

```
1  /*
2  ALEX FRIEDBERG
3
4  LINKED LIST .cpp
5  */
6
7  #include "stdafx.h"
8  #include <iostream>
9  #include <fstream>
10 #include <string>
11 #include <sstream>
12 #include "linklist.h"
13
14 using namespace std;
15
16 linklist::linklist() {
17     head = NULL;
18 }
19
20 /*
21 void linklist::removeName(string name){
22
23     LinkListEntry *curr;
24     LinkListEntry *prev;
25     prev = NULL;
26
27     for(curr = head; curr != NULL; curr = curr->next){
28         if(curr->name.compare(name) == 0){
29             if(prev == NULL){
30                 //First Entry in list
31                 head = curr->next;
32                 delete curr;
33             }else{
34                 //more than one entry in list
35                 prev->next = curr->next;
36                 delete curr;
37             }
38             return;
39         }
40         prev = curr;
41     }
42     cout << "ERROR: '" << name << "' not found in list" << endl;
43
44 }
45 */
46
47 LINK_LIST_ENTRY_TYPE linklist::removeAtBegin() {
48     LinkListEntry *prev;
49     LinkListEntry *curr;
```

```
50     LINK_LIST_ENTRY_TYPE returnValue;
51     prev = NULL;
52
53     for (curr = head; curr != NULL; curr = curr->next) {
54         if (curr->next == NULL) {
55             //Last Entry
56             if (prev == NULL) {
57                 //First and only entry in list
58                 head = NULL;
59             }
60             else {
61                 prev->next = NULL;
62             }
63             returnValue = curr->data;
64             delete curr;
65             break;
66         }
67         prev = curr;
68     }
69
70     return returnValue;
71 }
72
73
74 LINK_LIST_ENTRY_TYPE linklist::removeAtEnd() {
75     LinkListEntry *prev;
76     LinkListEntry *curr;
77     LINK_LIST_ENTRY_TYPE returnValue;
78
79     if (head != NULL) {
80         curr = head;
81         head = head->next;
82
83         returnValue = curr->data;
84         delete curr;
85     }
86     return returnValue;
87 }
88
89
90 void linklist::addAtBegin(LINK_LIST_ENTRY_TYPE data) {
91
92     LinkListEntry *n = new LinkListEntry;
93
94     LinkListEntry *curr;
95     LinkListEntry *prev;
96     prev = NULL;
97
98     for (curr = head; curr != NULL; curr = curr->next) {
```

```
99     prev = curr;
100 }
101
102 // LinkListEntry *n = new linkListEntry;
103
104 n->data = data;
105 n->next = NULL;
106 if (head != NULL) {
107     prev->next = n;
108 }
109 else {
110     head = n;
111 }
112 }
113
114 void linklist::addAtEnd(LINK_LIST_ENTRY_TYPE data) {
115
116     LinkListEntry *n = new LinkListEntry; //NEW NODE
117
118     n->data = data;
119
120     n->next = head;
121     head = n;
122
123 }
124
125
126 void linklist::printList() {
127     LinkListEntry *curr;
128
129     for (curr = head; curr != NULL; curr = curr->next) {
130
131         cout << curr->data;
132         if (curr->next != NULL){
133             cout << endl;
134         }
135
136     }
137     cout << endl;
138
139
140 }
141
142 //Returns data at the beginning of the queue
143 LINK_LIST_ENTRY_TYPE linklist::peekAtBegin() {
144
145     LinkListEntry *curr;
146     LinkListEntry *prev;
147     prev = NULL;
```

```
148     //person dummyPerson;
149
150     for (curr = head; curr != NULL; curr = curr->next) {
151         prev = curr;
152     }
153
154     if (prev != NULL) {
155         return prev->data;
156     }
157
158     return NULL;
159
160 }
161
162 LINK_LIST_ENTRY_TYPE linklist::peekAtEnd() {
163
164     //person dummyPerson;
165
166     if (head != NULL) {
167         return head->data;
168     }
169
170     return NULL;
171
172 }
173
174 //Returns data value at the index of the list
175 LINK_LIST_ENTRY_TYPE linklist::getData(int index) {
176     LinkListEntry *curr;
177     int currIndex = 0;
178     //person dummyPerson;
179
180     for (curr = head; curr != NULL; curr = curr->next) {
181         if (currIndex == index) {
182             return curr->data;
183         }
184         currIndex++;
185     }
186     return NULL;
187 }
188
189 //Get number of entries in linklist
190 int linklist::countList() {
191     LinkListEntry *curr;
192     int currIndex = 0;
193
194     for (curr = head; curr != NULL; curr = curr->next) {
195         currIndex++;
196     }
```

```
197     return currIndex;
198 }
199 /*END OF FILE*/
```