```cpp
//
//  main.cpp
//  AbsoluteCpp_ch5_1
//

//Reads in 5 scores and shows how much each
//score differs from the highest score.
#include <iostream>
#include <cstdlib>
#include <fstream>
using namespace std;

int main( )
{
    int i, score[5], max;
    char* word = "word";

    cout << "Enter 5 scores:\n";
    cin >> score[0];
    max = score[0];
    for (i = 1; i < 5; i++)
    {
        cin >> score[i];
        if (score[i] > max)
            max = score[i];
        //max is the largest of the values score[0],..., score[i].
    }

    cout << "The highest score is " << max << endl
         << "The scores and their\n"
         << "differences from the highest are:\n";
    for (i = 0; i < 5; i++)
        cout << score[i] << " off by "
             << (max - score[i]) << endl;

    // test out of boundary of array
    cout << "Let's step out of boundary score[5] is " << score[5] << endl;
    cout << "Let's step out of boundary score[6] is " << score[6] << endl;
    cout << "Let's step out of boundary word[4] is " << word[4] << endl;
    cout << "Let's step out of boundary word[5] is " << word[5] << endl;
    cout << "Size of word is " << sizeof(word) << endl;
    return 0;
}
```

```
//
//  main.cpp
//  AbsoluteCpp_ch5_4
//

//Reads data and displays a bar graph showing productivity for each plant.
#include <iostream>
#include <cmath>
using namespace std;
const int NUMBER_OF_PLANTS = 4;

void inputData(int a[], int lastPlantNumber);
//Precondition: lastPlantNumber is the declared size of the array a.
//Postcondition: For plantNumber = 1 through lastPlantNumber:
//a[plantNumber-1] equals the total production for plant number plantNumber.

void scale(int a[], int size);
//Precondition: a[0] through a[size-1] each have a nonnegative value.
//Postcondition: a[i] has been changed to the number of 1000s (rounded to
//an integer) that were originally in a[i], for all i such that 0 <= i <=
 size-1.

void graph(const int asteriskCount[], int lastPlantNumber);
//Precondition: a[0] through a[lastPlantNumber-1] have nonnegative values.
//Postcondition: A bar graph has been displayed saying that plant
//number N has produced a[N-1] 1000s of units, for each N such that
//1 <= N <= lastPlantNumber

void getTotal(int& sum);
//Reads nonnegative integers from the keyboard and
//places their total in sum.

int roundNew(double number);
//Precondition: number >= 0.
//Returns number rounded to the nearest integer.

void printAsterisks(int n);
//Prints n asterisks to the screen.

int main( )
{
    int production[NUMBER_OF_PLANTS];

    cout << "This program displays a graph showing\n"
         << "production for each plant in the company.\n";

    inputData(production, NUMBER_OF_PLANTS);
    scale(production, NUMBER_OF_PLANTS);
    graph(production, NUMBER_OF_PLANTS);
    return 0;
}

void inputData(int a[], int lastPlantNumber)
{
    for (int plantNumber = 1;
```

```
                    plantNumber <= lastPlantNumber; plantNumber++)
    {
        cout << endl
             << "Enter production data for plant number "
             << plantNumber << endl;
        getTotal(a[plantNumber - 1]);
    }
}

void getTotal(int& sum)
{
    cout << "Enter number of units produced by each department.\n"
         << "Append a negative number to the end of the list.\n";

    sum = 0;
    int next;
    cin >> next;
    while (next >= 0)
    {
        sum = sum + next;
        cin >> next;
    }

    cout << "Total = " << sum << endl;
}

void scale(int a[], int size)
{
    for (int index = 0; index < size; index++)
        a[index] = roundNew(a[index]/1000.0);
}

int roundNew(double number){
    return static_cast<int>(floor(number + 0.5));
}

void graph(const int asteriskCount[], int lastPlantNumber)
{
    cout << "\nUnits produced in thousands of units:\n";
    for (int plantNumber = 1;
                plantNumber <= lastPlantNumber; plantNumber++)
    {
        cout << "Plant #" << plantNumber << " ";
        printAsterisks(asteriskCount[plantNumber - 1]);
        cout << endl;
    }
}

void printAsterisks(int n)
{
    for (int count = 1; count <= n; count++)
        cout << "*";
}
```

```
//
//  main.cpp
//  AbsoluteCpp_ch5_6
//

//Searches a partially filled array of nonnegative integers.
#include <iostream>
using namespace std;
const int DECLARED_SIZE = 20;

void fillArray(int a[], int size, int& numberUsed);
//Precondition: size is the declared size of the array a.
//Postcondition: numberUsed is the number of values stored in a.
//a[0] through a[numberUsed-1] have been filled with
//nonnegative integers read from the keyboard.

template <class T>
int search(const T a[], int numberUsed, T target);
//Precondition: numberUsed is <= the declared size of a.
//Also, a[0] through a[numberUsed -1] have values.
//Returns the first index such that a[index] == target,
//provided there is such an index, otherwise returns -1.

int main( )
{
    int arr[DECLARED_SIZE], listSize, target;

    fillArray(arr, DECLARED_SIZE, listSize);

    char ans;
    int result;
    do
    {
        cout << "Enter a number to search for: ";
        cin >> target;

        result = search(arr, listSize, target);
        if (result == -1)
            cout << target << " is not on the list.\n";
        else
            cout << target << " is stored in array position "
                 << result << endl
                 << "(Remember: The first position is 0.)\n";

        cout << "Search again?(y/n followed by return): ";
        cin >> ans;
    }while ((ans != 'n') && (ans != 'N'));

    cout << "End of program.\n";
    return 0;
}

void fillArray(int a[], int size, int& numberUsed)
{
    cout << "Enter up to " << size << " nonnegative whole numbers.\n"
```

```
          << "Mark the end of the list with a negative number.\n";

    int next, index = 0;
    cin >> next;
    while ((next >= 0) && (index < size))
    {
        a[index] = next;
        index++;
        cin >> next;
    }

    numberUsed = index;
}

template <class T>
int search(const T a[], int numberUsed, T target)
{
    int index = 0;
    bool found = false;
    while ((!found) && (index < numberUsed))
    if (target == a[index])
        found = true;
    else
        index++;

    if (found)
        return index;
    else
        return -1;
}
```

```
//
//  main.cpp
//  AbsoluteCpp_ch5_8
//

//Tests the procedure sort.
#include <iostream>
using namespace std;

void fillArray(int a[], int size, int& numberUsed);
//Precondition: size is the declared size of the array a.
//Postcondition: numberUsed is the number of values stored in a.
//a[0] through a[numberUsed - 1] have been filled with
//nonnegative integers read from the keyboard.

// selection sort
void sort(int a[], int numberUsed);
//Precondition: numberUsed <= declared size of the array a.
//The array elements a[0] through a[numberUsed - 1] have values.
//Postcondition: The values of a[0] through a[numberUsed - 1] have
//been rearranged so that a[0] <= a[1] <= ... <= a[numberUsed - 1].

void swapValues(int& v1, int& v2);
//Interchanges the values of v1 and v2.

int indexOfSmallest(const int a[], int startIndex, int numberUsed);
//Precondition: 0 <= startIndex < numberUsed. References array elements
 have values.
//Returns the index i such that a[i] is the smallest of the values
//a[startIndex], a[startIndex + 1], ..., a[numberUsed - 1].

int main() {
    cout << "This program sorts numbers from lowest to highest.\n";

    int sampleArray[10], numberUsed;
    fillArray(sampleArray, 10, numberUsed);
    sort(sampleArray, numberUsed);

    cout << "In sorted order the numbers are:\n";
    for (int index = 0; index < numberUsed; index++)
        cout << sampleArray[index] << " ";
    cout << endl;

    return 0;
}

void fillArray(int a[], int size, int& numberUsed) {
    cout << "Enter up to " << size << " nonnegative whole numbers.\n"
            << "Mark the end of the list with a negative number.\n";

    int next, index = 0;
    cin >> next;
    while ((next >= 0) && (index < size)) {
        a[index] = next;
        index++;
```

```
        cin >> next;
    }

    numberUsed = index;
}

void sort(int a[], int numberUsed) {
    int indexOfNextSmallest;
    for (int index = 0; index < numberUsed - 1; index++) { //Place the
     correct value in a[index]:
        indexOfNextSmallest = indexOfSmallest(a, index, numberUsed);
        swapValues(a[index], a[indexOfNextSmallest]);
        //a[0] <= a[1] <=...<= a[index] are the smallest of the original
         array
        //elements. The rest of the elements are in the remaining positions.
    }
}

void swapValues(int& v1, int& v2) {
    int temp;
    temp = v1;
    v1 = v2;
    v2 = temp;
}

int indexOfSmallest(const int a[], int startIndex, int numberUsed) {
    int min = a[startIndex], indexOfMin = startIndex;
    for (int index = startIndex + 1; index < numberUsed; index++)
        if (a[index] < min) {
            min = a[index];
            indexOfMin = index;
            //min is the smallest of a[startIndex] through a[index]
        }

    return indexOfMin;
}
```

```cpp
//
//  main.cpp
//  AbsoluteCpp_ch5_9
//

//Sorts an array of integers using Bubble Sort.
#include <iostream>
using namespace std;

void bubblesort(int arr[], int length);
//Precondition: length <= declared size of the array arr.
//The array elements arr[0] through a[length - 1] have values.
//Postcondition: The values of arr[0] through arr[length - 1] have
//been rearranged so that arr[0] <= a[1] <= ... <= arr[length - 1].

int main() {
    int a[] = { 3, 10, 9, 2, 5, 1 };

    bubblesort(a, 6);
    for (int i = 0; i < 6; i++) {
        cout << a[i] << " ";
    }
    cout << endl;
    return 0;
}

void bubblesort(int arr[], int length) {
    // Bubble largest number toward the right
    for (int i = length - 1; i > 0; i--)
        for (int j = 0; j < i; j++)
            if (arr[j] > arr[j + 1]) {
                // Swap the numbers
                int temp = arr[j + 1];
                arr[j + 1] = arr[j];
                arr[j] = temp;
            }

}
```

```
//
//  main.cpp
//  AbsoluteCpp_ch5_10
//

//Reads quiz scores for each student into the two-dimensional array grade
 (Code to fill array has been added).
//Computes the average score for each student and
//the average score for each quiz. Displays the quiz scores and the
 averages.
#include <iostream>
#include <iomanip>
using namespace std;
const int NUMBER_STUDENTS = 4, NUMBER_QUIZZES = 3;

void computeStAve(const int grade[][NUMBER_QUIZZES], double stAve[]);
//Precondition: Global constant NUMBER_STUDENTS and NUMBER_QUIZZES
//are the dimensions of the array grade. Each of the indexed variables
//grade[stNum-1, quizNum-1] contains the score for student stNum on quiz
 quizNum.
//Postcondition: Each stAve[stNum-1] contains the average for student
 number stuNum.

void computeQuizAve(const int grade[][NUMBER_QUIZZES], double quizAve[]);
//Precondition: Global constant NUMBER_STUDENTS and NUMBER_QUIZZES
//are the dimensions of the array grade. Each of the indexed variables
//grade[stNum-1, quizNum-1] contains the score for student stNum on quiz
 quizNum.
//Postcondition: Each quizAve[quizNum-1] contains the average for quiz
 numbered
//quizNum.

void display(const int grade[][NUMBER_QUIZZES], const double stAve[],
        const double quizAve[]);
//Precondition: Global constant NUMBER_STUDENTS and NUMBER_QUIZZES are the
//dimensions of the array grade. Each of the indexed variables
 grade[stNum-1,
//quizNum-1] contains the score for student stNum on quiz quizNum. Each
//stAve[stNum-1] contains the average for student stuNum. Each
 quizAve[quizNum-1]
//contains the average for quiz numbered quizNum.
//Postcondition: All the data in grade, stAve, and quizAve have been output.

int main() {
    int grade[NUMBER_STUDENTS][NUMBER_QUIZZES];
    double stAve[NUMBER_STUDENTS];
    double quizAve[NUMBER_QUIZZES];

    grade[0][0] = 10;
    grade[0][1] = 10;
    grade[0][2] = 10;
    grade[1][0] = 2;
    grade[1][1] = 0;
    grade[1][2] = 1;
    grade[2][0] = 8;
```

```
    grade[2][1] = 6;
    grade[2][2] = 9;
    grade[3][0] = 8;
    grade[3][1] = 4;
    grade[3][2] = 10;

    computeStAve(grade, stAve);
    computeQuizAve(grade, quizAve);
    display(grade, stAve, quizAve);
    return 0;
}

void computeStAve(const int grade[][NUMBER_QUIZZES], double stAve[]) {
    for (int stNum = 1; stNum <= NUMBER_STUDENTS; stNum++) { //Process one
     stNum:
        double sum = 0;
        for (int quizNum = 1; quizNum <= NUMBER_QUIZZES; quizNum++)
            sum = sum + grade[stNum - 1][quizNum - 1];
        //sum contains the sum of the quiz scores for student number stNum.
        stAve[stNum - 1] = sum / NUMBER_QUIZZES;
        //Average for student stNum is the value of stAve[stNum-1]
    }
}

void computeQuizAve(const int grade[][NUMBER_QUIZZES], double quizAve[]) {
    for (int quizNum = 1; quizNum <= NUMBER_QUIZZES; quizNum++) { //Process
     one quiz (for all students):
        double sum = 0;
        for (int stNum = 1; stNum <= NUMBER_STUDENTS; stNum++)
            sum = sum + grade[stNum - 1][quizNum - 1];
        //sum contains the sum of all student scores on quiz number quizNum.
        quizAve[quizNum - 1] = sum / NUMBER_STUDENTS;
        //Average for quiz quizNum is the value of quizAve[quizNum-1]
    }
}

void display(const int grade[][NUMBER_QUIZZES], const double stAve[],
        const double quizAve[]) {
    cout.setf(ios::fixed);
    cout.setf(ios::showpoint);
    cout.precision(1);

    cout << setw(10) << "Student" << setw(5) << "Ave" << setw(15)
            << "Quizzes\n";
    for (int stNum = 1; stNum <= NUMBER_STUDENTS; stNum++) { //Display for
     one stNum:
        cout << setw(10) << stNum << setw(5) << stAve[stNum - 1] << " ";
        for (int quizNum = 1; quizNum <= NUMBER_QUIZZES; quizNum++)
            cout << setw(5) << grade[stNum - 1][quizNum - 1];
        cout << endl;
    }

    cout << "Quiz averages = ";
    for (int quizNum = 1; quizNum <= NUMBER_QUIZZES; quizNum++)
        cout << setw(5) << quizAve[quizNum - 1];
```

```
    cout << endl;
}
```