CMSC 216 Project #1 Spring 2016

Grade Calculator **Due: Wed Feb 24, 8:00PM**

1 Overview

For this project you will write a C program that reads assignment scores and computes numeric grades and statistical information. There are two deadlines associated with the project. Those deadlines are:

- Wed Feb 17, 8:00PM - Your code must pass the first two public tests (public01, public02). That is the only requirement for this deadline. We will not grade the code for style. This first part is worth .5% of your course grade (NOT .5% of this project grade). Notice you can still submit late for this part.
- Wed Feb 24, 8:00PM - Final deadline for the project. Notice you can still submit late (as usual).

1.1 Obtaining project files

To obtain the project files, copy the folder project1 available in the 216 public directory to your 216 directory.

You need to copy the project1 folder we have provided as it has a file (.submit) that will allow you to submit your project. Notice that the project description can be found in the project_descriptions directory.

2 Specifications

2.1 Input Data

Your project will read information about class assignments and compute a numeric score. The data provided consists of:

- Number of assignments
- Points penalty per day late
- Number of assignments to drop
- Whether statistical information will be generated
- Assignments information (assignment number, score, weight, days late).

The data format is:

```
Points_Penalty_Per_Day_Late Drop_N_Lower_Value_Assignments Stats_Y/N
Number_of_Assignments(n)
Assignment Info #1
Assignment Info #2
...
Assignment Info #n
```

Each assignment info entry has the following information: assignment's number, assignment's score, assignment's weight (percentage), days late (integer). The following is an example of the data your program will process:

```
10 2 Y
4
2, 82, 40, 1
1, 91, 40, 0
4, 84, 10, 3
3, 73, 10, 3
```

For the above data, there is a 10 points penalty per day late, the 2 lower scores need to be dropped, statistical information will be generated (Y), and a total of four assignments are provided. For assignment number 2 the student's score is 82, the assignment represents 40% of the student's grade and it was submitted 1 day late.

2.2 Processing

Your program will compute the numeric score after dropping the n lowest scoring assignments and taking into account days late, penalty per day late, and the weight associated with the assignments. If statistical information is requested, the mean and standard deviation will be computed. For example, for the above data, your program is expected to generate the following output:

```
Numeric Score: 81.5000
Points Penalty Per Day Late: 10
Number of Assignments Dropped: 2
Values Provided:
Assignment, Score, Weight, Days Late
1, 91, 40, 0
2, 82, 40, 1
3, 73, 10, 3
4, 84, 10, 3
Mean: 65.0000, Standard Deviation: 18.2346
```

Regarding the data and processing:

1. Use double as your floating-type (e.g., double tmp, double numeric_score).
2. The assignment number will be a value between 1 and the maximum number of assignments. You can assume valid assignment numbers are provided.
3. Assignments can be provided in any order; however they must be printed in order (by assignment number).
4. An assignment score will be an integer value between 0 (inclusive) and 100 (inclusive). You can assume we will provide valid scores.
5. The weight will be an integer value between 0 (inclusive) and 100 (inclusive). You need to check the that sum of weights for all assignments add to a 100. If after reading the data the total weights do not add to a 100, your program will generate the error message **ERROR: Invalid values provided** and the program will terminate. The message should be printed on a line by itself.
6. Your program should remove the n lowest valued assignments before performing any numeric score computation. We define value as an assignment's score $\times$ the assignment's weight. For a total of x assignments, we will provide a value of assignments to drop that is in the inclusive range (0 .. x - 1). Notice that number of days late and the penalty per day WILL NOT be used in order to decide what assignment to drop. If two assignments have the same value (score $\times$ weight) the one with the lowest assignment number will be dropped.
7. The numeric score will be a value between 0 (inclusive) and a 100 (inclusive). For the numeric grade computation, adjust the score for an assignment based on the number of days late and the points penalty per day late. An assignment score will be set to 0 if the assignment's score becomes less than 0 after the late penalty is applied. This adjusted score along with the assignment's weight will allow you to compute the numeric score.
8. If any assignment is dropped, the sum of assignment weights will, nearly always, not correspond to a 100.
9. Either 'Y' or 'y' will request statistical information. Any other character will indicate that no statistical information will be generated.

10. For the computation of the mean and standard deviation you need to apply the late penalty, but do not drop any assignments (even if there was a assignment drop request). In addition, do not use weights for the computation of mean and standard deviation.

2.3 Functions Requirements

1. You must have at least two other functions in addition to main.
2. One of your functions must take at least one array as a parameter.

2.4 Other

1. Use `%5.4f` as the format for a float.
2. The maximum number of assignments in the input is 50.
3. You may not use C structures for this project.
4. You may not use global variables. If you define any global arrays you will lose a significant percentage on your grade.
5. You may not use two-dimensional arrays.
6. To use the sqrt function you need to follow the instructions at the top of "man sqrt" on grace. In particular, note the lines involving `#include` and Link. (Hint: gcc grades.c -lm , assuming gcc has been aliased with the flags described below.)
7. You must name your C file grades.c otherwise it will not compile on the submit server.
8. You may not use the qsort function.

2.5 Compilation

Your code must compile using (at least) the following gcc flags:

```
-ansi -Wall -g -O0 -Wwrite-strings -Wshadow -pedantic-errors -fstack-protector-all
```

2.6 Execution

We will use input and output redirection in order to provide data to your program. For example, assuming data is present in the public01.in file, we will run your program as follows: a.out < public01.in. You can compare the results of your program against expected results by using the diff command.

3 Grading Criteria

Your project grade will be determined with the following weights:

Results of public tests	20%
Results of secret tests	70%
Code style grading	10%

3.1 Style grading

For this project, your code is expected to conform to the following style guidelines:

- Your code must have a comment at the beginning with your name, university ID number, and UMD Directory ID (i.e., your username on Grace).
- No lines longer than 80 columns are allowed. You can check your code's line lengths using the `linecheck` program in grace. Just run "`linecheck filename.c`" and it will report any lines that are too long.
- Use reasonable and consistent indentation.

- Use descriptive and meaningful identifiers.
- Do not use global variables.
- Feel free to use helper functions for this project. Just make sure to declare them as static.
- Each function must have, at a minimum, a comment describing its purpose and operation. If you use a complicated algorithm to implement a function, you definitely need an extra comment explaining the complicated steps of your algorithm.
- Follow the C style guidelines available at:

  http://www.cs.umd.edu/class/spring2016/cmsc216/content/resources/coding-style.html
- A standard deviation calculator can be found at:

  http://www.mathsisfun.com/data/standard-deviation-calculator.html

4 Testing

Make sure you test your code with different input data sets (sets different from the ones we have provided as public input). You can take one of the provided input files (e.g., public01.in), update it with different values, and use input redirection to generate output. Notice you will need to manually check your results (you may not compare your results with the results of another student's code). To come up with test cases read the project description carefully. It is best if you think of test cases as you implement your project.

5 Submission

5.1 Deliverables

For this project, the only file that we will grade is `grades.c` (which **must** be the name of your source file).

5.2 Procedure

You can submit your project by executing, in your project directory (project1), the **submit** command. This will prompt you for your UMD Directory ID and password, and if all goes well, inform you of a successful submission. You should then log onto the submit server (there is a link on the course website) and check your public test results to be sure that things worked as you expected.

You need to execute submit in the project1 directory, as we gave you a .submit file that allows you to submit. If you did not copy the project1 folder we have provided, you will not be able to submit (you will be missing the necessary .submit file).

Immediately after copying the project1 folder, try to submit your project (even if you have not started). Do not wait until the day the project is due in order to try the submission process.

5.3 Possible problem with submit command

If you try to submit your project in grace, and you get the error:

"Exception in thread 'main' java.lang.OutOfMemoryError: unable to create new native thread"

then close all terminals windows except one, and try to submit again.

6 Academic Integrity

Please see the syllabus for project rules and academic integrity information. All programming assignments in this course are to be written individually (unless explicitly indicated otherwise in a written project handout). Cooperation between students is a violation of the Code of Academic Integrity.