PIC 10A Section 5 - Homework # 4 (due Friday, October 30, by 6 pm)

You should upload each .cpp (and .h file) separately and submit them to CCLE before the due date/time! Your work will otherwise not be considered for grading. Do not submit a zipped up folder or any other type of file besides .h and .cpp.

Be sure you upload files with the precise name that you used in your editor environment otherwise there may be linker and other errors when your homeworks are compiled on a different machine. Also be sure your code compiles and runs on Visual Studio 2019.

COMPUTING AN AVERAGE HOMEWORK GRADE

In this homework, you will write a program that can compute the average homework grade for a student, by choosing the top scores, with each homework equally weighted. An appropriate message will then be displayed based upon the homework average. *You should submit a single file* **hw_main.cpp** (the other files will automatically be included when compiling and testing your code).

The task may sound daunting at first, but you will be using a **HomeworkList** class that is already writtten. The various member functions it provides will do a lot of the more difficult parts of the work for you. What you will focus on in this homework is using a class, along with basic control flow.

You will need to download **HomeworkList.h** and **HomeworkList.cpp** and make them part of your workspace in Visual Studio before writing your own **.cpp** file. For quick reference, the relevant parts of the class interface are included on the last page of this homework.

You must use the **HomeworkList** class that is given to you in this homework. Using other logic to avoid it or writing your own class will result in a grade of zero.

Please refer to the syllabus for how the work will be graded. The program should execute in the following manner. You must make sure the output matches the format demonstrated perfectly.

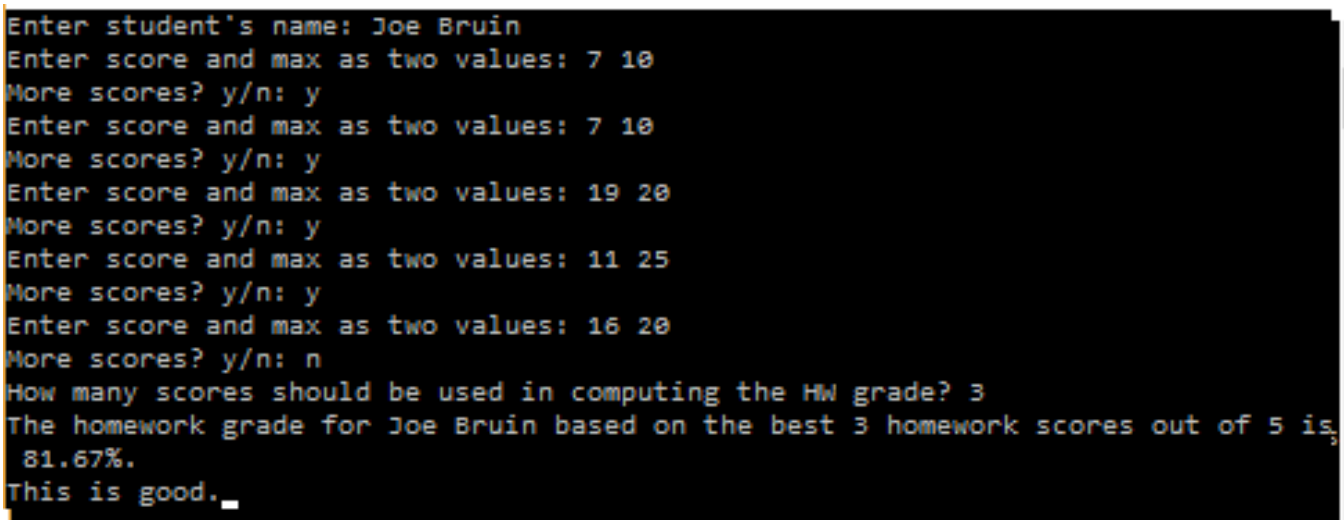
The program should execute in the following manner. You must make sure the output matches the format below perfectly.

Enter student's name: [USER ENTERS THE NAME]
[
UNTIL THE USER HAS NO MORE SCORES TO PROVIDE...
Enter score and max as two values: [SCORE IS GIVEN] [TOTAL POINTS ARE GIVEN]
More scores? y/n: [USER ENTERS 'y' OR 'n']
]
How many scores should be used in computing the HW grade? [USER ENTERS NUMBER]
The homework grade for [CORRECT NAME] *based on the best* [CORRECT NUMBER] *homework scores out of* [CORRECT TOTAL NUMBER OF HOMEWORKS] *is* [CORRECT PERCENTAGE DISPLAYED WITH EXACTLY 2 DIGITS OF PRECISION] *%.* [APPROPRIATE MESSAGE]

The [APPROPRIATE MESSAGE] is as follows:

- if the score is 95% or greater, "This is excellent.";
- if the score is greater than or equal to 80% but less than 95%, "This is good.";
- if the score is greater than or equal to 70% but less than 80%, "This is decent."; and
- otherwise the message should be "This is poor."

In the example given, the scores of 7/10 = 70%, 7/10 = 70%, 19/20 = 95%, 11/25 = 44%, and 16/20 = 80% can be stored in the **HomeworkList** data structure, without you needing to worry about how those values are stored. In addition, the average of the top number of homeworks, say top 3 (average of 95%, 80%, and 70%) can be given to you through the **HomeworkList** data structure without you having to worry about the internal logic of how to sort the scores and compute that top 3 average.


Enter student's name: Joe Bruin
Enter score and max as two values: 7 10
More scores? y/n: y
Enter score and max as two values: 7 10
More scores? y/n: y
Enter score and max as two values: 19 20
More scores? y/n: y
Enter score and max as two values: 11 25
More scores? y/n: y
Enter score and max as two values: 16 20
More scores? y/n: n
How many scores should be used in computing the HW grade? 3
The homework grade for Joe Bruin based on the best 3 homework scores out of 5 is
81.67%.
This is good.

```
/**
Constructor: creates a HomeworkList class storing no data
*/
HomeworkList();

/**
This function adds a homework score to the collection of homework grades. The
homework grades are all stored as percentages out of 100.

@param score the score on a given homework
@param maxPossible the maximum possible score for a given homework
*/
void addScore(int score, int maxPossible);

/**
This function will compute the overall homework percentage by averaging only the
 top number of scores specified.

@param topNumber the number of scores to include in the calculated average, for
 example if topNumber is 3, then the top 3 homework scores  are used to compute
  the overall homework average

@return the homework average: the best topNumber of scores among all scores present
*/
double percentageFromBest(size_t topNumber) const;

/**
This function returns the number of homework scores stored in the object.

@return the number of homework scores stored
*/
size_t numberOfHWs() const;
```