Fall 2022

CPSC2620 Assignment 4

Due: November 25 (Friday), 2020 11:55pm

(30 marks)

Please note that due to the potential COVID situation, the requirements of this assignment might be changed and also the compilation and submission procedure might be changed. You will be notified should a situation arise.

Notes:
(*) Please note that no collaboration is allowed for this assignment. Should any plagiarism be identified, all the involved students will get zero for the assignment and will be reported to the Chair of the department for further action. The marker has been asked to check your submissions carefully.
(*) Your submission will be marked on the correctness and readability (including comments) of your program.
(*) Please follow the guidelines to name your files (see below).
(*) Late submission policy: 1 day late: 5% off, 2 days late: 10% off, more than 2 days late: you will get 0 for the assignment.

NEW NEW NEW

Many Thanks to Nicole, who created and tested the Makefile and submission script for our assignments.

Please be advised that you have a new procedure to compile and submit your source code for your assignments. The details are provided in the README.txt file, which you have used for Assignment 3 already.

For this Assignment, replace N with 4 in the README.txt file.

Please use the provided Makefile to compile your program. Please see the REAME.txt. You don't submit your own Makefile.

Please follow the submission procedure in README.txt to submit your source code.

Please follow the requirement on the names of the source code files. They should have suffix .cc and .h. Other files are not accepted.

Notes:

(*) Files to be created for Question (1) are: *MatrixPro2.cc*, *MatrixPro2.h*, and *testMatrixPro2.cc* (which is used to test the class *CMatrixPro2* you create). The executable for this question is called *testMatrixPro2*. Please see below for more details.

(*) File to be created for Question (2) are: *FreqCalc.cc*. There is no header file for this question. The executable for this question is called *FreqCalc*. Please see below for more details.

Question (1) Let us continue making the semi-professional class CMatrixPro in Question 2 of Assignment 3. We will modify it in this question. Let's call the new class *CMatrixPro2*. All requirements regarding element indices are the same as before. Data members are the same as before. We have declared and implemented the following operators.

>> (implemented as a friend), >> (implemented as a friend), = (implemented as a member operator), + (implemented as member operators for both versions), - (implemented as member operators for both versions), * (implemented as member operators for both versions), > (implemented as a member operator), < (implemented as a member operator), == (implemented as a member operator).

Now, let us do more.

(a) Modify operator =. Now it returns CMatrixPro2& such that we can do cascading for operator =. Also we can assign a bigger matrix to a smaller one or assign a smaller matrix to a bigger one, by resizing the original matrix accordingly.
(b) Add operator +=. It returns CMatriPro2&. It also has two versions, as what we did for + in Assignment 3.
(c) Add operator -=. It returns CMatrixPro2&. It also has two versions, as what we did for – in Assignment 3.
(d) Add operator *=. It returns CMatrixPro2&. It also has two versions, as what we did for * in Assignment 3.

Create a test program called *testMatrixPro2.cc*. Your test should show that the member functions/operators of the matrix class work as expected. One good way to do this: create two 3 x 3 matrices, read elements in them, do difference arithmetic operations defined by the class between the two matrices or between one matrix and an integer using the corresponding operators. Each time you print the result out to verify (show) that your program works.

Question (2) Write a program called *FreqCalc.cc* that reads from keyboard a list of words and produces two lists for outputting. The first list you need to maintain contains the distinct words from the input as well as the number of times each word occurs. The words should be converted to lower case (write a helper function to convert a character to its lower case equivalent by using *transform* in STL). The second list you need to maintain is the list of (lower case) letters in the words from the first list and their frequencies. Ignore any non-letter characters.

You may assume that the words are separated by white spaces and are terminated by end-of-file, represented by CTRL+D.

The outputs from your program should be the same as what the following example shows.

Please input a list of words: adba computer sicence acdd Computer

Here is the summary:

Words	Frequencies
acdd	1
adba	1
computer	2
sicence	1

Frequencies	Words
2	computer
1	acdd
1	adba
1	sicence

Letters	Frequencies
a	3
b	1
c	5
d	3
e	4
i	1
m	2
n	1
o	2
p	2
r	2
s	1
t	2

```
u                    2

Frequencies  Letters
---------------------------------------
5            c
4            e
3            a
3            d
2            m
2            o
2            p
2            r
2            t
2            u
1            b
1            i
1            n
1            s
```

You can see that the first list is printed out after being sorted according to the words in the alphabetical order and then is printed again after being sorted according to the frequency from the highest to the lowest.

The same applies to the second list.

Hint: You may wish to use a map to build the word list. To get a sorted version of the same list, try to use *pair* data type. The same idea is also applicable to the letter list.