Exercise 5

Structures and Records: Standard Deviation on Record

Introduction:

Until this point, the programs that you've created would only deal with processing information sent by the source code or the keyboard input. Yet, a program may also acquire its information from a file on a disk. For this exercise, the source of your information come from a text file that is accessed sequentially.

Note that when reading a string data, the `getline()` function would be used, and it has the newline as the default delimiter. For example,

```
      string str;
      getline(inRec, str, '\t ');//reading a string with a tab delimiter
```

The standard deviation is acquired by the positive square root of the **variance**. The variance computed for intended for sample data is designated by s2.

$$s^2 = \frac{\sum(x-\mu)^2}{n-1}$$

x - element in a population
μ – mean (average)
n – size of the population

The **standard deviation** is the most commonly used measure of spread (dispersion). It indicates how compactly the values of a data set are clustered all around the mean.

Learning Outcomes:

- Read data on a sequential file and place it on a structure.
- Compute the standard deviation of the record and write the result on another sequential file.

Problem background

1. Create a text file named "StudentScore.txt" on the notepad with the following content.

```
John        99
Caleb       79
Timothy     82
Ruth        85
Hannah      78
Peter       92
Joshua      88
```

   Note: The delimiter between the student name and the score is a tab stop (\t) while the delimiter between the score and the student name is a new line (\n). Be sure that there is no stray character at the end of your record.

2. On your C++ source code, define a structure as follows.

```
struct student
{
```

```
                        string studentName;
                        double score;
                     };
```

3. Implement the functions below:

```
/* get the data from the text file and store it on the array of student
structure. Read a single record first then use a looping structure controlled
by an  ifStreamObj.eof() function to read all of the content of the text
file.
*/
void readFromRecord (student studScoreRec[])

/*
 to compute the average score on the studScoreRec[] array with size s.
*/
double recAverage (student studScoreRec[], int s)

/*
to compute the standard deviation of the record.
*/
double recSTDev(student studScoreRec[], int s)

/*
to write to a sequential text file named “scoreDescStat.txt”  with the
average and standard deviation as content.
*/
void writeResultToFile (double ave, double stDev)
```

4. Call all the functions created to function main