CIS 251 • C++ Programming
Laboratory Assignment 8: Arrays

In this laboratory assignment you will apply C++ syntax to store a set of values in an array and process these values in various ways during the execution of the program.

Background

An instructor recently gave an exam and wants to know which students earned each of the various letter grades on the exam. Specifically, the instructor would like a list of all students who earned an A, then all students who earned a B, then all students who earned a C, etc. The exam grades are whole numbers, and the letters assigned follow the usual distribution:

Letter Grade	Range
A	90 or above
B	80 to 89
C	70 to 79
D	60 to 69
F	59 or below

The instructor will enter these grades in the console, and the results should appear in the console as well.

Step 1: Launch Visual Studio and Set Up an Empty C++ Project with Code

Open Visual Studio from the Start Menu. Start a new project of installed template / type **Visual C++**, subtype **Win32**, template **Win32 Console Application**, project name and location as desired (usually the Visual Studio **Projects** folder), and **Create directory for solution** box **unchecked**. In the **Win32 Application Wizard**, go to **Application Settings**, **check** the box for **Empty project**, and click **Finish**.

In the **Solution Explorer**, right-click **Source Files**, and select **Add > New Item…** from the shortcut menu. Select the category **Code** and the file type / template **C++ File (.cpp)**. Enter the file name **FLast_Lab08.cpp** (with your **first initial** and **last name** replacing **F** and **Last**, respectively). Click **Browse…** to select the location where you want to save your code (flash drive preferred).

Step 2: Begin Your Code

Add a single block comment or a set of line comments including your name, the course number, the laboratory assignment number, the date (optional), and a brief description of the program's contents. Place each of these items on a separate line. As you continue your code, you may go ahead and add comments to the statements or groups of statements that you believe deserve additional explanation, or you may wait until after you have written your entire program to add comments.

Next, enter the standard items that are part of your program: the `include` and `using` directives, the header for the `main` function, the opening curly bracket, a blank line after the bracket, the `return` statement, and the closing curly bracket.

Step 3: Declare Your Variables and Constant

The following table describes the variables and constant needed in your main function (one per table row):

Description	Data Form	Initial Value(s)
Array size constant	Whole number	15
Array of exam grades	Whole number array	From input
Subscript variable	Whole number	Assigned in loops

Add a declaration to the `main` function for each constant and variable.

Declare the size constant first (use all uppercase letters for the name); then, use the size constant as the size when declaring the array (the size goes inside the square brackets).

Step 4: Create the Input Loop

Prompt the user with a message to enter 15 exam grades. Use a loop to read these grades directly into the array.

Your array subscript must be between 0 and one less than the size of the array. Initialize the subscript variable to 0, and loop while this variable is less than the size constant.

Step 5: Use a Loop to Display Specific Values

The remainder of the program will be a series of additional loops. The first of these new loops will be responsible for displaying the numbers (from 1 to 15) of the students who earned an A. First, use an output statement to display the label for the beginning of this line (with a space or two after the colon):

```
Students who earned an A:
```

An output statement preceding a set of values displayed in a loop should not be written inside the loop! Instead, it should be written just before the loop header.

Then, write a loop that goes through all 15 array elements. Inside the loop body, check to see if the exam grade at the current subscript is in the range for an A; if it is, display the value one greater than the subscript (if the grade at subscript 3 is an A, the program would display 4) followed by a space. After the loop terminates, the line should look like this (the actual student numbers will vary depending on the input):

```
Students who earned an A: 4 7 9 12
```

When you're dealing with a set of user-entered values and a series of subscripts, keep in mind that the user probably won't think of the first value as number "zero". You may need to add 1 to the subscript when displaying it to make sense to the user.

Resist the temptation to put `"\n"` or `endl` at the end of each output statement. If the values in separate output statements are supposed to be on the same line, you shouldn't break it up with new lines. Instead, put a single space inside quotation marks as the separation between values.

After this loop, use an output statement to start a new line of output in the console (that is, write an output statement containing either `endl` or `"\n"`.

Step 6: Write Additional Loops for Output

Repeat step 5 for each of the four additional grade ranges. Remember: you'll have one output statement before the loop to label the grades with the letter grade, the loop with a nested decision to only display the students whose grades fit within the range for that letter grade, and one output statement after the loop to move to the next line. When finished, your program will contain a total of six loops.

Step 7: Build Your Project and Run the Executable

Build your project. If there are syntax errors, double-click each error description to get an idea of where the error is. Once the build succeeds, select **Start Without Debugging** from the Debug menu. Enter fifteen grades, and verify that the students appear beside the appropriate letter grade. Run the program several times to ensure that the output is correct regardless of how many students earn each particular grade and the order in which the grades occur.

Step 8: Add Inline Comments and Submit Your Code

Make sure that your code has **three or more inline comments** near the statements that you believe warrant the additional description (especially those that pertain to arrays). These can be simple line comments at the end of a line of code that needs explanation, or block comments where each precedes a series of statements described in the comment.

In Blackboard, return to the Assignments page for the current assignment. Under the **Submission** section, click **Add Attachments**. Click **My Computer**; then, click the first **Browse…** button. Navigate to where you saved the source code file **FLast_Lab08.cpp** (with your name), select it, and then click **OK**. Verify that your file appears in the Submission section, and click the **Submit** button at the bottom of the page.

If you do not click the Submit button, your attachment will not be saved!