Data mining your health

Due: Thursday, Feb. 6th

Collecting experimental data in remote locations is changing. In the "old school", researchers would record their measurements in the field with pencil and paper and then type-in those data once they returned to their labs. Today, mobile and/or embedded devices with data collection and analysis software allow scientists to collect their data in real time and immediately perform preliminary analyses on those data.

PURPOSE

To provide practice with: (0) simulating the collection of real time measures of health, (1) translating algebraic formulas into computational equations, (2) calling mathematical library functions, (4) using selectional control with the IF-ELSE statement, and (5) performing an elementary statistical analysis based on the obtained data.

You are to write a C++ program that determines whether there exists a LINEAR relationship between a patient's body temperature (°F) and that same patient's respiratory rate (BPM, breaths per minute). Each pair of (temp, breathRate) values (recorded at the same time) constitute an (x,y) pair. So in this example, the exact "meaning" of one (x,y) pair is:

(*your body temperature (°F), your respiratory rate* (BPM))

where *x* is your temperature and *y* is your respiratory rate at this particular time.

It should be relatively easy for you to imagine a real-world situation where we might want to investigate the relationship between two types of values. For example, if you were interested in whether plant height and location are correlated (that is, you want to know if there is a relationship between the plant height and location of those plants in the forest), the x-values could represent plant height and the y-values could represent location. Likewise, you might want to know if (x) baby length (at birth) is strongly correlated with (y) adult height.

METHOD

The problem is be handled as follows: First, your program must ask for three (3) REAL (x,y) pairs. That is, you'll need to input three pairs of values; for example:

```
double x1, y1; // hold first (temp, respriatoryRate) pair
x1 = getBodyTemperature_F();       // function returns value
y1 = getRespirationRate_BPM();     // function returns value
```

(NOTE: This type of correlational work is typically done with *many* more (x,y) data pairs than three, however, we will work with three in this assignment to keep the problem relatively simple). Given the three (x,y) pairs, a CORRELATION COEFFICIENT (often referred to as the variable, **r**) can then be calculated using the formula shown on the next page. The CORRELATION COEFFICIENT (r) is a statistical measure to indicate whether there is a LINEAR relationship between sets of values.

$$r = \frac{n\sum x_i y_i - \sum x_i \sum y_i}{\sqrt{n\sum x_i^2 - (\sum x_i)^2}\ \sqrt{n\sum y_i^2 - (\sum y_i)^2}}.$$

where

n	is the number of (x,y) pairs
$\sum x_i$	is the sum of the x values
$\sum y_i$	is the sum of the y values
$\sum x_i^2$	is the sum of the squares of the x values
$\sum y_i^2$	is the sum of the squares of the y values
$\sum x_i y_i$	is the sum of the products of corresponding x and y values

Having calculated r, you should then calculate **r^2** to obtain what is commonly called the STRENGTH of ASSOCIATION. This value is a number characterizing the joint distribution of two variables. Think of r^2 as a more interpretable form of r.

Having calculated the STRENGTH of ASSOCIATION, your program should then determine if the strength of association is "weak" or "strong"; that is, how strongly does X explain or predict Y. For example, in a baby length example, we want to know: "Does baby length REALLY predict adult height?" If the STRENGTH of ASSOCIATION (the square of r) is greater than or equal to (>=) 0.5, then we will agree that the association is "strong enough." So, if the association value is greater than or equal to 0.5, your program should print a message which indicates this is so; otherwise, your program should print a message which indicates the association is "weak."

INPUT
All body temperature and respiratory rate are obtained from the keyboard (standard input or stdin). These inputs are your simulation of "real" values.

*Note: if you want to **test** your program with your own values (and of course you do!), you can of course (temporarily) explicitly set (x,y) values with assignment statements so you don't have to keep inputting six values over and over as you debug your code. **Just make sure you submit code that requires input values.***

OUTPUT
Your output should include (1) a title; (2) a summary (echo) of the three (x,y) pairs in the **(x,y)** coordinate format; (3) the correlation coefficient (r) and the strength of association values, r^2. Of course, the output should also print a message indicating a "strong" or "weak" relationship. See the last page for an example output based on a three (x,y) pairs of values.

PROGRAM DETAILS
Your program must have an initial comment box, giving your name, date, and program number. A second comment box must include the purpose of the program (that is,

what does it do); a description of the INPUT that the program uses (e.g. three pairs of values from the stdin); and a description of the program's OUTPUT (indicate *where* the output appears). You MUST have these comments placed in the comment boxes as described in class.

- You MUST comment EACH variable. Variables MUST have good names.
- You MUST use constants where appropriate. Constants should be named using all UPPERCASE letters.
- Note: a more "real world" scenario would collect the three sets of data points at various times, not one right after another. But here, you can just collect the three body temperatures and three respiratory rates one after another.
- All output is to be formatted to as shown in the example output.
- Since the formulas require division, your program MUST trap division by zero in ALL cases. If the data values entered would result in division by zero, your program must output an ERROR message and stop executing. **When can this happen? You must determine when and trap all those cases!**
- Along with your folder of code, make a (text) file called: README.txt. This file should explain the assignment (briefly) and list the names and contents of each file. At the bottom, please tell me the status of your program, for example: *"All code tested and working fine.", or "Almost works, correlation value appears wrong? Not sure why"*, etc.
- When your code is fully tested (is it *really tested?)*, make a .zip of your folder and submit *only* that .zip file via the onCourse site. Remember: the program is due on Thur., Feb. 6th (so you really can submit it up to early Friday morning at 4am, but no later: onCourse will turn off for this submission after 4am). *Always* submit something, even if it doesn't work.

Mark's SAMPLE OUTPUT BASED ON 3 sets of values

- **All text in** Courier **is the actual output from my program**
- **All explanations in {*italics BOLD*} are not output from my program**

```
******************************************************
         Regression analysis on three pairs of
          (BodyTemp, RespirationRate) values
******************************************************
```

{Next my program asks to collect three sets of vital signs. Once collected I echo pairs to the console.}

```
     Your three (BodyTempF, RespirationRate) pairs are:
             ( 98.6,  18.0)
             ( 97.4,  17.0)
             ( 96.8,   7.0)
```

{ I test for "bad" input that might cause division by zero here, but all looks ok so I continue ...}
{ "Crunch, crunch, crunch" (that's my program computing r and r-squared);
now I'll print those out..... }

```
     ------------------------------------------------
            correlation coefficient =    0.80
            strength of association =    0.63
     ------------------------------------------------
```

{Now I'll use an if-else statement to check if the strength of association is "strong" or "weak"; I'll also print the 0.50 constant so a reader of this output will know how I reached the conclusion}

```
     STRONG r-squared (means r^2 is 0.50 or greater)
```

Superior Effort:
As you'll note in the Starter Kit, the two functions that return your vital signs are defined in `vitalSigns.cpp`. At the start, this data is "generated" from the keyboard, that is, the person running the program must enter the values.

For a superior grade, alter the functions to produce random numbers that return realistic, but randomly generated body temperatures and Breaths Per Minute (BPM) values. For example, the body temperatures should be somewhere between 95°F (or is even this unreasonably low?) and 105°F. Do the research to establish a reasonable range of values for each item.