```cpp
/*
 * Programming Challenge 13
 */
#include <cassert>
#include <iostream>
#include <string>
using namespace std;

/*
 * Allocate memory for a dynamic array of integers.
 * @param size the desired size of the dynamic array
 * @return a pointer to the newly allocated integer array
 */
int* makeDynoIntArray (unsigned int size);

/*
 * Free the memory associated with a dynamic array and NULL out its pointer.
 * @param theArray a pointer (passed by reference) to a dynamic array of integers
 */
void clearDynoIntArray (int*& theArray);

/*
 * Compute the sum of an array.
 * @param theArray the array for which the sum will be computed
 * @param arraySize the size of theArray
 * @return an integer containing the sum of the array
 * @throw ArrayException with the message "NULL ARRAY REFERENCE" if theArray is NULL
 */
int sum (int* theArray, unsigned int arraySize);

/*
 * Identify the max value in an array.
 * @param theArray the array for which the max value will be identified
 * @param arraySize the size of theArray
 * @return an integer containing the max value in the array
 * @throw ArrayException with the message "NULL ARRAY REFERENCE" if theArray is NULL
 */
int max (int* theArray, unsigned int arraySize);

/*
 * Identify the min value in an array.
 * @param theArray the array for which the min value will be identified
 * @param arraySize the size of theArray
 * @return an integer containing the min value in the array
 * @throw ArrayException with the message "NULL ARRAY REFERENCE" if theArray is NULL
 */
int min (int* theArray, unsigned int arraySize);

/* for unit testing -- do not alter */
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 {
```

```
    if( theArray == NULL )
        throw ArrayException( "NULL ARRAY REFERENCE" );

    int intMaximumValue = theArray[0];

    for( int count = 0; count < (arraySize - 1); count++ )
    {
        if( intMaximumValue < theArray[(count + 1)] )
            intMaximumValue = theArray[(count + 1)];
    }

    return intMaximumValue;
}

int min (int* theArray, unsigned int arraySize)
{
    if( theArray == NULL )
        throw ArrayException( "NULL ARRAY REFERENCE" );

    int intMinimumValue = theArray[0];

    for( int count = 0; count < (arraySize - 1); count++ )
    {
        if( intMinimumValue > theArray[(count + 1)] )
            intMinimumValue = theArray[(count + 1)];
    }

    return intMinimumValue;
}

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

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

    int* myArray = 0; // = makeDynoIntArray(10);
    unsigned int myArraySize = 0;

    try {
        sum(myArray, myArraySize);
    } catch (ArrayException e) {
        try {
            btassert<bool>(e.message == "NULL ARRAY REFERENCE");
            cout << "Passed TEST 1: sum EXCEPTION HANDLING (INT*) () \n";
        } catch (bool b) {
            cout << "# FAILED TEST 1: sum EXCEPTION HANDLING (INT*) () #\n";
        }
```

```
    }

    try {
        min(myArray, myArraySize);
    } catch (ArrayException e) {
        try {
            btassert<bool>(e.message == "NULL ARRAY REFERENCE");
            cout << "Passed TEST 2: min EXCEPTION HANDLING (INT*) () \n";
        } catch (bool b) {
            cout << "# FAILED TEST 2: min EXCEPTION HANDLING (INT*) () #\n";
        }
    }

    try {
        max(myArray, myArraySize);
    } catch (ArrayException e) {
        try {
            btassert<bool>(e.message == "NULL ARRAY REFERENCE");
            cout << "Passed TEST 3: max EXCEPTION HANDLING (INT*) () \n";
        } catch (bool b) {
            cout << "# FAILED TEST 3: max EXCEPTION HANDLING (INT*) () #\n";
        }
    }

    myArray = makeDynoIntArray(3);

    try {
        btassert<bool>(myArray != 0);
        cout << "Passed TEST 4: INT ARRAY INITIALIZATION () \n";
    } catch (bool b) {
        cout << "# FAILED TEST 4: INT ARRAY INITIALIZATION () #\n";
    }

    myArray[0] = 30, myArray[1] = 20, myArray[2] = 10;

    try {
        btassert<bool>(sum(myArray, 3) == 60);
        cout << "Passed TEST 5: sum (array) \n";
    } catch (bool b) {
        cout << "# FAILED TEST 5: sum (array) #\n";
    }

    try {
        btassert<bool>(min(myArray, 3) == 10);
        cout << "Passed TEST 6: min (array) \n";
    } catch (bool b) {
        cout << "# FAILED TEST 6: min (array) #\n";
    }

    myArray[0] = 30, myArray[1] = 10, myArray[2] = 20;
```

```
    try {
        btassert<bool>(min(myArray, 3) == 10);
        cout << "Passed TEST 7: min (array) \n";
    } catch (bool b) {
        cout << "# FAILED TEST 7: min (array) #\n";
    }

    myArray[0] = 30, myArray[1] = 20, myArray[2] = 10;

    try {
        btassert<bool>(max(myArray, 3) == 30);
        cout << "Passed TEST 8: max (array) \n";
    } catch (bool b) {
        cout << "# FAILED TEST 8: max (array) #\n";
    }

    myArray[0] = 20, myArray[1] = 10, myArray[2] = 30;

    try {
        btassert<bool>(max(myArray, 3) == 30);
        cout << "Passed TEST 9: max (array) \n";
    } catch (bool b) {
        cout << "# FAILED TEST 9: max (array) #\n";
    }

    clearDynoIntArray(myArray);

    try {
        btassert<bool>(myArray == 0);
        cout << "Passed TEST 10: clearDynoArray () \n";
    } catch (bool b) {
        cout << "# FAILED TEST 10: clearDynoArray () #\n";
    }

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

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