**The Assignment:**

You will implement and test a small class called statistician, which is similar to some of the small classes in Chapter 2 of the text.

**Purposes:**

Ensure that you can write a small class that meets a *precise* specification.

Make sure you understand how to write a class that is separated into a header file and an implementation file.

Give you experience in using a test program to track down bugs in a class's implementation.

**Before Starting:**

Read all of Chapter 2.

Know how to compile and run C++ programs on your system.

**Files that you must write:**

1. stats.h: The header file for the new statistician class. Actually, you don't have to write much of this file. Just start with [our version](http://www.cs.colorado.edu/~main/projects/stats.h)and add your name and other information at the top. If some of your member functions are implemented as inline functions, then you may put those implementations in this file too.
2. stats.cxx: The implementation file for the new statistician class. You will write all of this file, which will have the implementations of all the statistician's member functions.

**Other files that you may find helpful:**

1. [stattest.cxx:](http://www.cs.colorado.edu/~main/projects/stattest.cxx) A simple interactive test program.
2. [statexam.cxx:](http://www.cs.colorado.edu/~main/projects/statexam.cxx) A non-interactive test program that will be used to grade the correctness of your statistician class.

**The Statistician Class   
Discussion of the Assignment**

As indicated above, you will implement a new class called statistician, using a header file (most of which is written for you) and an implementation file (which you will write by yourself). The statistician is a class that is designed to keep track of simple statistics about a sequence of real numbers. There are two member functions that you should understand at an informal level before you proceed any further. The prototypes for these two functions are shown here as part of the statistician class declaration:

class statistician

{

public:

...

void next(double r);

double mean( ) const;

...

};

The member function "next" is used to give a sequence of numbers to the statistician one at a time. The member function "mean" is a constant member function that returns the arithmetic mean (i.e., the average) of all the numbers that have been given to the statistician.

Example: Suppose that you want a statistician to compute the mean of the sequence 1.1, 2.8, -0.9. Then you could write these statements:

// Declares a statistician object called s

statistician s;

// Give the three numbers 1.1, 2.8 and -0.9 to the statistician

s.next(1.1);

s.next(2.8);

s.next(-0.9);

// Call the mean function, and print the result followed by a carriage return

cout << s.mean( ) << endl;

The output statement will print 1.0, since 1.0 is the mean of the three numbers 1.1, 2.8 and -0.9.

Once you understand the workings of the next and mean member functions, you can look at the complete specification of the statistician class, which is in the file [stats.h](http://www.cs.colorado.edu/~main/projects/stats.h). Notice that the statistician class in this file is part of a namespace called main\_savitch\_2C. You should use this namespace for your statistician. In this file you will find a precondition/postcondition contract for all the statistician's member functions, including:

* A default constructor, which merely does any initialization needed for the statistician to start its work
* The next and mean functions, described above
* A constant member function called length, which returns the count of how many numbers have been given to the statistician
* Two constant member functions called minimum and maximum, which return the smallest and largest numbers that have been given to the statistician. (By the way, these two functions and the mean function all have a precondition that requires length( ) > 0. You cannot use these three member functions unless the statistician has been given at least one number!)
* A constant member function called sum, which returns the sum of all the numbers that have been given to the statistician. This function does NOT have a precondition. It may be called even if the statistician has NO numbers (in which case it should return 0).
* An overloaded operator == which tests to see whether two statisticians are "equal". The prototype is:
* int operator ==(const statistician& s, const statistician& t);

In order for two statisticians to be equal, they must have the same length (i.e., they have been given the same number of numbers). Also, if their length is greater than zero, they must also have the same mean, the same minimum, the same maximum, and the same sum. For example: Suppose that a statistician s has been given four numbers 1, 2, 3, 4. A second statistician t has been given four numbers 1, 1.5, 3.5, 4. Then the test (s==t) must return true since both s and t have equal values for all the member functions, as shown here:

* 1. s.length( ) and t.length( ) are both 4
  2. s.mean( ) and t.mean( ) are both 2.5
  3. s.sum( ) and t.sum( ) are both 10.0
  4. s.minimum( ) and t.minimum are both 1
  5. s.maximum( ) and t.maximum are both 4
* An overloaded + operator which has two statisticians as arguments, and returns a third statistician, as shown in this prototype:
* statistician operator +(const statistician& s, const statistician& t);

* An overloaded \* operator which allows you to "multiply" a double number times a statistician. Here is the prototype:
* statistician operator \*(double scale, const statistician& s);

This is not a member function. The result of a multiplication such as 2\*s is a new statistician that looks as if it had been given all the numbers of s, multiplied by the constant 2. Examples: Suppose that s is a statistician that has been given 1, 2, 3, and u is another statistician. Then the assignment statement u=2\*s will result in u behaving as if it had been given the numbers 2, 4, 6. As another example, the assignment statement u=-3\*s will result in u behaving as if it had been given the numbers -3, -6, -9. Notice that neither + nor == are member functions. (See Section 2.5 in the class notes). The result of s+t is a new statistician that looks as if it had been given all the numbers of the sequence for s, followed by all the numbers of the sequence for t. For example: Suppose that we have three statisticians s, t, and u. The statistician s has been given the numbers 1, 2, 3; the statistician t has been given the numbers 4, 5. Then the assignment statement u=s+t will result in u behaving as if it had been given the five numbers 1, 2, 3, 4, 5.