```cpp
#include <iostream>
#include <fstream>
#include <string>
#include <iomanip>
#include "concordance.h"
using namespace std;

struct Node {
    string word;
    Node* next;
};

struct AZNode {
    AZNode* next;
    Node* branch;
    char letter;
};

concordance::concordance() {
    top = bottom = nullptr;
    for (char lttr = 'a'; lttr <= 'z'; lttr++) {
        AZNode* p = new AZNode;
        p->letter = lttr;
        p->branch = nullptr;
        if (top == nullptr && bottom == nullptr) {
            top = bottom = p;
            p->next = nullptr;
        }
        else {
            p->next = nullptr;
            bottom->next = p;
            bottom = p;
        }
    }
}//constructor

concordance::concordance(string file) {
    concordance();
    createFrom(file);
}//overloaded constructor

void concordance::createFrom(string file) {
    ifstream theText;
    theText.open(file);
    if (theText) {
        string word;
        while (theText >> word) {
            // this block takes the word and removes non-letter characters from
               the front and back
```

```
49              // then makes all letters lowercase in the word.
50              while (true) {
51                  if (word.size() == 0) break;
52                  char b = word.back();
53                  char f = word.front();
54                  if (b < 'A' || (b > 'Z' && b < 'a') || b > 'z') {
55                      word.pop_back();
56                      continue;
57                  }
58                  if (f < 'A' || (f > 'Z' && f < 'a') || f > 'z') {
59                      word.erase(0, 1);
60                      continue;
61                  }
62                  break;
63              }// remove bad end characters
64              for (unsigned i = 0; i < word.length(); i++) { //I got an error here fixed by unsigned instead of int, why?
65                  if (word[i] < 'A' || (word[i] > 'Z' && word[i] < 'a') || word[i] > 'z') continue;
66                  if (word[i] >= 'A' && word[i] <= 'Z') word[i] += 32;
67              }// make lowercase unless it is a dash or apostrophy or mid-word character (e.g. can't)
68
69              if (word.size() == 0) continue; // this in case of string with all odd characters proccessed
70
71              //put the word into the concordance
72              AZNode* runner = top;
73              int flag = 0;
74              do {
75                  if (runner->letter == word[0]) {
76                      Node* runnerTwo = runner->branch;
77                      Node* prevPointer = runnerTwo;
78                      int flagTwo = 0;
79                      do {
80                          if (runnerTwo == nullptr) {
81                              Node* toInsert = new Node;
82                              toInsert->word = word;
83                              toInsert->next = nullptr;
84                              runner->branch = toInsert;
85                              break;// flagTwo = 1;
86                          }
87                          else if (runnerTwo->word == word) {
88                              break;// flagTwo = 1;
89                          }
90                          else if (runnerTwo->next == nullptr) {
91                              Node* toInsert = new Node;
92                              toInsert->word = word;
93                              if (word > runnerTwo->word) {
```

```
 94                                  toInsert->next = nullptr;
 95                                  runnerTwo->next = toInsert;
 96                              }
 97                              else {
 98                                  if (prevPointer == runner->branch) {
 99                                      toInsert->next = runner->branch;
100                                      runner->branch = toInsert;
101                                  }
102                                  else {
103                                      toInsert->next = prevPointer->next;
104                                      prevPointer->next = toInsert;
105                                  }
106                              }
107                              break;// flagTwo = 1;
108                          }
109                          else if (runnerTwo == runner->branch && word <
                         runnerTwo->word) {
110                              Node* toInsert = new Node;
111                              toInsert->word = word;
112                              toInsert->next = runner->branch;
113                              runner->branch = toInsert;
114                              break;// flagTwo = 1;
115                          }
116                          else if ((runnerTwo->word < word) && (runnerTwo->next-
                         >word > word)) {
117                              Node* toInsert = new Node;
118                              toInsert->word = word;
119                              toInsert->next = runnerTwo->next;
120                              runnerTwo->next = toInsert;
121                              break;// flagTwo = 1;
122                          }
123                          else {
124                              prevPointer = runnerTwo;
125                              runnerTwo = runnerTwo->next;
126                          }
127                      } while (true);//flagTwo == 0);
128                      flag = 1;
129                  }
130                  else runner = runner->next;
131              } while (flag == 0);
132          }
133      }
134
135      theText.close();
136  }//createFrom
137
138  bool concordance::existanceOf(string word) {
139      AZNode* runner = top;
140      int flag = 0;
```

```
141     while(true) {
142         if (runner->letter == word[0]) {
143             Node* runnerTwo = runner->branch;
144             while (true) {
145                 if (runnerTwo == nullptr) {
146                     return false;
147                 }
148                 else if (runnerTwo->word == word) {
149                     return true;
150                 }
151                 else if (runnerTwo->word > word) {
152                     return false;
153                 }
154                 else {
155                     runnerTwo = runnerTwo->next;
156                 }
157             } while (true);
158         }
159         else {
160             runner = runner->next;
161             if (runner == nullptr) {
162                 break;
163             }
164         }
165     }
166     return false;
167 }//existanceOf
168 
169 void concordance::display(char letter) {
170     AZNode* runner = top;
171     cout << left;
172     do {
173         if (runner->letter == letter) {
174             cout << char(runner->letter - 32) << ":" << endl;
175             Node* runnerTwo = runner->branch;
176             int space = 0;
177             while (runnerTwo != nullptr) {
178                 if (space % 3 < 2) {
179                     cout << left << setw(15) << runnerTwo->word;
180                     space++;
181                 }
182                 else {
183                     cout << left << setw(15) << runnerTwo->word << endl;
184                     space++;
185                 }
186                 runnerTwo = runnerTwo->next;
187             }
188             if (space % 3 < 1) { cout << endl; }
189             else { cout << endl << endl; }
```

```
190             break;
191         }
192         else runner = runner->next;
193     } while (runner != nullptr);
194 } //display (concordance of specific letter)
195 
196 void concordance::display() {
197     AZNode* runner = top;
198     cout << left;
199     do {
200         cout << char(runner->letter - 32) << ":" << endl;
201         Node* runnerTwo = runner->branch;
202         int space = 0;
203         while (runnerTwo != nullptr) {
204             if (space % 3 < 2) {
205                 cout << left << setw(15) << runnerTwo->word;
206                 space++;
207             }
208             else {
209                 cout << left << setw(15) << runnerTwo->word << endl;
210                 space++;
211             }
212             runnerTwo = runnerTwo->next;
213         }
214         if (space % 3 < 1) { cout << endl; }
215         else { cout << endl << endl; }
216         runner = runner->next;
217     } while (runner != nullptr);
218 } //display (entire concordance)
```