

```

1  /*
2   * Programming Challenge 13
3   */
4  #include <cassert>
5  #include <iostream>
6  #include <string>
7  using namespace std;
8
9  /*
10   * Allocate memory for a dynamic array of integers.
11   * @param size the desired size of the dynamic array
12   * @return a pointer to the newly allocated integer array
13   */
14  int* makeDynoIntArray (unsigned int size);
15
16  /*
17   * Free the memory associated with a dynamic array and NULL out its pointer.
18   * @param theArray a pointer (passed by reference) to a dynamic array of integers
19   */
20  void clearDynoIntArray (int*& theArray);
21
22  /*
23   * Compute the sum of an array.
24   * @param theArray the array for which the sum will be computed
25   * @param arraySize the size of theArray
26   * @return an integer containing the sum of the array
27   * @throw ArrayException with the message "NULL ARRAY REFERENCE" if theArray is NULL
28   */
29  int sum (int* theArray, unsigned int arraySize);
30
31  /*
32   * Identify the max value in an array.
33   * @param theArray the array for which the max value will be identified
34   * @param arraySize the size of theArray
35   * @return an integer containing the max value in the array
36   * @throw ArrayException with the message "NULL ARRAY REFERENCE" if theArray is NULL
37   */
38  int max (int* theArray, unsigned int arraySize);
39
40  /*
41   * Identify the min value in an array.
42   * @param theArray the array for which the min value will be identified
43   * @param arraySize the size of theArray
44   * @return an integer containing the min value in the array
45   * @throw ArrayException with the message "NULL ARRAY REFERENCE" if theArray is NULL
46   */
47  int min (int* theArray, unsigned int arraySize);
48
49  /* for unit testing -- do not alter */
50  struct ArrayException

```

```

51 {
52     ArrayException (string newMessage="error")
53     : message(newMessage)
54     {
55     }
56
57     string message;
58 };
59
60 template <typename X, typename A>
61 void btassert(A assertion);
62 void unittest ();
63
64 int main (int argc, char* argv[])
65 {
66     unittest();
67
68     return 0;
69 }
70
71 // CODE HERE -- FUNCTION DEFINITIONS
72
73 int* makeDynoIntArray (unsigned int size)
74 {
75     return new int[size];
76 }
77
78 void clearDynoIntArray (int*& theArray)
79 {
80     delete [] theArray;
81     theArray = NULL;
82 }
83
84 int sum (int* theArray, unsigned int arraySize)
85 {
86     if( theArray == NULL )
87         throw ArrayException( "NULL ARRAY REFERENCE" );
88
89     int intTotal = 0;
90
91     for( int count = 0; count < arraySize; count++ )
92     {
93         intTotal += theArray[count];
94     }
95
96     return intTotal;
97 }
98
99 int max (int* theArray, unsigned int arraySize)
100 {

```

```

101     if( theArray == NULL )
102         throw ArrayException( "NULL ARRAY REFERENCE" );
103
104     int intMaximumValue = theArray[0];
105
106     for( int count = 0; count < (arraySize - 1); count++ )
107     {
108         if( intMaximumValue < theArray[(count + 1)] )
109             intMaximumValue = theArray[(count + 1)];
110     }
111
112     return intMaximumValue;
113 }
114
115 int min (int* theArray, unsigned int arraySize)
116 {
117     if( theArray == NULL )
118         throw ArrayException( "NULL ARRAY REFERENCE" );
119
120     int intMinimumValue = theArray[0];
121
122     for( int count = 0; count < (arraySize - 1); count++ )
123     {
124         if( intMinimumValue > theArray[(count + 1)] )
125             intMinimumValue = theArray[(count + 1)];
126     }
127
128     return intMinimumValue;
129 }
130
131 /*
132  * Unit testing functions. Do not alter.
133  */
134
135 void unittest ()
136 {
137     cout << "\nSTARTING UNIT TEST\n\n";
138
139     int* myArray = 0; // = makeDynoIntArray(10);
140     unsigned int myArraySize = 0;
141
142     try {
143         sum(myArray, myArraySize);
144     } catch (ArrayException e) {
145         try {
146             btassert<bool>(e.message == "NULL ARRAY REFERENCE");
147             cout << "Passed TEST 1: sum EXCEPTION HANDLING (INT*) () \n";
148         } catch (bool b) {
149             cout << "# FAILED TEST 1: sum EXCEPTION HANDLING (INT*) () #\n";
150         }
151     }

```

```

151     }
152
153     try {
154         min(myArray, myArraySize);
155     } catch (ArrayException e) {
156         try {
157             btassert<bool>(e.message == "NULL ARRAY REFERENCE");
158             cout << "Passed TEST 2: min EXCEPTION HANDLING (INT*) () \n";
159         } catch (bool b) {
160             cout << "# FAILED TEST 2: min EXCEPTION HANDLING (INT*) () #\n";
161         }
162     }
163
164     try {
165         max(myArray, myArraySize);
166     } catch (ArrayException e) {
167         try {
168             btassert<bool>(e.message == "NULL ARRAY REFERENCE");
169             cout << "Passed TEST 3: max EXCEPTION HANDLING (INT*) () \n";
170         } catch (bool b) {
171             cout << "# FAILED TEST 3: max EXCEPTION HANDLING (INT*) () #\n";
172         }
173     }
174
175     myArray = makeDynoIntArray(3);
176
177     try {
178         btassert<bool>(myArray != 0);
179         cout << "Passed TEST 4: INT ARRAY INITIALIZATION () \n";
180     } catch (bool b) {
181         cout << "# FAILED TEST 4: INT ARRAY INITIALIZATION () #\n";
182     }
183
184     myArray[0] = 30, myArray[1] = 20, myArray[2] = 10;
185
186     try {
187         btassert<bool>(sum(myArray, 3) == 60);
188         cout << "Passed TEST 5: sum (array) \n";
189     } catch (bool b) {
190         cout << "# FAILED TEST 5: sum (array) #\n";
191     }
192
193     try {
194         btassert<bool>(min(myArray, 3) == 10);
195         cout << "Passed TEST 6: min (array) \n";
196     } catch (bool b) {
197         cout << "# FAILED TEST 6: min (array) #\n";
198     }
199
200     myArray[0] = 30, myArray[1] = 10, myArray[2] = 20;

```

```

201
202     try {
203         btassert<bool>(min(myArray, 3) == 10);
204         cout << "Passed TEST 7: min (array) \n";
205     } catch (bool b) {
206         cout << "# FAILED TEST 7: min (array) #\n";
207     }
208
209     myArray[0] = 30, myArray[1] = 20, myArray[2] = 10;
210
211     try {
212         btassert<bool>(max(myArray, 3) == 30);
213         cout << "Passed TEST 8: max (array) \n";
214     } catch (bool b) {
215         cout << "# FAILED TEST 8: max (array) #\n";
216     }
217
218     myArray[0] = 20, myArray[1] = 10, myArray[2] = 30;
219
220     try {
221         btassert<bool>(max(myArray, 3) == 30);
222         cout << "Passed TEST 9: max (array) \n";
223     } catch (bool b) {
224         cout << "# FAILED TEST 9: max (array) #\n";
225     }
226
227     clearDynoIntArray(myArray);
228
229     try {
230         btassert<bool>(myArray == 0);
231         cout << "Passed TEST 10: clearDynoArray () \n";
232     } catch (bool b) {
233         cout << "# FAILED TEST 10: clearDynoArray () #\n";
234     }
235
236     cout << "\nUNIT TEST COMPLETE\n\n";
237 }
238
239 template <typename X, typename A>
240 void btassert (A assertion)
241 {
242     if (!assertion)
243         throw X();
244 }

```