University of Washington
Outreach Program

Name_____

1. We require a container, Histogram, that can be used to keep count of the number of alphas (letter or digit), digits (0..9), punctuation, and whitespace (\t, \v, \f, \r, or \n) characters in an input file.

   a. Develop the use cases for the Histogram. Don't forget that the use case has the graphical, textual, and exception condition components.
   b. Develop a class diagram for the Histogram.
   c. Implement the Histogram class. *Remember, a histogram is a container only; it stores things, it does not do any parsing.*
   d. Write a function parseAdd that takes a C-style string by value and an instance of the Histogram by reference. Invoke the function parseAdd to parse the string and update the Histogram entries. The string is to be read from a file.
   e. Write a short test plan stating *what* must be tested to verify your design.
   f. Based upon your test plan, write test cases and test your design using a file that contains several sentences. Ensure that the sentences contain characters in each category in the Histogram.

2. In addition to being able to individually access query the number of entries in any of the categories, overload the ostream operator to support printing instances of a Histogram to stdout.

3. Let's now explore a more open ended design problem to which we can apply our skills in the design of object centered systems.

   As the president of Big Bucks Consulting, as well as its chief designer, you have been hired by Fleecem National Bank to upgrade its customer database. You have chosen to rewrite the data base using C++.

   Each customer account consists of a savings account and a checking account.

   a. We begin the design by developing a set of use cases for the combined account that will describe how a customer may interact with the account and each of the pieces.

      Don't forget that the use case has the graphical, textual, and exception condition components.

b. As a next step, specify and describe, textually, the primary public and private components of a customer account. Express these in a class diagram. Don't get carried away here - there are only a couple of basic attributes of such an account.

c. Now the design and implementation – and test too. Design a C++ class that implements the requirements of parts a and b. First write the C++ code and classes that implement that design then integrate them into the final system.

d. Write a short test plan stating *what* must be tested to verify your design.

e. Based upon your test plan, write test cases and test your design, including boundary conditions, to ensure that it behaves as you (and the customer) intended.

   Be sure to include all the data and function members.