University of Western Ontario, Computer Science Department
CS1026A SU19, Computer Organization

Assignment 3 **Due: May 31, 2019**

General Instructions: This assignment consists of 4 pages, 1 exercise, and is marked out of 100. Assignments are the independent work of each student. Software may be used to detect cheating.

Non-Functional Code Instructions:

1. Include brief comments in your code. Identifying yourself (the code's author) by name and user ID in the initial comment header. Comment also on key instructions and calculations in your code.
   e.g.
   ```
   ##
   # A program for parsing CSV files.
   # Student Name:  Alex Brandt
   # Student ID: abrandt5
   ```

2. Follow *good coding style* and normal Python conventions. This includes, but is not limited to:
   (i) Meaningful variable names.
   (ii) Conventions for naming variables and constants.
   (iii) Use of constants over "magic numbers".
   (iv) *Readability*: indentation, appropriate white space (blank spaces) within instructions, consistency in the use of all of the above.

Evaluation:

1. Functional Requirements:
   (i) Does your module correctly implement the six statistical functions?
   (ii) Does your main program behave according to the specifications?
   (iii) Does your main program handle invalid input?
   (iv) Does your main program output the files according to the specifications?

2. Non-Functional Requirements (above).
3. Ability to follow directions precisely.

Submission Instructions: Your submission should include exactly two files (zipped into a zip file, if you'd like). These two files are: `myStatistics.py` and `userid_main.py` where userid is replaced by your UWO User ID (everything preceding "@" in your UWO email; e.g. abrandt5). The contents of these two files are described below.

Learning Outcomes: In this assignment we will look at using functions, dictionaries, lists, and file I/O.

Exercise 1. This problem deals with *parsing numerical data* and performing simple *statistical analysis*. Imagine this scenario. Your chemistry lab-mate has collected many measurements of an experiment and has put them all into a single file for you. This file is named `A3-data-file.txt` and is posted on OWL alongside these instructions. It is now your job to analyze the results. In this experiment there were four distinct trials and you must perform the analysis on each trial individually. Unfortunately, your lab-mate has mixed data from different trials together! Fortunately, each measurement has a *label* to indicate of which trial it is a part. For example, the first few lines of the data file are:

```
trial1 123.43
trial3 341.32
trial2 123.42
trial4 89.337
trial3 355.12
```

Therefore, your program must perform the analysis of the data in three steps:

1. Read the data in the file and sort each measurement based on which trial it is a part.
2. Perform the statistical analysis on each trial.
3. Write the statistical analysis of each trial to new file (i.e. write out four different files).

To perform these three steps your program should be broken into two parts.

Part 1. In this section we will define the contents of the `myStatistics.py` file. In this file we look to implement six functions for statistical analysis: `myMin`, `myMax`, `myAverage`, `myMedian`, `myStandardDeviation`, `myCountBins`.

For all functions do not use Python's statistics package nor the built-in `min`, `max` functions. You must implement the math yourself.

`myMin` is a function which takes as its only parameter a list of floating point values and returns the minimum value among all values in the list.

`myMax` is a function which takes as its only parameter a list of floating point values and returns the maximum value among all values in the list.

`myAverage` is a function which takes as its only parameter a list of floating point values and returns the average of all values in the list.

`myMedian` is a function which takes as its only parameter a list of floating point values and returns the median of the values in the list.

`myStandardDeviation` is a function which takes as its only parameter a list of floating point values and returns the standard deviation of the sample of values. Standard deviation (σ) can be computed by the following formula:

$$\sigma = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}\left((x_i - \bar{x})^2\right)}$$

where x_i are the individual values in a list of n values, and $\bar{x}$ is the average of the list of values.

`myCountBins` is a function which takes two parameters: a list of floating point values, and a floating point number. This second parameter is the *bin size*. This function will implement a simplified form of *data binning* https://en.wikipedia.org/wiki/Data_binning. This function will go through the list of values given as the first parameter to count the number of values in the list which fall into a certain "bin". The bins are defined as: $0 \le x_i < \textit{bin size}$, $\textit{bin size} \le x_i < 2 \times \textit{bin size}$, $2 \times \textit{bin size} \le x_i < 3 \times \textit{bin size}$, ... until all values in the input list have been found to exist in a certain bin. For example, if the maximum value in the input list is 30 and the *bin size* is 10 then there should be 4 bins: $0 \le x_i < 10$, $10 \le x_i < 20$, $20 \le x_i < 30$, $30 \le x_i < 40$.

All functions only need to handle lists of floating point numbers. That is to say, if you come across a non-number in the input list then your program **is** allowed to crash. The `myCountBins` function only needs to handle **non-negative numbers** in its input list. All other functions must handle all possible floating point numbers.

Part 2. In this section we define the contents of the `userid_main.py` file. In this file you shall `import` your other file (e.g. by the command `from myStatistics import *`) and then use those imported functions within your `main` function to prompt the user for the name of the data file, read the data in that file, and then output the results of the analysis to four different files. In particular, your main function should:

1. Prompt the user to input the name of the file which contains the data to analyze.
2. Open the file for reading, if possible. If the file is not found or not available for any reason, simple print an error message "Sorry, the file `<filename>` is not available" and then terminate the program.
    - *Hint:* use `try:  except:` to accomplish this.
3. Read all of the data in the file and separate the data into four different lists, one list for each trial.
    - *Hint:* A dictionary of lists would be very helpful here!
4. For each trial compute, using your `myStatistics` module,
    - the minimum,
    - the maximum,
    - the average,

- the median,
- the standard deviation, and
- the list of bin counts **for a bin size of 25**.

5. For each trial we wish to output the computed data to the files `trial1-data-analysis.txt`, `trial2-data-analysis.txt`, `trial3-data-analysis.txt`, and `trial4-data-analysis.txt` where trial1 data goes in the file `trial1-data-analysis.txt`, etc. The data should be output in the following format where each item inside angled brackets (e.g. `<minimum>`) is replaced by the actual value computed for that trial. All numbers should be printed using 5 digits after the decimal place, except for `bin_count` which can simply be the string representation of the list of counts.

   ```
   minimum  : <minimum>
   maximum  : <maximum>
   average  : <average>
   median   : <median>
   std_dev  : <standard deviation>
   bin_count: <list of bin counts>
   ```

   An example output file can be found on OWL next to these assignment instructions but is also repeated here as an example. `trial1-data-analysis.txt` should have the following contents:

   ```
   minimum  : 0.47074
   maximum  : 398.21285
   average  : 207.06971
   median   : 209.04432
   std_dev  : 115.91112
   bin_count: [16, 12, 16, 15, 18, 19, 16, 13, 13, 12, 29, 11, 19, 14, 19, 19]
   ```