ACMS 20210 Assignment 4

Due: 11:59 PM on Tuesday April 9, 2017

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
```

1. Rewrite Problem 3 (the statistical analysis problem) from the previous homework so that, instead of prompting the user for input, the program takes a single command line argument containing the name of a file. The file will contain two doubles on each line, separated by a space, with the first representing an x value and the second representing a y value. Use these values to form your vectors of x and y observations. The program should report all of the same analysis as in the previous homework to the user. You can assume that the file contains at least two lines of observations.

   See the appropriate folder for a sample of what the input might look like.

2. Write a program that takes an integer N and a filename for output using command line arguments. The program then uses Euler's method to approximate the solution of the differential equation

$$\begin{aligned} y' &= y \\ y(0) &= 1 \end{aligned}$$

   over the interval $[0, 1]$, dividing the interval into N subintervals. Write the corresponding (x_i, y_i) values to a file, with one pair of (x_i, y_i) values per line, separated by a space (with no parentheses). Use setprecision(12) for writing doubles to your output file.

   See the accompanying file for an example of what the output file should look like for $N = 8$.

See the last page of this document for a brief explanation of Euler's method if you do not recall it (or did not see it!) in your calculus classes.

3. Write a program that takes an arbitrary number of file names as command line arguments and counts the frequency of each letter in the English alphabet (A through Z) appearing among all the files. Ignore all characters that are not part of the basic English alphabet. You should regard upper and lower case of the same letter as the same. Report the count of each letter, along with its overall frequency (as a percentage of total letters) to the user. You do not have to worry about letters with diacritical marks (e.g. umlauts) or other variants from the standard alphabet.

4. Write a program that takes a command line argument for the name of a file. The program then reads two matrices (whose entries are integers) from the file, with both matrices guaranteed to be of the same dimensions. The matrices will be separated from each other in the file by a blank line. The entries in each row of the matrix will be separated by a blank space, with no characters other than spaces and those forming the integers appearing on each separate line. The program then outputs the sum of the two matrices to a file named "problem_4_solution.dat", with each row of the matrix on a separate line in the file, and the entries of each row in the matrix separated by a space. (Remember that addition of matrices is defined by adding corresponding elements.) I suggest using the type vector<vector<int>> to store each of your matrices, though you could use dynamically allocated memory as well.

   See the appropriate folder for a sample of what the input might look like.

1 Euler's Method

To approximate the solution of the differential equation

$$\begin{aligned} y' &= f(x, y) \\ y(0) &= y_0 \end{aligned}$$

over the interval $[0, b]$, divide the interval $[0, b]$ into N distinct subintervals, each of equal length $h = b/N$. The subintervals will have endpoints $[x_{i-1}, x_i]$, where $x_0 = 0$, $x_N = b$ and $x_i = b/N$. Let y_i denote the approximate solution at the point x_i, i.e. $y_i \approx y(x_i)$. To compute the values y_i, we use an iterative procedure.

Take y_0 to be the initial value $y(0)$. For $i = 1, \ldots, N$, define iteratively

$$y_i = y_{i-1} + f(x_{i-1}, y_{i-1})h$$

This procedure is just repeated use of the approximation

$$y(x + h) \approx y(x) + y'(x)h = y(x) + f(x, y(x))h$$

using values at the former iteration for x_i and y_i.