```
/*
 * Programming Challenge 16
 */
#include <cassert>
#include <iostream>
using namespace std;

/*
 * Apply the bubble sort algorithm to sort an array of integers.
 * @param theArray the integer array to be sorted
 * @param size an unsigned integer containing the size of theArray
 */
void bubbleSort (int theArray[], unsigned int size);

/*
 * Apply the selection sort algorithm to sort an array of integers.
 * @param theArray the integer array to be sorted
 * @param size an unsigned integer containing the size of theArray
 */
void selectionSort (int theArray[], unsigned int size);

/*
 * Swap the positions of two values in an integer array. The two
 * index values must be valid for the array.
 * @param values an integer array
 * @param index1 the position of the first value to be swapped
 * @param index2 the position of the second value to be swapped
 */
void swapValues (int values[], int index1, int index2);

/* for unit testing -- do not alter */
template <typename X, typename A>
void btassert(A assertion);
void unittest ();
bool compareArrays (int arrayOne[], int arrayTwo[], unsigned int size);

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

    return 0;
}

// CODE HERE -- FUNCTION DEFINITIONS

/*
 * Apply the bubble sort algorithm to sort an array of integers.
 * @param theArray the integer array to be sorted
 * @param size an unsigned integer containing the size of theArray
 */
```

```
51  void bubbleSort (int theArray[], unsigned int size)
52  {
53      bool blnSwapped = false;
54      int intSwapValue    = 0;
55
56      do {
57          blnSwapped = false;
58          for( int count = 1; count <= size - 1; count++ )
59          {
60              if( theArray[(count - 1)] > theArray[count] )
61              {
62                  swapValues( theArray, (count - 1), count );
63                  blnSwapped = true;
64              }
65          }
66
67          size--;
68
69      } while( blnSwapped );
70  }
71
72  void selectionSort (int theArray[], unsigned int size)
73  {
74      int intPosition     = 0;
75      int intSwapValue    = 0;
76
77      for( int count01 = 0; count01 < size; count01++ )
78      {
79          intPosition = count01;
80
81          for( int count02 = count01 + 1; count02 < size; count02++ )
82          {
83              if( theArray[count02] < theArray[intPosition] )
84                  intPosition = count02;
85          }
86
87          swapValues( theArray, count01, intPosition );
88      }
89  }
90
91  void swapValues (int values[], int index1, int index2)
92  {
93      int intSwapValue    = 0;
94
95      intSwapValue    = values[index1];
96      values[index1]  = values[index2];
97      values[index2]  = intSwapValue;
98  }
99
100 /*
```

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

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

    int master[5] = {10, 20, 30, 40, 50};
    int tester[5] = {50, 40, 30, 20, 10};

    bubbleSort(tester, 5);
    try {
        btassert<bool>(compareArrays(master, tester, 5));
        cout << "Passed TEST 1: bubble sort (50,40,30,20,10) \n";
    } catch (bool b) {
        cout << "# FAILED TEST 1: bubble sort (50,40,30,20,10) #\n";
    }

    bubbleSort(tester, 5);
    try {
        btassert<bool>(compareArrays(master, tester, 5));
        cout << "Passed TEST 2: bubble sort (10,20,30,40,50) \n";
    } catch (bool b) {
        cout << "# FAILED TEST 2: bubble sort (10,20,30,40,50) #\n";
    }

    tester[0]=10, tester[1]=30, tester[2]=20, tester[3]=50, tester[4]=40;

    bubbleSort(tester, 5);
    try {
        btassert<bool>(compareArrays(master, tester, 5));
        cout << "Passed TEST 3: bubble sort (10,30,20,50,40) \n";
    } catch (bool b) {
        cout << "# FAILED TEST 3: bubble sort (10,30,20,50,40) #\n";
    }

    master[0]=10, master[1]=30, master[2]=30, master[3]=50, master[4]=50;
    tester[0]=50, tester[1]=30, tester[2]=10, tester[3]=30, tester[4]=50;

    bubbleSort(tester, 5);
    try {
        btassert<bool>(compareArrays(master, tester, 5));
        cout << "Passed TEST 4: bubble sort (50,30,10,30,50) \n";
    } catch (bool b) {
        cout << "# FAILED TEST 4: bubble sort (50,30,10,30,50) #\n";
    }

    master[0]=50, master[1]=50, master[2]=50, master[3]=50, master[4]=50;
    tester[0]=50, tester[1]=50, tester[2]=50, tester[3]=50, tester[4]=50;

```

```
    bubbleSort(tester, 5);
    try {
        btassert<bool>(compareArrays(master, tester, 5));
        cout << "Passed TEST 5: bubble sort (50,50,50,50,50) \n";
    } catch (bool b) {
        cout << "# FAILED TEST 5: bubble sort (50,50,50,50,50) #\n";
    }

    master[0]=10, master[1]=20, master[2]=30, master[3]=40, master[4]=50;
    tester[0]=50, tester[1]=40, tester[2]=30, tester[3]=20, tester[4]=10;

    selectionSort(tester, 5);
    try {
        btassert<bool>(compareArrays(master, tester, 5));
        cout << "Passed TEST 6: selection sort (50,40,30,20,10) \n";
    } catch (bool b) {
        cout << "# FAILED TEST 6: selection sort (50,40,30,20,10) #\n";
    }

    selectionSort(tester, 5);
    try {
        btassert<bool>(compareArrays(master, tester, 5));
        cout << "Passed TEST 7: selection sort (10,20,30,40,50) \n";
    } catch (bool b) {
        cout << "# FAILED TEST 7: selection sort (10,20,30,40,50) #\n";
    }

    tester[0]=10, tester[1]=30, tester[2]=20, tester[3]=50, tester[4]=40;

    selectionSort(tester, 5);
    try {
        btassert<bool>(compareArrays(master, tester, 5));
        cout << "Passed TEST 8: selection sort (10,30,20,50,40) \n";
    } catch (bool b) {
        cout << "# FAILED TEST 8: selection sort (10,30,20,50,40) #\n";
    }

    master[0]=10, master[1]=30, master[2]=30, master[3]=50, master[4]=50;
    tester[0]=50, tester[1]=30, tester[2]=10, tester[3]=30, tester[4]=50;

    selectionSort(tester, 5);
    try {
        btassert<bool>(compareArrays(master, tester, 5));
        cout << "Passed TEST 9: selection sort (50,30,10,30,50) \n";
    } catch (bool b) {
        cout << "# FAILED TEST 9: selection sort (50,30,10,30,50) #\n";
    }

    master[0]=50, master[1]=50, master[2]=50, master[3]=50, master[4]=50;
    tester[0]=50, tester[1]=50, tester[2]=50, tester[3]=50, tester[4]=50;
```

```
    selectionSort(tester, 5);
    try {
        btassert<bool>(compareArrays(master, tester, 5));
        cout << "Passed TEST 10: selection sort (50,50,50,50,50) \n";
    } catch (bool b) {
        cout << "# FAILED TEST 10: selection sort (50,50,50,50,50) #\n";
    }

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

bool compareArrays (int arrayOne[], int arrayTwo[], unsigned int size)
{
    for (unsigned int i=0; i<size; i++)
        if (arrayOne[i] != arrayTwo[i])
            return false;

    return true;
}

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