163     bool rc = false;
164     //checking for index within the list
165     if ((index >= 0) && (index < listCount)) {
166         int read = first;
167         int i = 0;
168         //shifts entries to the left of index
169         while (i < index) {
170             i++;
171             incVal(read);
172         }
173         text = a[read];
174         rc = true;
175     }
176     return rc;
177 }
178 
179 //*****************************************************************************
180 //Function for deleting the first entry
181 //Written by Zach
182 
183 bool cStringList::deleteFirst(string &text) {
184     bool rc = false;
185     if (listCount > 0) {
186         //copy the text that is being removed
187         text = a[first];
188         //move the first indicator
189         incVal(first);
190         listCount--;
191         rc = true;
192     }
193     return rc;
194 }
195 
196 //*****************************************************************************
197 //Function for deleting the last entry
198 //Written by Zach
199 
200 bool cStringList::deleteLast(string &text) {
201     bool rc = false;
202     if (listCount > 0) {
203         //copy the text that is being deleted
204         text = a[last];
205         //move the last indicator
206         decVal(last);
207         listCount--;
```

```
208             rc = true;
209         }
210         return rc;
211     }
212
213     //*************************************************************************
214     //Fucntion for clearing list
215     //written by Parker
216
217     void cStringList::clear() {
218         //empties list
219         listCount = 0;
220         first = last = 0;
221     }
222
223     //*************************************************************************
224     //function for returning size of list
225     //written by Parker
226
227     int cStringList::count() const{
228         //returns number of list entries
229         return listCount;
230     }
231
232     //*************************************************************************
233     //Function for returning the index of the given string
234     //Written by Zach
235
236     int cStringList::getIndex(string text) {
237         int rc = -1;
238         int i = first;
239         //search for the text in a loop til last
240         while ((a[i] != text) && (i != last)) {
241             incVal(i);
242         }
243         //if found, record the index it was found
244         if (a[i] == text) {
245             rc = i;
246         }
247         return rc;
248     }
249
250     //*************************************************************************
251     //Function for printing values in circular list
252     //Written by Parker
253
254     void cStringList::printIt() {
255         if (listCount) {
256             int print = first;
257             int temp = last;
258             incVal(temp);
259             int i = 0;
260             while (print != temp){
261                 cout << "At pos " << i << " there is " << a[print] << "\n";
262                 incVal(print);
263                 i++;
264             }
265         }
266     }
267
268     /*Professor, in the assignment, you listed that the printIt is intended
269     to be a const function, but I dont understand a way of writing that such
270     that it would not need to involve the incVal function. Given the fact
271     that we can't simply walk through the list. As this was not a part of
272     p3a, I wrote it like this for now, and will be following up with you
273     tomorrow. This just serves to acknowledge that it is different from the
274     assignment page.
275     */
```