Assignment 3 – Structs, Methods & Arrays in C#

DNET 4200 – H. Kaeid

In this lab you will create a C# console program that processes an array of percentage grades and displays a report using a number of functions.

Note the following:

Grade Range	Letter Grade	Description
90 to 100	A+	Outstanding
85 to <90	A	Exemplary
80 to <85	A-	Excellent
75 to <80	B+	Very Good
70 to <75	B	Good
65 to <70	C+	Satisfactory
60 to <65	C	Acceptable
55 to <60	D+	Conditional Pass
50 to <55	D	Conditional Pass
<50	F	Failure

Conversion Functions

~~You will write two methods: **PercentToLetterGrade( )** and **PercentToDescription( )**. Both should take a double argument representing a student's percentage grade and both should return a string. PercentToLetterGrade( ) returns the corresponding letter grade string. PercentToDescription( ) returns the corresponding description string. If the percentage grade is not between 0.0 and 100.0, the string returned should be **"INVALID"**.~~

~~Define a delegate called **PercentToFeedback( )** which takes a double argument and returns a string. This will be used later as a reference to either PercentToGrade( ) or PercentToDescription( ) based on command line arguments.~~

Array Processing

~~You will create a data structure (struct) called **GradeStats**. It will include a public double to hold the average grade, and three public integers to hold the number of passing grades, the number of failures, and the number of invalid marks.~~

~~Write one array processing function called **CalculateGradeStats( )** that will take an array of doubles representing a list of marks and a GradeStats struct to hold the average, pass count, fail count, and invalid count. Do not include invalid grades in the average. This function will return the total number of grades processed.~~

~~Array Output~~

~~Write a function called ShowGradeReport( ) that takes an array of doubles representing a list of marks and a variable of type PercentToFeedback (the delegate you created). Call the GradeStats( ) function to~~

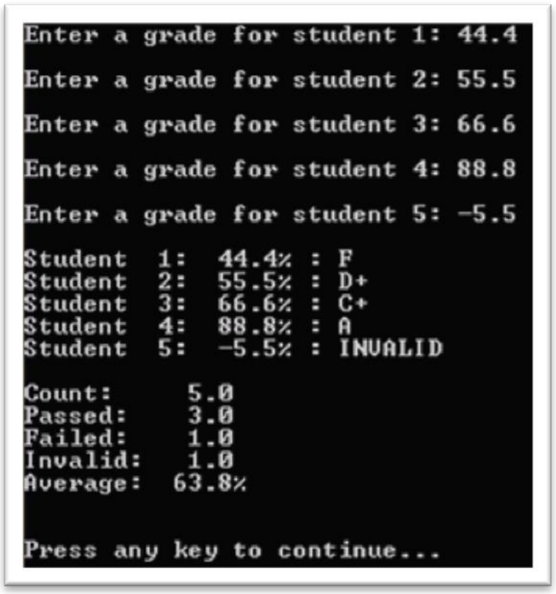
~~do all the required processing. Output each of the marks followed by the feedback string returned from PercentToFeedback( ). Once all the marks are displayed, show the overall count, average, number of passes, number of fails, and number of invalids. Use the output examples as a guide.~~

Main

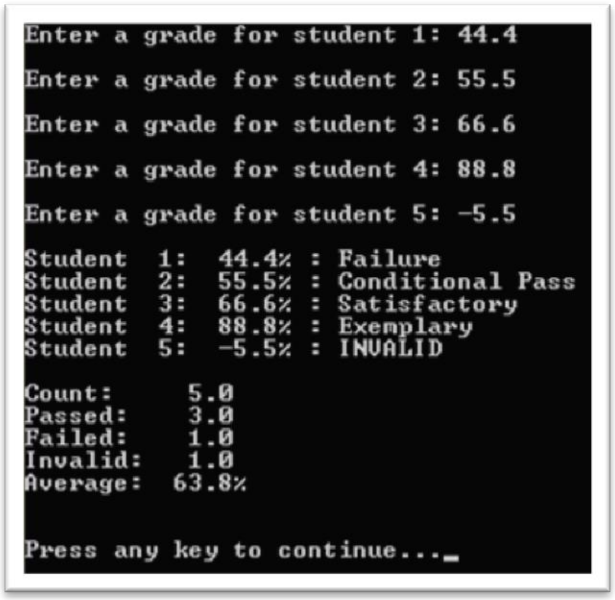
In Main( ), do the following

- ~~Declare an array of 5 doubles.~~
- ~~Declare a delegate variable of type PercentToFeedback and initialize it with the PercentToLetter method.~~
- ~~Use a loop to prompt the user for each of the 5 grades. Use exception handling to ensure only numeric input is allowed.~~
- In a try block, change the PercentToFeedback variable to PercentToDescription if the first command line argument is “description”.
- The following catch block is acceptable: `catch (Exception) { ;}`
- Call ShowGradeReport( ), passing the array and the PercentToFeedback variable.

Output Samples



```
Enter a grade for student 1: 44.4

Enter a grade for student 2: 55.5

Enter a grade for student 3: 66.6

Enter a grade for student 4: 88.8

Enter a grade for student 5: -5.5

Student  1:  44.4% : F
Student  2:  55.5% : D+
Student  3:  66.6% : C+
Student  4:  88.8% : A
Student  5:  -5.5% : INVALID

Count:      5.0
Passed:     3.0
Failed:     1.0
Invalid:    1.0
Average:   63.8%


Press any key to continue...
```



```
Enter a grade for student 1: 44.4

Enter a grade for student 2: 55.5

Enter a grade for student 3: 66.6

Enter a grade for student 4: 88.8

Enter a grade for student 5: -5.5

Student  1:  44.4% : Failure
Student  2:  55.5% : Conditional Pass
Student  3:  66.6% : Satisfactory
Student  4:  88.8% : Exemplary
Student  5:  -5.5% : INVALID

Count:      5.0
Passed:     3.0
Failed:     1.0
Invalid:    1.0
Average:   63.8%


Press any key to continue..._
```

General Requirements

- Include an opening comment with your name, the name of the program, the date, and a short description.
- Follow the style guide! Use descriptive names and sensible data-types for solutions, projects, variables, constants, arrays, methods, etc. that follow our naming conventions. Use good spacing and make sure braces ({}) are located where they are supposed to be.
- Utilize comments to explain the purpose of any code block (example a comment before a certain loop or a method call, another example is comments within the block of a method). Your comments should also explain any data type decisions and / or algorithms.
- Attach your entire solution folder zipped to the assignment drop-box.
- Submit only *one* zipped folder and name it as follows: Lastname_firstname_assignment 3