ACMS 20210 Assignment 4

Due: 11:59 PM on October 12, 2018

Submit the C++ programs below using Sakai. If you are compiling from the command line on the CRC or another Linux system, I suggest using the command

```
g++ your_program_name.cpp -std=c++1y -o your_executable_name
```

This tells the compiler (called g++) to compile your .cpp source file into an executable file. To run this executable file from the command line, type

```
./your_executable_name
```

If you do not yet know how to use the CRC and do not have a compiler on your local machine, I suggest using the website *cpp.sh* to test your code (be sure to save a copy on your local machine, though!). You must submit your code in separate .cpp files.

You may always assume that the user enters a reasonable value (e.g. if the problem states the user enters a positive integer, you can assume that they enter a positive integer).

Use the following naming scheme for submitting your solutions on Sakai: For exercise n on homework m, the name of your submission should be hw_m_ex_n.cpp. As an example, for homework 1 exercise 3, the name of your submission should be: hw_1_ex_3.cpp.

1. Write a program that repeatedly prompts the user to enter real numbers, storing these numbers in a vector of doubles, ceasing to prompt the user when the user enters a string that does not represent a real number. The program then calls two functions, each of which accepts a vector of doubles, passed by constant reference. One of the functions should return the maximum entry in the vector, and the other function should return the minimum entry. Report back to the user the maximum and minimum values found.

2. Write a program that repeatedly prompts the user to enter a sequence of first names and corresponding (integer) scores for an exam. (The prompt should alternate between names and scores, or prompt for both on the same line). The program terminates when one of three possibilities occurs: a duplicate name has been entered, or the string "EndInput" is given for a name, or a score outside the set of integers 0 to 100 inclusive is entered. Your program should ignore the name and score of whatever input was used to terminate the entering data phase of the program. For this program, use vectors to store the names and scores (even though there are better ways that we have yet to discuss).

   The program then reports the highest and lowest scores on the exam, along with the first names of all of the people who obtained each of these scores (in case of ties).

3. In this exercise, we will write a program that performs a simple statistical analysis, including finding a least squares linear regression line.

   Write a program that prompts the user to enter a finite sequence of alternating x and y real number values (or to enter a string for an x value or y value to stop entering input). I recommended storing these values in vectors *x_obs* and *y_obs*, which should ultimately be of the same length. These values will serve as a sample of x and y valued observations on which to perform our statistical calculations. If the number of (x, y) pairs entered by the user is 0 or 1, the program should terminate with a message indicating the sample is too small to be useful.

   After prompting the user to enter these x and y values, your program should then call a function to compute the (sample) correlation r, using the formula

$$r = \frac{1}{n-1} \sum_{i=1}^{n} \left(\frac{x_i - \bar{x}}{s_x} \right) \left(\frac{y_i - \bar{y}}{s_y} \right)$$

   where $\bar{x}, \bar{y}$ are the sample means of x and y respectively and s_x and s_y are the sample standard deviations of x and y respectively, where the sample consists of the data $(x_1, y_1), \ldots, (x_n, y_n)$ (be careful of discrepancies in indexing between the mathematical notation and the C++ implementation!). Use this formula for your calculation, even if you know a more efficient formula. Your function for the correlation should use pass by constant reference for the vectors of observations, and it should return the value of the correlation.

   Your program should then calculate the least squares regression line, given by $\hat{y} = mx + b$ where $m = r\frac{s_y}{s_x}$ and $b = \bar{y} - m\bar{x}$, provided $s_x \neq 0$ and $s_y \neq 0$. Report the following information to the user: the sample means of x and y, the sample standard deviations of x and y, the sample correlation, and the least squares regression line. You should account for the possibility that s_x is 0, in which case you should report that the correlation and least squares linear regression line are undefined. If $s_y = 0$ but $s_x \neq 0$, report that the correlation is undefined, but that the regression line is the interpreted to be the horizontal line given by the only y value in the sample.

4. In this exercise, we are going to approximate Brun's Constant. A twin prime is a positive prime number p for which at least one of $p+2$ and $p-2$ is also prime. The first few twin primes are $3, 5, 7, 11, 13, 17, 19, 29, 31$. It is unknown whether or not there are infinitely many twin primes. However, regardless of whether or not there are infinitely many twin primes, the sum of the reciprocals of the twin primes is known to converge. Brun's constant differs from the sum of the reciprocals of the twin primes by $\frac{1}{5}$.

   Write a program that prompts the user for a positive integer N and computes the sum

$$S(N) = \sum_{\substack{2 \le p \le N \\ p \text{ and } p+2 \text{ are prime}}} \left(\frac{1}{p} + \frac{1}{p+2} \right).$$

   $S(N)$ is an approximation for Brun's constant, which is defined to be

$$\begin{aligned} B &= \sum_{\substack{2 \le p \\ p \text{ and } p+2 \text{ are prime}}} \left(\frac{1}{p} + \frac{1}{p+2} \right) \\ &= \left(\frac{1}{3} + \frac{1}{5} \right) + \left(\frac{1}{5} + \frac{1}{7} \right) + \left(\frac{1}{11} + \frac{1}{13} \right) + \cdots \end{aligned}$$

   Be careful to include all the appropriate twin primes in your sum. Report your approximation to the user to 12 decimal places. The true value of Brun's constant is thought to be approximately 1.902160583104 (approximated using twin primes below some large threshold).

5. For any positive integer n, the totient function $\phi(n)$ is the number of positive integers not exceeding n that are relatively prime to n. For $n = 1$, $\phi(n) = 1$. The first few values (starting at $n = 1$) for $\phi(n)$ are: $1, 1, 2, 2, 4, 2, 6, 4, 6$.

   Suppose n is a positive integer with the unique prime factorization

$$n = p_1^{e_1} \dots p_m^{e_m}$$

   where the p_i are the distinct prime factors of n. It can be shown then that

$$\phi(n) = \left(p_1^{e_1} - p_1^{e_1-1}\right) \cdots \left(p_m^{e_m} - p_m^{e_m-1}\right).$$

   Write a program that prompts the user for a positive integer N and computes the summatory totient function $S(N) = \sum_{k=1}^{N} \phi(k)$.