Assignment 2 — Using Maps

Due: Wednesday, September 23

Objective: The objective of this assignment is for students to work with and show they understand how to use associative containers (course objective 5 from the syllabus). *This program is shorter and less involved than others in this class, so this assignment will be graded out of 50 points — half the weight of other assignments.*

High-Level Description: In the first assignment you wrote a program to figure out word frequencies: how often a word appears in a text file. In this assignment you are to consider character frequencies, but also consider the notion of "context" — in this case the "context" is the preceding character, so rather than counting the number of times the character "`u`" occurs (for example), you will count the number of times "`u`" follows "`q`", how many times "`u`" follows "`r`", and so on.

Details: You are to write two programs for this assignment, `count1.cpp` and `count2.cpp`. Both programs read the input (from standard input) character-by-character, and update counts after each character is read. Use `cin.get()` for character input.

For `count1.cpp` you are simply to keep track of how many times each character occurs. This is a basic `map<char,int>` object that increments the count as each character is seen. At the end of the program, print out each character and the number of times it occurs.

For `count2.cpp` you are to keep a *separate* set of counts for each context — in other words, for each character "context" (meaning the previous character) you will have a complete set of counts. This is a map of maps — and part of this assignment is to figure out exactly what that means as a C++ `map` declaration, and to use it in your program. At the end of the program, print out each pair of characters and the count of the number of times that pair occurred.

See the sample input and output on the back of this handout and follow the basic form shown there for your program's output.

To Turn In: Use the `330submit` program (see Handout 3) in order to turn in your code, using assignment name `assign2`. You should submit all of your code (for both programs) so that it can be compiled and tested. On the due date, you should turn in a printout of your code.

Sample Input/Output:

SAMPLE INPUT

```
Whammer Jammer
```

SAMPLE OUTPUT (FROM COUNT1)

```
: 1
 : 1
J: 1
W: 1
a: 2
e: 2
h: 1
m: 4
r: 2
```

SAMPLE OUTPUT (FROM COUNT2)

```
( ,J): 1
(J,a): 1
(W,h): 1
(a,m): 2
(e,r): 2
(h,a): 1
(m,e): 2
(m,m): 2
(r,
): 1
(r, ): 1
```