```
/*
 * Programming Challenge 7
 */
#include <cassert>
#include <fstream>
#include <iostream>
#include <map>
#include <sstream>
#include <string>
using namespace std;

/*
 * Open and read the contents of a text file. Each line of the
 * file will contain a single integer of possible values 10, 20,
 * 30, 40, or 50. 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 filename a string containing the name of the file to
 *        be processed
 * @return true if filename was successfully opened and processed, else false
 */
bool processFile (string filename);

/* for unit testing -- do not alter */
map<int,int> counters;
void onTen ();
void onTwenty ();
void onThirty ();
void onForty ();
void onFifty ();
void onError ();

template <typename X, typename A>
void btassert(A assertion);
void unittest ();

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

    return 0;
}

// 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
```

```
    return true;

}

/*
 * Unit testing functions. Do not alter.
 */

void unittest ()
{
    cout << "\nSTARTING UNIT TEST\n\n";

    counters[10] = 0, counters[20] = 0, counters[20] = 0, counters[40] = 0, counters[50] = 0;
    counters[99] = 0; // errors

    processFile("challenge-7-input.txt");

    try {
        btassert<bool>(counters[10] == 15);
        cout << "Passed TEST 1: counters[10]\n";
    } catch (bool b) {
        cout << "# FAILED TEST 1 counters[10] #\n";
    }

    try {
        btassert<bool>(counters[20] == 14);
        cout << "Passed TEST 2: counters[20]\n";
    } catch (bool b) {
        cout << "# FAILED TEST 2 counters[20] #\n";
    }

    try {
        btassert<bool>(counters[30] == 13);
        cout << "Passed TEST 3: counters[30]\n";
    } catch (bool b) {
        cout << "# FAILED TEST 3 counters[30] #\n";
    }

    try {
        btassert<bool>(counters[40] == 12);
        cout << "Passed TEST 4: counters[40]\n";
    } catch (bool b) {
        cout << "# FAILED TEST 4 counters[40] #\n";
    }

    try {
        btassert<bool>(counters[50] == 11);
        cout << "Passed TEST 5: counters[50]\n";
    } catch (bool b) {
        cout << "# FAILED TEST 5 counters[50] #\n";
```

```
    }

    try {
        btassert<bool>(counters[99] == 35);
        cout << "Passed TEST 6: counters[99]\n";
    } catch (bool b) {
        cout << "# FAILED TEST 6 counters[99] #\n";
    }

    try {
        btassert<bool>(processFile("non-existent-file.txt") == false);
        cout << "Passed TEST 7: processFile(\"non-existent-file.txt\")\n";
    } catch (bool b) {
        cout << "# FAILED TEST 7 processFile(\"non-existent-file.txt\") #\n";
    }

    cout << "\nUNIT TEST COMPLETE\n\n";
}

void onTen ()
{
    counters[10]++;
}

void onTwenty ()
{
    counters[20]++;
}

void onThirty ()
{
    counters[30]++;
}

void onForty ()
{
    counters[40]++;
}

void onFifty ()
{
    counters[50]++;
}

void onError ()
{
    counters[99]++;
}

template <typename X, typename A>
```

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