

```

1  /*
2   * Programming Challenge 7
3   */
4  #include <cassert>
5  #include <fstream>
6  #include <iostream>
7  #include <map>
8  #include <sstream>
9  #include <string>
10 using namespace std;
11
12 /*
13  * Open and read the contents of a text file. Each line of the
14  * file will contain a single integer of possible values 10, 20,
15  * 30, 40, or 50. Perform the following operations on the input values:
16  * <ul>
17  * <li>10 -- invoke the function onTen</li>
18  * <li>20 -- invoke the function onTwenty</li>
19  * <li>30 -- invoke the function onThirty</li>
20  * <li>40 -- invoke the function onForty</li>
21  * <li>50 -- invoke the function onFifty</li>
22  * <li>any other value -- invoke the function onError</li>
23  * </ul>
24  * @param filename a string containing the name of the file to
25  *               be processed
26  * @return true if filename was successfully opened and processed, else false
27  */
28 bool processFile (string filename);
29
30 /* for unit testing -- do not alter */
31 map<int,int> counters;
32 void onTen ();
33 void onTwenty ();
34 void onThirty ();
35 void onForty ();
36 void onFifty ();
37 void onError ();
38
39 template <typename X, typename A>
40 void btassert(A assertion);
41 void unittest ();
42
43 int main (int argc, char* argv[])
44 {
45     unittest();
46
47     return 0;
48 }
49
50 // CODE HERE -- FUNCTION DEFINITIONS

```

```
51
52 bool processFile( string filename ) {
53
54     ifstream input;
55     int intNumberRead;
56
57     input.open( filename.c_str() );
58
59     if( input.fail() ) {
60
61         return false;
62
63     }
64
65     while( !input.eof() ) {
66
67         input >> intNumberRead;
68
69         switch( intNumberRead ) {
70
71             case 10:
72                 onTen();
73                 break;
74
75             case 20:
76                 onTwenty();
77                 break;
78
79             case 30:
80                 onThirty();
81                 break;
82
83             case 40:
84                 onForty();
85                 break;
86
87             case 50:
88                 onFifty();
89                 break;
90
91             default:
92                 onError();
93                 break;
94
95         }
96
97     }
98
99     input.close();
100
```

```

101     return true;
102
103 }
104
105 /*
106  * Unit testing functions. Do not alter.
107  */
108
109 void unittest ()
110 {
111     cout << "\nSTARTING UNIT TEST\n\n";
112
113     counters[10] = 0, counters[20] = 0, counters[20] = 0, counters[40] = 0, counters[50] = 0;
114     counters[99] = 0; // errors
115
116     processFile("challenge-7-input.txt");
117
118     try {
119         btassert<bool>(counters[10] == 15);
120         cout << "Passed TEST 1: counters[10]\n";
121     } catch (bool b) {
122         cout << "# FAILED TEST 1 counters[10] #\n";
123     }
124
125     try {
126         btassert<bool>(counters[20] == 14);
127         cout << "Passed TEST 2: counters[20]\n";
128     } catch (bool b) {
129         cout << "# FAILED TEST 2 counters[20] #\n";
130     }
131
132     try {
133         btassert<bool>(counters[30] == 13);
134         cout << "Passed TEST 3: counters[30]\n";
135     } catch (bool b) {
136         cout << "# FAILED TEST 3 counters[30] #\n";
137     }
138
139     try {
140         btassert<bool>(counters[40] == 12);
141         cout << "Passed TEST 4: counters[40]\n";
142     } catch (bool b) {
143         cout << "# FAILED TEST 4 counters[40] #\n";
144     }
145
146     try {
147         btassert<bool>(counters[50] == 11);
148         cout << "Passed TEST 5: counters[50]\n";
149     } catch (bool b) {
150         cout << "# FAILED TEST 5 counters[50] #\n";

```

```
151     }
152
153     try {
154         btassert<bool>(counters[99] == 35);
155         cout << "Passed TEST 6: counters[99]\n";
156     } catch (bool b) {
157         cout << "# FAILED TEST 6 counters[99] #\n";
158     }
159
160     try {
161         btassert<bool>(processFile("non-existent-file.txt") == false);
162         cout << "Passed TEST 7: processFile(\"non-existent-file.txt\")\n";
163     } catch (bool b) {
164         cout << "# FAILED TEST 7 processFile(\"non-existent-file.txt\") #\n";
165     }
166
167     cout << "\nUNIT TEST COMPLETE\n\n";
168 }
169
170 void onTen ()
171 {
172     counters[10]++;
173 }
174
175 void onTwenty ()
176 {
177     counters[20]++;
178 }
179
180 void onThirty ()
181 {
182     counters[30]++;
183 }
184
185 void onForty ()
186 {
187     counters[40]++;
188 }
189
190 void onFifty ()
191 {
192     counters[50]++;
193 }
194
195 void onError ()
196 {
197     counters[99]++;
198 }
199
200 template <typename X, typename A>
```

```
201 void btassert (A assertion)
202 {
203     if (!assertion)
204         throw X();
205 }
```