UMBC
AN HONORS UNIVERSITY IN MARYLAND

CMSC 201 Section 60
Fall 2020

Project #1
Due date:
Design Document - Wednesday, October 28 2020 at 11:59:59 pm
Completed Project - Monday, November 2, 2020 at 11:59:59 pm
Value: 100 points

Collaboration: For Project 1, **collaboration is not allowed** – you must work individually. You may still see your TA and come to office hours for help, but you may not work with any other CMSC 201 students.

Note:

Make sure that you have a complete file header comment at the top of your file and that all information is correctly filled in.

```
# File:    FILENAME.py  (OR: FILENAME.txt)
# Author:  YOUR NAME
# Date:    THE DATE
# Section: YOUR DISCUSSION SECTION NUMBER
# E-mail:  YOUR_USERNAME@umbc.edu
# Description:
#    DESCRIPTION OF WHAT THE PROGRAM DOES
```

Project 1 is the first assignment where you've had to turn in a "design document" in addition to the actual code. The design document is intended to help you practice deliberately constructing your program and how it will work, rather than coding as you go along or starting without a plan.

Instructions

UMBC
AN HONORS UNIVERSITY IN MARYLAND

For this project, you will be creating a single program, but one that is bigger in size and complexity than any individual homework problem. This assignment will focus on using functions to break a large task down into smaller parts.

You are ***required*** *to follow the design details provided to you*! You may add additional functions, but you must implement all the specified functions as described in the Design Information section below.

At the end, your Project 1 file must run without any Python errors.
It must also be called proj1.py (case sensitive).

Additional Instructions – Creating the proj1 Directory

During the semester, you'll want to keep your different Python programs organized, organizing them in appropriately named folders (also known as directories).

You should create a directory in which to store your Project 1 files. We recommend calling it `proj1` and creating it inside a newly-created directory called `Projects` inside your `201` directory.

If you need help on how to do this, refer back to the detailed instructions in Homework 1.

Objective

The goal of Project 1 is to demonstrate a mastery of:

- Program design and the use of functions
- Reading from and writing to files
- Using and manipulating lists

In addition, you will demonstrate an understanding of the other control structures and language artifacts that we have used so far this semester.

This includes strings, floats and ints; casting variables from one type to another; and using basic arithmetic operators.

Task

You will write a program that computes descriptive statistics on CMSC course grades so that the Computer Science Department can determine whether CMSC 201 is being taught at an appropriate level of difficulty. You will be given four data files, representing the course grades for students in each of four semesters. You will read each data file into a separate two-dimensional list. You will calculate the mean and median of each class, and write the results of your analysis to a new data file. You must write this program following an approved design method, using functions as described in this instruction sheet.

Test Data:

There are four data files, each representing course grades from an imaginary class of CMSC 201 students. The files, which are available from the class github site, are:

Fall_2020.txt
Spring_2021.txt
Fall_2021.txt
Spring_2022.txt.

Each file begins with two header lines that were inserted when the files were created. After that, there is one student record per line, with entries on a line separated by white space. Your program must read each file into a separate two-dimensional (2D) list.

Each line of data contains:

- The student last name, a string
- The student first name, a string
- The student ID, a string
- The student's grade on Project 1, an integer
- The student's grade on Project 2, an integer

- The student's grade on Project 3, an integer
- The student's grade on Test 1, an integer
- The student's grade on Test 2, an integer
- The student's grade on Test 3, an integer.

Coding Standards

Prior to this assignment, you should re-read the Coding Standards, available on the Project 1 GitHub repository at https://github.com/Al-Arsenault201/Fall_2020_Project1.

For now, you should pay special attention to the sections about:

- Comments
  - ***File header comments***
  - ***Function header comments***
  - Note that "Input" and "Output" in the function header comment do NOT mean what is shown on the screen with `print()` or what is gotten from the user with `input()`.
    They refer to the **parameters** taken in, and the **return value**.
    (Both "Input" and/or "Output" may be None, if appropriate.)
- Constants
  - You must use constants instead of "magic numbers" or strings!!!
- Make sure to read the last page of the Coding Standards document, which prohibits the use of certain tools and Python keywords. (Note that you can use the built-in `sqrt()` function and `sort()` method – only – for this project.)

Additional Specifications

For this assignment, **you must follow the design overview** in this document.

For this assignment, you may assume that the data are valid. All integers are positive and meaningful; there will be no textual data where there are supposed to be numbers.

UMBC
AN HONORS UNIVERSITY IN MARYLAND

The project is worth a total of 100 points. Of those points, 10 will be based on your design document, 10 will be based on following the coding standards, and 80 will be based on the functionality and completeness of your project.

Design Document

The design document will make sure that you begin thinking about your project in a serious way early. This will give you experience doing design work and help you gauge the number of hours you'll need to set aside to complete the project. **Your design document must be called design1.txt.**

For Project 1, the design overview is presented in the Design Information section below, and you are **required to follow it**. For future projects, you will be creating the design entirely on your own and may choose to design it however you like.

Your design document must have four separate parts:

1. A file header, similar to those for your homework assignments
2. Constants
    a. A list of all the constants your program will need, including a short comment describing what each "group" of constants is for
3. Function headers
    a. A complete function header comment for each function
4. Pseudocode for main()
    a. A brief but descriptive breakdown of the steps your main() function will take to completely solve the problem; note function calls under relevant comments (if applicable).

Although you will be presented with a design overview, you must still create the function headers, and the pseudocode for the main program on your own.

A start for your design is provided on the class GitHub site at https://github.com/Al-Arsenault201/Fall_2020_Project1**. It is called design_fall_20.pdf**

NOTE: The sample design provided is not complete and is missing many function header comments. You need to add them! You may also add constants if you find it necessary.

Your **`design1.txt`** file will be compared to your **`proj1.py`** file. Minor changes to the design are allowed. A minor change might be the addition of another function or a small change to **`main()`**.

Major changes between the design and your project will cause a loss of points. This would indicate that you didn't give sufficient thought to your design.

(If your submitted design doesn't work, it is generally better to lose the points on the design and to have a functional program, rather than turning in a broken program that follows the design. The ultimate decision is up to you.)

Design Information

You are required to implement and use at least the following seven functions for Project 1, in addition to the main program. You may implement more functions if you think them necessary, but the functions below must be implemented **and used**. The information for each function is given below.

Helper Functions

- **`def mean(values)`**
  - Computes the mean of a set of numeric values
    - **`values`** is a list of numeric (integer, floating point, or mixed) values
  - Returns the mean of the values
  - Preconditions:
    - **`values`** contains >= 1 value
  - Note that this function does not display the mean. It simply computes and returns it.

- **`def median(values)`**
  - Finds the median of a set of numeric values
    - **`values`** is list of numeric (integer, floating point, or mixed) values
  - Returns the median of the values
  - Preconditions:
    - **`values`** contains >= 2 values
  - Note that this function does not display the median. It simply finds and returns it.
  - You will need to sort the list of values from smallest to largest to find the median. Use the Python built-in **`sort()`** method, as you did in Homework 5, Part 2. Don't forget to make a local copy of **`values`** before sorting!

- **`def stdev(values)`**
  - Computes the standard deviation of a set of numeric values
    - **`values`** is a list of numeric (integer, floating point, or mixed) values
  - Returns the floating point standard deviation of the values
  - Preconditions:

        - **`values`** contains >= 2 values
    - Note that this function does not display the standard deviation. It simply computes and returns it.
    - The standard deviation computation requires the use of a square root. Use the Python built-in **`sqrt()`** method. Use it as follows:

        `math.sqrt(`*`value`*`)` where ***`value`*** = a numeric value

    - ❖ **HINT**: Take advantage of the **`mean()`** function that you have written!

General Functions

These are the "heavy lifters" of the program, and do the majority of the work, and are almost always called from **`main()`**. They often call other functions, often more than once. You must implement and use all the general functions given below.

- **`def read_file(filename)`**
    - Reads in the contents of a text file of values
    - Takes in a filename as a string
    - Returns the file contents, except for the first two rows, as a single string
    - The first two lines of every data file are HEADER rows and do not contain student scores. This function should read in these two lines and essentially throw them away, as they are not needed.
    - The rest of the file contains: student name, student id number, and six scores – three for projects and three for tests.

- **`def create_lists (file_contents)`**
    - Takes the string representing the actual data from a file

  - Returns two lists: one list containing the sum of the three project grades for each student; and the other containing the sum of the three test grades for each student

- **`def calc_values (project_list, exam_list)`**
  - Calculate the mean, median, and standard deviation of the sum of the project scores and the sum of the exam scores. You do NOT need to calculate these values for EACH project score or each test score.
  - Use the helper functions mean, median, and stdev to do this
  - Create a list containing in this order: project mean, project median, project standard deviation, test mean, test median, , test standard deviation
  - Join the elements of that list with a tab to create a single string, and return that string. That is, you'll have a statement that looks something like
    result_string = "\t".join(results_list)

- **`def write_file (list)`**
  - Takes as input the list of strings containing the values from calc_values
  - Writes a single text file containing explanatory header information, and then the calculated values for each of the four classes

Main Program

Start your main program by creating a list containing the four data file names.Then execute a for each loop, going through the four files in that list. In that loop, you process a file by first calling the read_file function; then passing the resulting string to the create_lists function. Then you call the calc_values function.

When the loop has finished, you call the “write_file” function to print your output.

The main program should contain ***minimal*** code!! There should be no more code than is necessary to make the function calls described above.

Submitting

Once your **`proj1.py`** or **`design1.txt`** file is complete, it is time to turn it in with the **`submit`** command. (You may also turn the design and project in multiple times as you reach new milestones. To do so, run **`submit`** as normal.)