```
2
        Marshall Lindsay
3
        ECE 3220
        Lab8
5
        3/23/17
    * /
6
7
8
9
    #include <iostream>
10
   #include <string>
11
    #include <vector>
12
13
   using namespace std;
14
15
   class Message{
16
        protected:
17
            string msg;
18
19
        public:
20
           Message();
21
            Message(string);
22
            virtual ~Message();
23
            virtual void print();
24 };
25
    /*
26
    Default constructor;
27
    Prompts the user for a new message and adds that message
28
    to the msg string member
29
    * /
30 Message::Message(){
        //cout<<"Constructor"<<endl;
31
32
        string userInput;
33
        cout<<"Please enter a message:"<<endl;</pre>
34
        getline(cin, userInput);
35
36
        //May need some error checking for special characters
37
38
        this -> msg = userInput;
39
    }
   /*
40
41 Parametric constructor;
42 Takes a string in as the message and adds that message to the
43 msg string member;
44 */
45 Message::Message(string userInput){
46
        //cout<<"Constructor"<<endl;</pre>
47
        this -> msg = userInput;
48
49
50
    /*Destructor.
51
    Nothing in this class needs to be deleted;
52
53
   Message::~Message() {
54
        //cout<<"Destructor"<<endl;</pre>
55
56
57
   void Message::print(){
58
        cout<<this ->msg<<endl;</pre>
59
60
61
                 _____
62
    //----Extended Message Class for Morse Code-----
63 class morseCodeMessage: public Message{
64
        protected:
65
            string morseMsg;
66
        public:
67
            morseCodeMessage* next;
68
            morseCodeMessage();
69
            morseCodeMessage(string);
```

```
~morseCodeMessage();
 71
              void translate(string);
 72
              void print();
 73
      };
 74
      //Default constructor. Asks for a message and saves it as a string in
 75
      //the base classes msg member. Then translates that msg into morse code
 76
      //and saves the translation to the extended morseMsq member
 77
     morseCodeMessage::morseCodeMessage(){
 78
          translate(this -> msq);
 79
          this -> next = NULL;
 80
      }
 81
 82
      //Parametric constructor. Takes a string as input. Calls the base
      //constructor with the same string. Then translates the string
 83
 84
      //to morse code and saves it to the extended morseMsg member.
 85
      morseCodeMessage::morseCodeMessage(string userInput) : Message(userInput) {
 86
          translate(this -> msg);
 87
          this -> next = NULL;
 88
      }
 89
 90
      morseCodeMessage::~morseCodeMessage() {
 91
          //cout<<"Destructor"<<endl;
 92
      }
 93
 94
      void morseCodeMessage::translate(string input){
 95
          char test;
 96
          for (int i = 0; i < input.length(); i++) {
 97
              input[i] = tolower(input[i]);
 98
 99
          for (int i = 0; i < input.length(); i++) {
100
              test = input[i];
101
              switch (test) {
102
                  case 'a':
103
                       this->morseMsg.append(".- ");
104
                  case 'b':
105
106
                       this->morseMsg.append("-...");
107
                       break;
108
                  case 'c':
109
                       this->morseMsg.append("-.-. ");
110
                       break;
111
                  case 'd':
112
                       this->morseMsg.append("-..");
113
                       break;
114
                  case 'e':
115
                       this->morseMsg.append(". ");
116
                       break;
117
                  case 'f':
118
                       this->morseMsq.append("..-. ");
119
                       break;
120
                  case 'g':
121
                       this->morseMsg.append("--. ");
122
                       break;
123
                   case 'h':
124
                       this->morseMsg.append("....");
125
                       break;
                  case 'i':
126
127
                       this->morseMsg.append(".. ");
128
                       break;
129
130
                       this->morseMsq.append(".--- ");
131
                       break;
132
                  case 'k':
133
                       this->morseMsg.append("-.- ");
134
                       break;
135
                  case '1':
136
                       this->morseMsg.append(".-.. ");
137
                       break;
138
                  case 'm':
```

```
139
                       this->morseMsq.append("-- ");
140
                       break:
                   case 'n':
141
142
                       this->morseMsg.append("-. ");
143
                       break:
144
                   case 'o':
145
                       this->morseMsg.append("--- ");
146
                       break;
147
                   case 'p':
148
                       this->morseMsg.append(".--. ");
149
                       break;
150
                   case 'q':
1.51
                       this->morseMsg.append("--.- ");
152
                       break:
153
                   case 'r':
154
                       this->morseMsg.append(".-. ");
155
                       break;
156
                   case 's':
157
                       this->morseMsg.append("...");
158
                       break;
159
                   case 't':
160
                       this->morseMsq.append("- ");
161
                       break:
                   case 'u':
162
                       this->morseMsg.append("..- ");
163
164
                       break;
165
                   case 'v':
166
                       this->morseMsq.append("...- ");
167
                       break;
168
                   case 'w':
169
                       this->morseMsg.append(".-- ");
170
                       break;
171
                   case 'x':
172
                       this->morseMsg.append("-..- ");
173
                       break:
174
                   case 'y':
175
                       this->morseMsg.append("-.-- ");
176
                       break;
177
                   case 'z':
178
                       this->morseMsq.append("--.. ");
179
                       break;
                   case '0':
180
181
                       this->morseMsg.append("---- ");
182
183
                   case '1':
                       this->morseMsg.append(".--- ");
184
185
                       break;
186
                   case '2':
187
                       this->morseMsg.append("..-- ");
188
                       break;
189
                   case '3':
190
                       this->morseMsg.append("...- ");
191
                       break;
192
                   case '4':
193
                       this->morseMsq.append("....- ");
194
                       break;
195
                   case '5':
196
                       this->morseMsg.append("....");
197
                       break;
198
                   case '6':
199
                       this->morseMsg.append("-...");
200
                       break;
201
                   case '7':
202
                       this->morseMsg.append("--...");
203
                       break;
204
                   case '8':
205
                       this->morseMsg.append("---..");
206
                       break;
                   case '9':
207
```

```
208
                     this->morseMsg.append("---. ");
209
                     break;
210
                 case ' ':
211
                     this->morseMsg.append(" ");
212
                     break;
213
                 default:
214
                     break;
215
             }//Switch
216
217
         }
218
219
     }
220
221
     void morseCodeMessage::print(){
222
         cout<< this->msg <<endl;</pre>
223
         cout<< this->morseMsq <<endl;</pre>
224
     }
225
     //----
226
227
     //-----Generic Stack Class-----
228
    class messageStack{
229
         private:
230
             morseCodeMessage* top;
231
         public:
232
            messageStack();
233
             ~messageStack();
234
             void push();
235
             void pop();
236
             void printStack();
237
238
    };
239
240
    //Default constructor. Sets the pointer to the top of the
241 //Stack to NULL.
242
     messageStack::messageStack(){
243
         this -> top = NULL;
244
245
246
     //Destructor. This will delete each object in the linked list.
247
     messageStack::~messageStack(){
248
         morseCodeMessage* temp;
249
         while(top != NULL) {
250
                 temp = this -> top;
251
                 this -> top = this -> top -> next;
252
                 delete temp;
253
         }
254
     }
255
256
     //Creates a new message and sets it to the top of the stack.
257
     void messageStack::push(){
258
         morseCodeMessage* newNode = new morseCodeMessage();
259
         newNode -> next = this -> top;
260
         this -> top = newNode;
261
262
263
    //Cycles down the stack and prints each message in turn.
264
    void messageStack::printStack(){
265
         morseCodeMessage* msgPtr = this -> top;
266
         while (msgPtr != NULL) {
267
             msgPtr -> print();
268
             msgPtr = msgPtr -> next;
269
         }
270
271
272
273
    //Deletes the top object from the stack. Sets the top of the
274
    //stack to the next one down.
275
     void messageStack::pop(){
276
         if(top == NULL) {
```

```
cout<<"\nThere is nothing on the stack!"<<endl;</pre>
278
              return;
279
          }
280
          morseCodeMessage* retObj = this -> top;
          this -> top = this -> top -> next;
281
282
          delete retObj;
283
     }
284
285 void optionMenu();
286 void errorMessage();
287
288
    int main(int argc, char* argv[]){
289
          messageStack stack;
290
291
         string userInput;
292
293
          while(1){
294
              optionMenu();
295
              getline(cin, userInput);
              if(userInput == "1"){
296
297
                  stack.push();
298
              }else if(userInput == "2"){
299
                  stack.pop();
              }else if(userInput == "3"){
300
301
                  stack.printStack();
              }else if(userInput == "4"){
302
303
                  return(0);
304
              }else{
305
                  errorMessage();
306
              }
307
          }
308
309
310 void optionMenu(){
311
          cout<<"\nWhat would you like to do?"<<endl;</pre>
312
          cout<<"1) Push message to the stack\n"</pre>
313
              <<"2) Pop message off the stack\n"
314
              <<"3) Print all messages on the stack\n"
315
              <<"4) Quit"<<endl;
316
      }
317
318
     void errorMessage(){
319
          cout<<"Invalid input!"<<endl;</pre>
320
```