CS215: Introduction to Program Design, Abstraction and Problem Solving
(Spring, 2025)
Lab Assignment 7
(20 points)
Today's Date: Tuesday, February 25
Due Date: Sunday, March 9

The purpose of this lab assignment is

- to practice creating an array to organize a sequence of items
- to get familiar with the Visual Studio debugger

Problem Statement

Write a program to calculate the execution score for the artistic gymnastics. The execution score, determined by a six-judge Panel, begins at 10 and deductions are made for errors and faults in technique, execution and artistry/composition. Each judge independently determines his/her score. The highest and lowest scores are dropped, and the gymnast's Execution Score is the average of the remaining four judges' scores. For exam, the six judges gave the gymnast scores of 9.3, 9.2, 9.55, 9.4, 8.9 and 9.18 respective, the Execution Score for this gymnast is: (9.3+9.2+9.4+9.18) / 4 = 9.27.

The following list gives the requirements for your program:

- After the user inputs six valid scores, your program should display the original sequence in the order the scores are typed and also display the sequence of scores after dropping the lowest and highest scores (the original order of the scores should not be changed except for the two dropping scores), then display the final Execution Score for the gymnast.
- Your program should handle the case when the user doesn't input the valid score, such as a letter instead of a double-floating point number, or the score is not in the range [0.0, 10.0]. The user is allowed to try as many times as possible.
- Define a function **`void dropTWO(double scores[], int& size)`** that removes the lowest score and highest score from the array passed in as the first parameter. The order of the original scores is preserved. For example, if the original scores are: 9.3, 9.2, 9.55, 9.4, 8.9 and 9.18. After calling this function, the array should contain: 9.3, 9.2, 9.4 and 9.18 in this order.
- Define a function **`double final_score(double scores[], int size)`** that calculate the final Execution Score for a gymnast after dropping the lowest and highest scores.

The following are some examples of running your program: (Note that the blue part represents the user input, and "↲" represents the return key from the user input.)

Sample output 1:

```
Please enter your score for the gymnast:
Nine point three↲
Invalid score! Expecting a score in the range [0.00, 10.00]
Please enter your score for the gymnast:
```

```
34↲
Score is NOT in the correct range!
Please enter your score for the gymnast:
do you mean 9.3?↲
Invalid score! Expecting a score in the range [0.00, 10.00]
Please enter your score for the gymnast:
9.3 is that ok?↲
Please enter your score for the gymnast:
9.2 I guess↲
Please enter your score for the gymnast:
9.55↲
Please enter your score for the gymnast:
9.4↲
Please enter your score for the gymnast:
8.9↲
Please enter your score for the gymnast:
ten↲
Invalid score! Expecting a score in the range [0.00, 10.00]
Please enter your score for the gymnast:
9.18↲
The scores from the judges are:
9.30    9.20    9.55    9.40    8.90    9.18
The scores after dropping the highest and lowest scores:
9.30    9.20    9.40    9.18
Final Execution Score is: 9.27
```

Sample output 2:

```
Please enter your score for the gymnast:
8.97↲
Please enter your score for the gymnast:
9.0↲
Please enter your score for the gymnast:
9.7↲
Please enter your score for the gymnast:
9.25↲
Please enter your score for the gymnast:
9.10↲
Please enter your score for the gymnast:
9.95↲
The scores from the judges are:
8.97    9.00    9.70    9.25    9.10    9.95
The scores after dropping the highest and lowest scores:
9.00    9.70    9.25    9.10
Final Execution Score is:  9.26
```

Sample output 3:

```
Please enter your score for the gymnast:
9.9↲
Please enter your score for the gymnast:
8.1↲
Please enter your score for the gymnast:
9.5↲
Please enter your score for the gymnast:
9.5↲
Please enter your score for the gymnast:
9.5↲
Please enter your score for the gymnast:
9.5↲
The scores from the judges are:
9.90    8.10    9.50    9.50    9.50    9.50
The scores after dropping the highest and lowest scores:
9.50    9.50    9.50    9.50
Final Execution Score is:  9.50
```

Sample output 4:

```
Please enter your score for the gymnast:
67.5 is my guess↲
Score is NOT in the correct range!
Please enter your score for the gymnast:
10.5↲
Score is NOT in the correct range!
Please enter your score for the gymnast:
9.05↲
Please enter your score for the gymnast:
9.5↲
Please enter your score for the gymnast:
8.7↲
Please enter your score for the gymnast:
8.9↲
Please enter your score for the gymnast:
9.9↲
Please enter your score for the gymnast:
9.4↲
The scores from the judges are:
9.05    9.50    8.70    8.90    9.90    9.40
The scores after dropping the highest and lowest scores:
9.05    9.50    8.90    9.40
Final Execution Score is:  9.21
```

Sample output 5:

```
Please enter your score for the gymnast:
9.05↲
Please enter your score for the gymnast:
9.05↲
Please enter your score for the gymnast:
9.05↲
Please enter your score for the gymnast:
9.05↲
Please enter your score for the gymnast:
9.05↲
Please enter your score for the gymnast:
9.05↲
The scores from the judges are:
9.05    9.05    9.05    9.05    9.05    9.05
The scores after dropping the highest and lowest scores:
9.05    9.05    9.05    9.05
Final Execution Score is:  9.05
```

Debugging

After your program passes compilation, you can start to test your program by executing your program. This time we will use debugging tool to help you in the detection and correction of errors in your program.

1. After collecting the user inputs and before your program prints out the scores from the Judges, set a **breakpoint**.
   a. Click in the grey gutter to the left of your code in Visual Studio. A red ball will appear indicating there is a breakpoint at that line.
2. Now rebuild your program and run it with debugging turned on (**Debug → Start Debugging** or press **F5**). You may type six double floating-points numbers for your program's input as usual.
3. Once your program's execution reaches the breakpoint, you will be back in Visual Studio, but with a different layout of panes (the "debug view").
   a. In the center is your code. A yellow arrow (probably on top of the red ball) indicates the next line of code to be executed.
   b. On the right are panes with lists of variables in scope ("Autos" or "Locals") and functions currently being executed ("call stack" **Debug →Windows →Call Stack** or press **Alt-7**).
   c. Look through the various tabs in these new panes.
   d. *Note: the locations of windows may differ in some versions of Visual Studio or with some screen sizes.*
4. Now we'll step through the code one line at a time. Use the menu item **Debug → Step Over** or press **F10**.
5. Continue single-stepping the program a few times. Observe what happens to the variables as your program executes.

6. Try pressing **F11** (menu item **Debug → Step Into**) when you are on a line with a function call (calling **`final_score()`** function). What happens then? What happens to the call stack?
7. Use **Debug → Step Out** (**Shift-F11**) to finish debugging the current function. This command pauses the program again immediately after the current function returns.
8. Use **Debug → Continue** (**F5**, just like starting debugging) to continue until the next breakpoint.
9. Use **Debug → Stop Debugging** (**Shift-F5**) to terminate your program before it finishes executing.

Demonstration is not required for Lab7 since there is no Lab class during the Midterm Exam week. However, if you want to gain the Domo Bonus 3 points, you need to demonstrate Lab7 by the end of Lab7 class. Any demonstration after Lab7 class is not valid for bonus 3 points.

Submission

Open the link to Course Canvas page (https://www.uky.edu/canvas) and log in to your account using your LinkBlue ID and password. Please submit your **source code in a .cpp file** through link "**`Lab 7`**".

Grading (20 points + Bonus 3 points)

1. Attend the lab session or have a documented excused absence. (5 points)
2. Submit your program in Canvas. (15 points)
   - Include comments as specified in the lecture notes. (2 points)
   - Handle the user input as described in the requirement (3 points)
   - Correctly define the function `dropTWO()` (3 points)
   - Correctly define the function `final_score()` (3 points)
   - Correctly display both the sequence before and after dropping the lowest and highest scores, and preserve its order. (2 points)
   - Correctly generate the final Execution Score. (2 points)

Demonstrate your program to your TA and answer TA's questions by the end of Lab7 class. (Bonus 3 points)