CS 1313 010: Programming for Non-Majors, Fall 2014
Programming Project #5: Big Statistics
Due by 11:20am Wednesday Nov 5 2014

This fifth programming project will give you experience writing programs that involve `for` loops and arrays. This programming project will use the same development process as in Programming Projects #2, #3 and #4, and will be subject to the same rules and grading criteria, along with some new criteria.

This specification will be less detailed than for previous programming projects. **You are expected to know how to perform basic tasks without having to be told explicitly,** based on your experience with previous programming projects.

To get full credit on this programming project, you **MUST** use `for` loops and dynamically allocated arrays appropriately.

I. PROJECT DESCRIPTION

You've been hired to develop statistics software. Specifically, your software will calculate various statistics, some of which you saw in PP#3, as well as something new (described below).

In each individual run of your software, you will input two lists of numbers, and these two lists will have the same length, a length that will be input at runtime just before inputting the lists.

For each of the two lists of numbers, you will need to calculate the following statistics: mean, bias-corrected sample variance, bias-corrected standard deviation, and standard error, as in PP#3.

In addition, the two lists of numbers together will be used to calculate the *correlation coefficient*, specifically the *Pearson product-moment correlation coefficient* [1].

[1] http://en.wikipedia.org/wiki/Pearson_product-moment_correlation_coefficient

II. PROGRAM DESCRIPTION

Write a program to calculate the statistics from input data. The body of the program **MUST** be broken into **THREE** subsections, rather than the usual four subsections (there **WON'T** be a greeting subsection):

1. an input subsection;
2. a calculation subsection;
3. an output subsection.

Because of how data will be input (see below), **THERE WON'T BE A GREETING SUBSECTION.**

You are **ABSOLUTELY FORBIDDEN** to have:

- **ANY** calculations in the input subsection (and the only outputs should be idiotproofing error messages);
- **ANY** inputs or outputs in the calculation subsection;
- **ANY** inputs or calculations in the output subsection.

That is, the three subsections **MUST BE COMPLETELY SEPARATE,** and **MUST BE CLEARLY LABELED.**

For this project, `if` blocks can occur in any subsection.

A. ARRAY DECLARATIONS

You **MUST** use **DYNAMIC** memory allocation and deallocation for **ALL** arrays. Therefore, **ALL** arrays **MUST** be declared as **POINTERS.** For example:

```
float* input_variable1 = (float*)NULL;
```

B. INPUT SUBSECTION

The program will take its input from a data file, rather than from a user typing live at the keyboard (see section III, **INPUT DATA FILES**, below).

The input data will be in the following format:

1. a single length, which is shared by both of the lists of numbers (for example, if a length of 5 is input, then the first list will have length 5 and the second list will also have length 5).
2. for each element in the two lists:
   (a) the value of that element of the first list;
   (b) the value of that element of the second list.

Several such data files will be provided, each representing an individual run. **YOU** should determine how to input the data **BY EXAMINING THE INPUT DATA FILES** (see **HOW TO FIND AND EXAMINE THE INPUT DATA FILES**, below).

Because of how the data will be input, **YOU WON'T PROMPT THE USER FOR THE INPUTS** (see **HOW THE DATA WILL BE INPUT**, below).

You **MUST** store the input data in appropriate one-dimensional arrays. You are **ABSOLUTELY FORBIDDEN** to use multidimensional arrays in PP#5.

C. ALLOCATING ARRAYS

You **MUST** use **DYNAMIC** memory allocation and deallocation for **ALL** arrays. Therefore, **ALL** arrays **MUST** be declared as **POINTERS. Note that ALL of the arrays MUST be allocated, at runtime, in the execution section, IMMEDIATELY AFTER INPUTTING AND IDIOTPROOFING THE LENGTH OF THE ARRAYS.** In other words, once you have input and idiotproofed the length of the arrays, you **MUST IMMEDIATELY** allocate both of the arrays. After allocating each array, the program **MUST** check whether the array was allocated successfully, and if not, the program **MUST** output a suitable, **UNIQUE** error message and then **MUST EXIT.**

For details, see the lecture slide packet "Array Lesson 2," slides 26-33.

D. IDIOTPROOFING

YOU MUST IDIOTPROOF ANY input that needs idiotproofing, to make sure that it has an appropriate value. **YOU** are responsible for figuring out all of the possible cases of idiocy that could come up. You should idiotproof each value **immediately after it is input. ALL IDIOTPROOFING MUST BE COMPLETED BEFORE ANY CALCULATIONS ARE PERFORMED;** that is, idiotproofing belongs in the input subsection.

Note that, for this programming project, you are **ABSOLUTELY FORBIDDEN** to use `while` loops for your idiotproofing; that is, upon detecting idiocy, the program **MUST EXIT.**

Idiotproofing error messages **MUST** be clear, complete English sentences that **COMPLETELY AND UNAMBIGUOUSLY** state the nature of the error. Thus, **EACH ERROR MESSAGE MUST BE UNIQUE.** For example, an error message might be:

```
ERROR: You cannot have a list length of -3.
```

E. CALCULATION SUBSECTION

In the calculation subsection, the program **MUST** calculate:

- for each list of numbers:
  1. the mean
  2. the bias-corrected sample variance
  3. the bias-corrected standard deviation
  4. the standard error.
- for the combination of the two lists:
  1. their *Pearson product-moment correlation coefficient* (described below).

In any `for` loop in the calculation subsection, you **MUST** calculate **EXACTLY ONE** kind of result; that is, you are **ABSOLUTELY FORBIDDEN** to calculate multiple kinds of results in a single `for` loop.

For example, the `for` loop that calculates the mean of the first list **CANNOT** also calculate the mean of the second list, nor the variance of the first list.

However, within a particular `for` loop, you may choose to calculate temporary scalar variables representing various subexpressions.

In your program, you are **ABSOLUTELY FORBIDDEN** to use variable names such as `x_bar`, `v` and `s`, because they violate the favorite professor rule.

Note that statistics such as mean, variance and standard deviation cannot be guaranteed to have integer values, and in real life very rarely are integers.