CIS 236 – Programming in C

Program:	**Week 07**
Points:	**20**
Chapters:	**5, 6**

Description

Add sorting to your homework from week 6.

Using the results of homework 6, display the levels array in descending order (high to low).

There is extra credit available for this assignment.

Please be sure to read the section titled “Additional Information to Keep in Mind.”

Learning Objectives

In this assignment, you will practice:

- Using an array to store related data
- Implementing the provided sort function

Requirements

1. See the requirements for homework 6. You will be adding sorting to that project, so finish that assignment first.
2. The version for homework 7 will add two functions: bubbleSort and swap. Both function definitions are provided to you. You will have to add the appropriate code to your existing homework 6 in order to use these two functions.
3. The bubbleSort function will be called from main. (See the requirements for main below.) **You will need to make one minor change to the bubbleSort function definition.** (See the requirements for the bubbleSort function below.) Be sure to place the bubbleSort function in the appropriate location in your program.
4. The swap function is only called from within bubbleSort. You do not have to change anything in the swap function. Be sure to place the swap function in the appropriate location in your program.
5. The output includes an additional display of the array, in descending order from the highest value to the lowest value.

Requirements for the bubbleSort Function

Purpose:	This function, as provided, sorts an integer array in **ascending** order.

Parameters:	1. the array to be sorted
Algorithm:	Make one change to the algorithm in this function so that it sorts in **descending order**. The change you need to make is literally only one character. Do not overthink this.
Return value:	None

Requirements for main

1. Add to a call to **bubbleSort** after the call to the display_levels function. Use the appropriate argument.
2. Call the **display_levels** function again to display the updated order of the array.

   NOTE: Do not worry that the count for a particular level is now associated with a different level the second time the output displays. It's ok. Really. This assignment is meant to be easy.

Sample Run 1

```
Player points (-1 to quit): 5
Player points (-1 to quit): 15
Player points (-1 to quit): 16
Player points (-1 to quit): 17
Player points (-1 to quit): 48
Player points (-1 to quit): 40
Player points (-1 to quit): 22
Player points (-1 to quit): 23
Player points (-1 to quit): 24
Player points (-1 to quit): 25
Player points (-1 to quit): 23
Player points (-1 to quit): 24
Player points (-1 to quit): 25
Player points (-1 to quit): 23
Player points (-1 to quit): 24
Player points (-1 to quit): 30
Player points (-1 to quit): 31
Player points (-1 to quit): 32
Player points (-1 to quit): 33
Player points (-1 to quit): -1

Before sorting:
T O T A L S
```

```
Level 1   1
Level 2   3
Level 3   9
Level 4   4
Level 5   2
Level 6   0

After sorting:
T O T A L S

Level 1   9
Level 2   4
Level 3   3
Level 4   2
Level 5   1
Level 6   0
```

Sample Run 2

```
Player points (-1 to quit): 5
Player points (-1 to quit): 15
Player points (-1 to quit): 25
Player points (-1 to quit): 35
Player points (-1 to quit): 45
Player points (-1 to quit): 55
Player points (-1 to quit): 37
Player points (-1 to quit): 24
Player points (-1 to quit): -1

Before sorting:
T O T A L S

Level 1   1
Level 2   1
Level 3   2
Level 4   2
Level 5   1
Level 6   1

After sorting:
T O T A L S

Level 1   2
Level 2   2
Level 3   1
Level 4   1
Level 5   1
Level 6   1
```

Sample Run 3

```
Player points (-1 to quit): -1

Before sorting:
T O T A L S

Level 1   0
Level 2   0
Level 3   0
Level 4   0
Level 5   0
Level 6   0

After sorting:
T O T A L S

Level 1   0
Level 2   0
Level 3   0
Level 4   0
Level 5   0
Level 6   0
```

Additional Information to Keep in Mind

One way to write a for loop is to include the loop control variable's declaration within the for header, like x in this example:

Example 1:

```
for (int x = 0; x < finalvalue; x++) {
   //loop body
}
```

If you use this approach, remember that the variable x is only in scope within that for loop. If the only reason for the existence of x is to control the loop, then this is not a problem.

If you need to use x after the loop for some other purpose, then you need to declare x above the loop, like this:

Example 2:

```
int x;
for (x = 0; x < finalvalue; x++) {
   //loop body
}
```

An additional thing to keep in mind is that the approach in Example 1 (declaring the LCV within the for header) is only valid in later versions of C. When you first install Dev-Cpp, it will not choose a particular version of C. This isn't usually an issue unless you want to use the style of for loop in Example 1 above. To change the setting in Dev-Cpp to make Example 1 work, these are the steps to follow:

1. In Dev-Cpp, click Tools in the menu, and then click Compiler Options.
2. The "General" tab should display first. Make sure there is nothing in the 2 boxes labeled "Add the following commands...". Uncheck the checkboxes next to them if they are checked.
3. Then, still in that same open window, click on the "Settings" tab.
4. On that tab, click below Settings on the "Code Generation" tab.
5. At the bottom of the list that displays, click the arrow to the right of "Language standard (-std)". Choose "ISO C99" from the list.
6. Click OK until you're back to your code and try to compile again.

Once you have set the standard to C99, you should not have to change it again, unless you reinstall Dev-Cpp.

EXTRA CREDIT – Maximum of 20 Additional Points

Your must meet all original requirements for this assignment to be eligible for additional extra credit points. See the submission requirements for instructions on how to submit the extra credit.

Use parallel arrays to keep the name of the level and its original count associated with each other during sorting. These arrays should be declared in main.

To add this logic, you will need to add some additional code to the bubbleSort function and the swap function.

You will also need to update the display_levels function

Your function calls in main will also have to change.

Extra Credit Sample Run 1

```
Player points (-1 to quit): 5
Player points (-1 to quit): 15
Player points (-1 to quit): 16
Player points (-1 to quit): 17
Player points (-1 to quit): 48
Player points (-1 to quit): 40
Player points (-1 to quit): 22
Player points (-1 to quit): 23
Player points (-1 to quit): 24
Player points (-1 to quit): 25
Player points (-1 to quit): 23
Player points (-1 to quit): 24
Player points (-1 to quit): 25
Player points (-1 to quit): 23
Player points (-1 to quit): 24
Player points (-1 to quit): 30
Player points (-1 to quit): 31
Player points (-1 to quit): 32
Player points (-1 to quit): 33
Player points (-1 to quit): -1

Before sorting:
T O T A L S

Level 1   1
Level 2   3
Level 3   9
Level 4   4
Level 5   2
Level 6   0

After sorting:
T O T A L S

Level 3   9
Level 4   4
Level 2   3
Level 5   2
Level 1   1
Level 6   0
```

Requirements for Full Credit on This Project

SUBMIT YOUR OWN WORK – Plagiarism is not tolerated in this course. Please review the section on the Academic Honor Code in the syllabus. I will not hesitate to drop you from this class if you submit a program that is plagiarized.

COMPLETE AND ACCURATE – Your program must compile, execute, and give accurate output.

FOLLOW ALL REQUIREMENTS ACCORDING TO THE INSTRUCTIONS – Follow the instructions as written for completing this project, even if you [think you] know a "better" way to do something.

COMMENTS – Include comments in your code. There must be a comment at the top of your program that includes your name, the program number, and a description of the program. There must be comments at each important step in your program that describes that step. Every variable must include a comment describing its purpose.

BEST PRACTICES – Follow best practices in C programming as discussed in class and in the textbook, including, but not limited to, appropriate use of white space, indenting, alignment, meaningful identifier names, etc. Points will be deducted for sloppy code that is hard to read, even if it works, so pay attention to these details.

SUBMIT ONLY .c SOURCE CODE – Pay attention to the file extension of the source code file you submit. I will deduct points for not following this requirement.

SUBMIT ALL FILES BEFORE THE DUE DATE – Submit your .c source code file to the dropbox for this assignment on Canvas before the due date. Do not submit executable files. Do not submit project files from an IDE. I will not accept links to online storage.

Submission of Extra Credit

The extra credit source code files must be completely separate source code files from the original assignment. The extra credit source code files must be clearly labeled as extra credit.

I will not grade extra credit if the original requirements are not met first. I will not grade extra credit if the source code files for the original requirements are not submitted as separate code files.

You can submit everything (original requirements and extra credit) all in one zip file, or as separate .c files.

Instructions for creating a Compressed Folder (zip file) are available here:
https://support.microsoft.com/en-us/help/14200/windows-compress-uncompress-zip-files