This lab investigates storing data in an array for further processing.

1. Make a copy of the program `stats_arr.cpp` in your own directory and compile and run it. This program incorporates our earlier program `stats.cpp` which computed the average and variance. However, our earlier program could not compute the median (middle value) because in order to do that, the numbers have to be stored and sorted (put in order). The new program `stats_arr.cpp` does this using an array `double x[MAXIMUM_DATA]` of double-precision reals.

2. Make up a datafile which has some values repeated and ending with the sentinel 0.0, for example, you might use

4.63 5.74 -0.02 -0.02 8.88 -2.57 11.03 3.44 11.03 5.61
11.03 -5.61 -0.02 -7.08 3.14 -9.97 11.03 -9.97 0.56 -2.20 -1.45 0.0

Run the program `stats_arr` again by redirecting standard input from the datafile.

3. After the number have all been entered the main program calls a function

```c
void insertion_sort(int num, double x[])
```

to do the sorting. Note how the array is passed to the function. It looks like pass-by-value but it is actually pass by reference (or, more precisely, pass by pointer). C/C++ observes the convention that arrays are always passed by reference/pointer because to pass them by value would involve too much copying. You should try to understand the logic of the insertion sort, but sorting functions get complicated quickly and the topic of efficient sorting goes beyond this course.

Assignment Write a program which will accept a sequence of double-precision reals, using 0.0 as the sentinel for end of data, which will find whether any values are duplicated and, if so, how many times. It should only print out when duplications are actually found. For example, such a program run on the data in (2.) above would find two duplications and output:

the value -0.02 occurs 3 times
the value 11.03 occurs 2 times

Before you start writing any code you should think about what algorithm you want to devise to solve this problem. You could, of course, check every value against every other value, although that might not be the most efficient solution. You could also sort the numbers first, for example, using

```c
void insertion_sort(int num, double x[]);
```

This would rearrange the data making any duplications occur together in a block. The algorithm you choose is up to you but I want a short comment at the beginning of your program explaining why you chose the algorithm you did and how it works. Most importantly, your program must work for any datafile not just the sample data given above.

Email me the pathname of your program in plain text, not as an attachment.