```
/*
 * Programming Challenge 8
 */
#include <cassert>
#include <fstream>
#include <iostream>
#include <map>
#include <sstream>
#include <string>
#include <cstring>
using namespace std;

/*
 * Process the argv array (command-line arguments to the program). Ignore
 * argv[0] as that is the program name. Perform the following operations on
 * the input values:
 * <ul>
 * <li>10 -- invoke the function onTen</li>
 * <li>20 -- invoke the function onTwenty</li>
 * <li>30 -- invoke the function onThirty</li>
 * <li>40 -- invoke the function onForty</li>
 * <li>50 -- invoke the function onFifty</li>
 * <li>any other value -- invoke the function onError</li>
 * </ul>
 * @param argc an integer containing the number of arguments passed to the program
 *        on the command-line
 * @param argv an array containing the command-line arguments
 */
void processArguments (int argc, char* argv[]);

/* for unit testing -- do not alter */
map<int,int> counters;
bool checkArgs (int argc, char* argv[]);
void onTen ();
void onTwenty ();
void onThirty ();
void onForty ();
void onFifty ();
void onError ();

template <typename X, typename A>
void btassert(A assertion);
void unittest (int argc, char* argv[]);

int main (int argc, char* argv[])
{
    unittest(argc, argv);

    return 0;
}
```

```
51
52  // CODE HERE -- FUNCTION DEFINITIONS
53
54  void processArguments( int argc, char* argv[] ) {
55
56      int intArg;
57
58      for( int count = 1; count < argc; count++ ) {
59
60          string test( argv[count] );
61
62          istringstream( test ) >> intArg;
63
64          switch( intArg )    {
65
66              case 10:
67                  onTen();
68                  break;
69
70              case 20:
71                  onTwenty();
72                  break;
73
74              case 30:
75                  onThirty();
76                  break;
77
78              case 40:
79                  onForty();
80                  break;
81
82              case 50:
83                  onFifty();
84                  break;
85
86              default:
87                  onError();
88                  break;
89
90          }
91
92      }
93
94  }
95
96  /*
97   * Unit testing functions. Do not alter.
98   */
99
100 void unittest (int argc, char* argv[])
```

```
101 {
102     if (argc > 1 && strcmp(argv[1], "teacher") == 0 && checkArgs(argc, argv))
103     {
104         cout << "\nSTARTING UNIT TEST\n\n";
105 
106         counters[10] = 0, counters[20] = 0, counters[20] = 0, counters[40] = 0, counters[50] = 0;
107         counters[99] = 0; // errors
108 
109         processArguments(argc, argv);
110 
111         try {
112             btassert<bool>(counters[10] == 1);
113             cout << "Passed TEST 1: counters[10]\n";
114         } catch (bool b) {
115             cout << "# FAILED TEST 1 counters[10] #\n";
116         }
117 
118         try {
119             btassert<bool>(counters[20] == 1);
120             cout << "Passed TEST 2: counters[20]\n";
121         } catch (bool b) {
122             cout << "# FAILED TEST 2 counters[20] #\n";
123         }
124 
125         try {
126             btassert<bool>(counters[30] == 1);
127             cout << "Passed TEST 3: counters[30]\n";
128         } catch (bool b) {
129             cout << "# FAILED TEST 3 counters[30] #\n";
130         }
131 
132         try {
133             btassert<bool>(counters[40] == 1);
134             cout << "Passed TEST 4: counters[40]\n";
135         } catch (bool b) {
136             cout << "# FAILED TEST 4 counters[40] #\n";
137         }
138 
139         try {
140             btassert<bool>(counters[50] == 1);
141             cout << "Passed TEST 5: counters[50]\n";
142         } catch (bool b) {
143             cout << "# FAILED TEST 5 counters[50] #\n";
144         }
145 
146         try {
147             btassert<bool>(counters[99] == 2); // teacher and 60 should be errors
148             cout << "Passed TEST 6: counters[99]\n";
149         } catch (bool b) {
150             cout << "# FAILED TEST 6 counters[99] #\n";
```

```
151         }
152
153         cout << "\nUNIT TEST COMPLETE\n\n";
154     }
155     else
156     {
157         cout << "\nRun program with the following argument list:\n";
158         cout << "\n\t\"teacher 10 20 30 40 50 60\"\n";
159         cout << "\nto run the UNIT TEST.\n\n";
160     }
161 }
162
163 void onTen ()
164 {
165     counters[10]++;
166 }
167
168 void onTwenty ()
169 {
170     counters[20]++;
171 }
172
173 void onThirty ()
174 {
175     counters[30]++;
176 }
177
178 void onForty ()
179 {
180     counters[40]++;
181 }
182
183 void onFifty ()
184 {
185     counters[50]++;
186 }
187
188 void onError ()
189 {
190     counters[99]++;
191 }
192
193 bool checkArgs (int argc, char* argv[])
194 {
195     if (argc == 8)
196     {
197         // convert the argv[2] to argv[7] contents to integers
198         int* temps = new int[6];
199         stringstream ss;
200         for (int i=0, j=2; i<6; i++,j++)
```

```
        {
            ss.str(argv[j]);
            ss >> temps[i];
            ss.clear();
        }

        // check to see that argv[2] to argv[7] match the expected launch UNIT TEST values
        for (int i=0,j=10; i<6; i++,j+=10)
        {
            if (temps[i] != j)
                return false;
        }

        delete [] temps;

        return true;
    }
    return false;
}

template <typename X, typename A>
void btassert (A assertion)
{
    if (!assertion)
        throw X();
}
```