```
1 //============================================================================
2 // Name        : LinkedList.cpp
3 // Author      : Kyle Voight
4 // Version     : 1.0
5 // Copyright   : Copyright © 2017 SNHU COCE
6 // Description : Lab 3-3 Lists and Searching
7 //============================================================================
8
9 #include <algorithm>
10 #include <iostream>
11 #include <time.h>
12
13 #include "CSVparser.hpp"
14
15 using namespace std;
16
17 //============================================================================
18 // Global definitions visible to all methods and classes
19 //============================================================================
20
21 // forward declarations
22 double strToDouble(string str, char ch);
23
24 // define a structure to hold bid information
25 struct Bid {
26     string bidId; // unique identifier
27     string title;
28     string fund;
29     double amount;
30     Bid() {
31         amount = 0.0;
32     }
33 };
34
35 //============================================================================
36 // Linked-List class definition
37 //============================================================================
38
39 /**
40  * Define a class containing data members and methods to
41  * implement a linked-list.
42  */
43 class LinkedList {
44
45 private:
46
47     Bid data;
48     node*nextNode;
49 };
50 node*head;
51 node*tail;
52 int size;
53
54 public:
55     LinkedList();
56     virtual ~LinkedList();
57     void Append(Bid bid);
```

```
58      void Prepend(Bid bid);
59      void PrintList();
60      void Remove(string bidId);
61      Bid Search(string bidId);
62      int Size();
63  };
64
65  /**
66   * Default constructor
67   */
68  LinkedList::LinkedList() {
69      head = nullptr;
70      tail = nullptr;
71      size = 0;
72  }
73
74  /**
75   * Destructor
76   */
77  LinkedList::~LinkedList() {
78  }
79
80  /**
81   * Append a new bid to the end of the list
82   */
83  void LinkedList::Append(Bid bid) {
84      node*newNode = new node;
85      newNode->data=bid;
86      newNode-> = nullptr;
87
88      if(head == nullptr){
89          head = newNode;
90          tail = newNode;
91          ++size;
92      }
93  }
94
95  /**
96   * Prepend a new bid to the start of the list
97   */
98  void LinkedList::Prepend(Bid bid) {
99      node*newNode = newnode;
100     newNode->data=bid;
101     newNode->nextNode=nullptr;
102
103     if(head==nullptr){
104         head=newNode;
105         tail=newNode;
106         ++size;
107     }
108     else{
109         newNode->nextNode=head;
110         head=newNode;
111         ++size;
112     }
113 }
114
```

```
115 /**
116  * Simple output of all bids in the list
117  */
118 void LinkedList::PrintList() {
119     node*temp=head;
120 
121     while(temp != nullptr){
122         cout<<temp->data.title<<"|";
123         cout<<temp->data.amount<<"|";
124         cout<<temp->data.fund<<endl;
125         temp=temp->nextNode;
126     }
127 }
128 
129 /**
130  * Remove a specified bid
131  *
132  * @param bidId The bid id to remove from the list
133  */
134 void LinkedList::Remove(string bidId) {
135     node* temp = head;
136     node* prevNode;
137     if(temp == nullptr){
138         return;
139        }
140     else if(head->data.bidId == bidId && head->nextNode == nullptr){
141         head = nullptr;
142         tail = nullptr;
143         delete temp;
144        }
145        else {
146            while(temp->data.bidId != bidId){
147                prevNode = temp;
148                temp = temp->nextNode;
149            }
150            prevNode->nextNode = temp->nextNode;
151            delete temp;
152        }
153 }
154 
155 /**
156  * Search for the specified bidId
157  *
158  * @param bidId The bid id to search for
159  */
160 Bid LinkedList::Search(string bidId) {
161     node* temp = head;
162     node* holder = new node;
163     holder->data.bidId = "";
164     while(temp != nullptr){
165         cout << temp->data.bidId << endl;
166         if(temp->data.bidId == bidId){
167             return temp->data;
168            }
169            cout <<"test";
170            temp = temp->nextNode;
171        }
```

```
172         return holder->data;
173 }
174
175 /**
176  * Returns the current size (number of elements) in the list
177  */
178 int LinkedList::Size() {
179     return size;
180 }
181
182 //============================================================================
183 // Static methods used for testing
184 //============================================================================
185
186 /**
187  * Display the bid information
188  *
189  * @param bid struct containing the bid info
190  */
191 void displayBid(Bid bid) {
192     cout << bid.bidId << ": " << bid.title << " | " << bid.amount
193          << " | " << bid.fund << endl;
194     return;
195 }
196
197 /**
198  * Prompt user for bid information
199  *
200  * @return Bid struct containing the bid info
201  */
202 Bid getBid() {
203     Bid bid;
204
205     cout << "Enter Id: ";
206     cin.ignore();
207     getline(cin, bid.bidId);
208
209     cout << "Enter title: ";
210     getline(cin, bid.title);
211
212     cout << "Enter fund: ";
213     cin >> bid.fund;
214
215     cout << "Enter amount: ";
216     cin.ignore();
217     string strAmount;
218     getline(cin, strAmount);
219     bid.amount = strToDouble(strAmount, '$');
220
221     return bid;
222 }
223
224 /**
225  * Load a CSV file containing bids into a LinkedList
226  *
227  * @return a LinkedList containing all the bids read
228  */
```

```
229 void loadBids(string csvPath, LinkedList *list) {
230     cout << "Loading CSV file " << csvPath << endl;
231 
232     // initialize the CSV Parser
233     csv::Parser file = csv::Parser(csvPath);
234 
235     try {
236         // loop to read rows of a CSV file
237         for (int i = 0; i < file.rowCount(); i++) {
238 
239             // initialize a bid using data from current row (i)
240             Bid bid;
241             bid.bidId = file[i][1];
242             bid.title = file[i][0];
243             bid.fund = file[i][8];
244             bid.amount = strToDouble(file[i][4], '$');
245 
246             //cout << bid.bidId << ": " << bid.title << " | " << bid.fund << " | " <<
    bid.amount << endl;
247 
248             // add this bid to the end
249             list->Append(bid);
250         }
251     } catch (csv::Error &e) {
252         std::cerr << e.what() << std::endl;
253     }
254 }
255 
256 /**
257  * Simple C function to convert a string to a double
258  * after stripping out unwanted char
259  *
260  * credit: http://stackoverflow.com/a/24875936
261  *
262  * @param ch The character to strip out
263  */
264 double strToDouble(string str, char ch) {
265     str.erase(remove(str.begin(), str.end(), ch), str.end());
266     return atof(str.c_str());
267 }
268 
269 /**
270  * The one and only main() method
271  *
272  * @param arg[1] path to CSV file to load from (optional)
273  * @param arg[2] the bid Id to use when searching the list (optional)
274  */
275 int main(int argc, char* argv[]) {
276 
277     // process command line arguments
278     string csvPath, bidKey;
279     switch (argc) {
280     case 2:
281         csvPath = argv[1];
282         bidKey = "98109";
283         break;
284     case 3:
```

```
285         csvPath = argv[1];
286         bidKey = argv[2];
287         break;
288     default:
289         csvPath = "eBid_Monthly_Sales_Dec_2016.csv";
290         bidKey = "98109";
291     }
292
293     clock_t ticks;
294
295     LinkedList bidList;
296
297     Bid bid;
298
299     int choice = 0;
300     while (choice != 9) {
301         cout << "Menu:" << endl;
302         cout << "  1. Enter a Bid" << endl;
303         cout << "  2. Load Bids" << endl;
304         cout << "  3. Display All Bids" << endl;
305         cout << "  4. Find Bid" << endl;
306         cout << "  5. Remove Bid" << endl;
307         cout << "  9. Exit" << endl;
308         cout << "Enter choice: ";
309         cin >> choice;
310
311         switch (choice) {
312         case 1:
313             bid = getBid();
314             bidList.Append(bid);
315             displayBid(bid);
316
317             break;
318
319         case 2:
320             ticks = clock();
321
322             loadBids(csvPath, &bidList);
323
324             cout << bidList.Size() << " bids read" << endl;
325
326             ticks = clock() - ticks; // current clock ticks minus starting clock ticks
327             cout << "time: " << ticks << " milliseconds" << endl;
328             cout << "time: " << ticks * 1.0 / CLOCKS_PER_SEC << " seconds" << endl;
329
330             break;
331
332         case 3:
333             bidList.PrintList();
334
335             break;
336
337         case 4:
338             ticks = clock();
339
340             bid = bidList.Search(bidKey);
341
```

```
342             ticks = clock() - ticks; // current clock ticks minus starting clock ticks
343
344             if (!bid.bidId.empty()) {
345                 displayBid(bid);
346             } else {
347                 cout << "Bid Id " << bidKey << " not found." << endl;
348             }
349
350             cout << "time: " << ticks << " clock ticks" << endl;
351             cout << "time: " << ticks * 1.0 / CLOCKS_PER_SEC << " seconds" << endl;
352
353             break;
354
355         case 5:
356             bidList.Remove(bidKey);
357
358             break;
359         }
360     }
361
362     cout << "Good bye." << endl;
363
364     return 0;
365 }
366
```