## Structures, Arrays, Functions

This programming assignment uses many of the ideas presented in sections 3 through 10 of the course notes, so you are advised to read those notes carefully as well as the following program specification.

## The Program Specification: Student Grades

In a certain class, three programming assignments and three tests are given. Your task is to compute average scores and overall grades for each student, as well as class averages and standard deviations as described in the output file description below.

## Input file description and sample:

Your program **must** read its input from a file named **`student.data`** — use of another input file name will generally result in a score of zero. The first two lines of the input file are preceded by an appropriate label and specify:

- The weight given to the program grade
- The weight given to the test grade

The third line is blank. The fourth line contains column labels, which should be ignored. The remaining lines of the input file will each contain:

- A string representing a student ID
- 3 integers representing program scores
- 3 integers representing test scores

Each input line will be terminated by a newline character. The data from the input file must be stored in an array of structures. The student ID field of the structure must be a character array. You may assume that there will be no more than 20 students. You may assume that all of the input values will be given in the specified order, but you may **NOT** assume that they will be logically correct. You must determine whether a score is valid; a score in the range 0 – 100 is considered valid.

Here is an example input file:

```
Program weight: 0.4
Test weight:    0.6

   ID       P1  P2  P3  T1  T2  T3
225299003   95 100  90  57  78  68
223191606  100  95 100  50  35  24
212949649  100 100 100  81  87  82
229254144   85 180 100  53  41  68
229295898   95 100 100  76  83  80
212043396   97 100 100  88  72  66
224020957  100  95  90  72  55 -12
138748208   95 100  90  78  42  56
```

## What to Calculate:

Your program must output a table showing all of the scores for each student in the input file. For each student, you must also calculate the student's average program score, average test score, overall grade, and overall letter grade. The table format is shown in the output section below.

## Output description and sample:

Your program must write its output data to a file named **`grades.out`** — use of any other output file name **will** result in a score of zero. A sample output file produced from the sample input file above is shown below:

```
Programmer: Donald Allison
Student Grades

Student          Programs                Tests          Overall  Letter
  ID        P1   P2   P3   Avg     T1   T2   T3   Avg    Grade   Grade
-------------------------------------------------------------------------
225299003   95  100   90  95.0     57   78   68  67.7     78.6     C
223191606  100   95  100  98.3     50   35   24  36.3     61.1     D
212949649  100  100  100 100.0     81   87   82  83.3     90.0     A
229254144              Invalid data in input file
229295898   95  100  100  98.3     76   83   80  79.7     87.1     B
212043396   97  100  100  99.0     88   72   66  75.3     84.8     B
224020957              Invalid data in input file
138748208   95  100   90  95.0     78   42   56  58.7     73.2     C
-------------------------------------------------------------------------
Averages                 97.6                   66.8     79.1
Standard Deviations       1.9                   15.8      9.8
```

As usual, the first line of your output should identify you by name, as shown. The second line should include the title "Student Grades" only. The third line should be blank. The fourth, fifth, and sixth lines should contain the specified column labels and a row of delimiters to mark the top of the table.

Next your output file will contain a table, with a line of output for each student. Each line should contain the student ID. If all of the scores for the student are valid, print the three program scores, the average of the program scores, the three test scores, the average of the test scores, the overall grade, and the corresponding letter grade. If any of the scores are invalid, print "Invalid data in input file". The table should be terminated by a row of delimiters.

After the table, print the class averages of the program averages, test averages, and overall grades. You will also print the standard deviations of the program averages, test averages and overall grades. Include **only** those students whose data is valid in your calculations of the class averages and standard deviations.

The standard 10 point scale should be used for determining the letter grades: ≥ 90 is an 'A'; ≥ 80 is a 'B'; ≥ 70 is a 'C'; ≥ 60 is a 'D'; < 60 is an 'F'.

The formula for standard deviation is as follows:

$$\sigma = \sqrt{\frac{\sum_{i=1}^{N} (X_i - \mu)^2}{N}}$$

where N is the number of students with valid scores, σ is the standard deviation, Σ is the sum from 1 to N, $X_i$ is the average for one student, and μ is the class average.

An example of calculating the standard deviation of the program averages is as follows:

$$\sigma = \sqrt{\frac{(95.0-97.6)^2 + (98.3-97.6)^2 + (100.0-97.6)^2 + (98.3-97.6)^2 + (99.0-97.6)^2 + (95.0-97.6)^2}{6}}$$

$$= 1.9$$

You are not required to use the exact horizontal spacing shown in the example above, but your output must satisfy the following requirements:

- You must use the specified header and column labels, and print a row of delimiters before and after the table body, as shown.
- You must arrange your output in neatly aligned columns. Use spaces, not tabs to align your output.
- You must use the same ordering of the columns as shown here, and print all real numbers with precision one.

## Programming Standards:

You'll be expected to observe good programming/documentation standards. All the discussions in class about formatting, structure, and commenting your code should be followed.

**Documentation:**

- Your header comment must describe what your program does.
- You must include a comment explaining the purpose of every variable or named constant you use in your program.
- You must use meaningful identifier names that suggest the meaning or purpose of the constant, variable, function, etc.
- Precede every major block of your code with a comment explaining its purpose.
- Precede every function you write with a header comment, explaining in one sentence what the function does.
- You must use indentation and blank lines to make control structures like loops and if-else statements more readable.

**Coding:**

- Use named constants instead of variables where appropriate.
- Use `double` variables for all averages and standard deviations.
- Write and use at least two user-defined function, not counting `main()`. Provide a valid C++ prototype for each function you write, excluding `main()` of course. The function `main()` must be the first function defined in your source code file.

## Testing:

Obviously, you should be certain that your program produces the output given above when you use the given input file. However, verifying that your program produces correct results on a single test case does not constitute a satisfactory testing regimen.

At minimum, you should test your program on **all** the posted input/output examples given along with this specification. The same program that will be used to test your solution generated those input/output examples. You could make up and try additional input files as well; of course, you'll have to determine by hand what the correct output would be.