```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 - 1) {
        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              decVal(first);
75          }
76          a[first] = text;
77          listCount++;
78          rc = true;
79      }
80      return rc;
81  }
82
83  //*************************************************************************
84  //Function for adding a string to the last position
85  //Written by Zach
86
87  bool cStringList::add(string text) {
88      bool rc = false;
89      if (listCount < listCapacity) {
90          //move the last place
91          incVal(last);
92          a[last] = text;
93          if (listCount == 0) {
94              first = last;
95          }
96          listCount++;
97          rc = true;
98      }
99      return rc;
100 }
101
102 //*************************************************************************
103 //Function for inserting a string at the given index
104 //Written by Zach
105
106 bool cStringList::insertAt(int index, string text) {
107     bool rc = false;
108     //Check for valid index
109     if ((index <= listCount) && (index >= 0)) {
110         incVal(last);
111         int temp = last;
112         //walk through the list
113         for (int i = listCount; i > index; i--) {
114             int move = temp;
115             decVal (move);
116             //copy the contents of the list down
117             a[temp] = a[move];
118             decVal(temp);
119         }
120         a[temp] = text;
121         listCount++;
122         rc = true;
123     }
124     return rc;
125 }
126
127 //*************************************************************************
128 //Funtion for deleting the contents of a given index
129 //Written by Zach
130
131 bool cStringList::deleteAt(int index, string &text) {
132     bool rc = false;
133     //check for valid index
134     if ((index >= 0) && (index < listCount)) {
135         int i = first;
136         //walk through the list strarting from first
137         while (i <= index) {
138             //use incVal to ensure proper wrapping
```

```
139             //this also allows us to use the index value
140             incVal(i);
141         }
142         //set the text to the string that is being deleted
143         text = a[i];
144         //overwrite the rest of the contents
145         while (i != last) {
146             int temp = i;
147             incVal(i);
148             a[temp] = a[i];
149         }
150         decVal(last);
151         listCount--;
152         rc = true;
153     }
154     return rc;
155 }
156 
157 //*************************************************************************
158 //Function for reading the value at given index
159 //written by Parker
160 
161 bool cStringList::readAt(int index, string &text) {
162     bool rc = false;
163     //checking for index within the list
164     if ((index >= 0) && (index < listCount)) {
165         int read = first;
166         int i = 0;
167         //shifts entries to the left of index
168         while (i < index) {
169             i++;
170             //use incVal to allow proper wrapping
171             //this ensures that index is correct
172             incVal(read);
173         }
174         text = a[read];
175         rc = true;
176     }
177     return rc;
178 }
179 
180 //*************************************************************************
181 //Function for deleting the first entry
182 //Written by Zach
183 
184 bool cStringList::deleteFirst(string &text) {
185     bool rc = false;
186     if (listCount > 0) {
187         //copy the text that is being removed
188         text = a[first];
189         //move the first indicator
190         incVal(first);
191         if (listCount == 1) {
192             last = first = 0;
193         }
194         listCount--;
195         rc = true;
196     }
197     return rc;
198 }
199 
200 //*************************************************************************
201 //Function for deleting the last entry
202 //Written by Zach
203 
204 bool cStringList::deleteLast(string &text) {
205     bool rc = false;
206     if (listCount > 0) {
207         //copy the text that is being deleted
```

```
208             text = a[last];
209             //move the last indicator
210             decVal(last);
211             if (listCount == 1) {
212                 last = first = 0;
213             }
214             listCount--;
215             rc = true;
216         }
217         return rc;
218     }
219
220     //*************************************************************************
221     //Fucntion for clearing list
222     //written by Parker
223
224     void cStringList::clear() {
225         //empties list
226         listCount = 0;
227         first = last = 0;
228     }
229
230     //*************************************************************************
231     //function for returning size of list
232     //written by Parker
233
234     int cStringList::count() const{
235         //returns number of list entries
236         return listCount;
237     }
238
239     //*************************************************************************
240     //Function for returning the index of the given string
241     //Written by Zach
242
243     int cStringList::getIndex(string text) {
244         int rc = -1;
245         int i = first;
246         //search for the text in a loop til last
247         while ((a[i] != text) && (i != last)) {
248             incVal(i);
249         }
250         //if found, record the index it was found
251         if (a[i] == text) {
252             rc = i;
253         }
254         return rc;
255     }
256
257     //*************************************************************************
258     //Function for printing values in circular list
259     //Written by Parker
260
261     void cStringList::printIt() const{
262         if (listCount) {
263             int print = first;
264             int temp = last;
265             if (temp < listCount) {
266                 temp++;
267             } else {
268                 temp = 0;
269             }
270             int i = 0;
271             while (print != temp){
272                 cout << "At pos " << i << " there is " << a[print] << "\n";
273                 if (print < listCount) {
274                     print++;
275                 } else {
276                     print = 0;
```

```
277                 }
278                 i++;
279             }
280         }
281     }
```