Overview

This homework contains two programming problems due at the start of next week's lab. After you have written, compiled and debugged your solutions, submit all source (.cpp) and header (.h) files together as attachments to a single email. (You will likely have 5 files for this assignment.) The subject line should just have Week 3 on it. The email address is

cs2projsx@cs.rpi.edu

where x should be replaced by your section number.

Practice

The following is a sketch of some practice exercises you might try on your own (or with friends) to be sure you understand the material

1. Do you understand parameter passing? In some cases, the difference is almost purely in efficiency (passing by value is slower than passing by constant reference because the entire structure is copied), and in some cases it is absolutely crucial to the correctness of your programs. Re-examine the way parameters are passed in the average_median_mode.cpp program to see if you understand each. Make changes to the way the parameters are passed, compile the program, and check to see if it is still correct. Make sure you can explain what happens.

2. If you are uncomfortable with vectors, make up small examples. For instance, write a program to read in a sequence of double values, sort them, and output them. Add the values. Add every other value. Output the results.

3. Similarly, to obtain extra practice with structs, try to make up small, almost meaningless examples to play with the mechanics.

4. Take a working program and introduce syntax mistakes on purpose to see what error messages the compiler gives. Experience like this in controlled settings helps you learn to deal with errors in new programs that you write.
Problems

Submit solutions to the following two problems for grading.

1. **(20 points)** Modify the `stats.h` and `stats.cpp` files from class on Monday to add a function that will find all modes — not just the one with the largest value — in a vector of integers. The function prototype should be

   ```c
   void all_modes( vector<int> values, vector<int>& modes,
                  int& mode_count );
   ```

   For example, for the following values,

   ```
   56, 22, 43, 56, 76, 22, 13, 76, 22, 76, 89, 76, 43, 22
   ```

   After the function is finished, the `modes` vector should contain 22 and 76 and `mode_count` — the number of occurrences of the mode — should be 4.

   Submit **only** the files `stats.h` and `stats.cpp`. We will use our own main program to test your function. Include in the comments in the .cpp file an explanation of why pass-by-value is used for the `values` parameter and pass-by-reference is used for the other two parameters.

   **Note:** many exam questions will consist of writing short functions like this. It is good practice to try to write the solution by hand first!

2. **(60 points)** Write a complete program to read in a sequence of employee records for a company, and calculate the pay for each employee for the current month. The employees will also be divided into salaried and hourly employees. Your program should output all of the employees names and their pay for the month alphabetically by last name. It should also output separate lists of the salaried employees and their pay, and the hourly employees and their pay. These two lists each should be ordered by increasing pay. Finally, the program should output the total amount of money the company must pay out this month.

   In the input, there will be one employee per line. Each input line will have the first name, the last name, and a 0 or 1 — the 0 indicates an hourly employee and the 1 indicates a salaried employee. For an input line with a 0, the rest of the line will contain an integer to indicate
the number of hours worked and a double to indicate the hourly wage. For an input line with a 1, the rest of the line will contain just a single integer giving the annual salary, which should be divided evenly across each month. An example input file is available at

http://www.cs.rpi.edu/~stewart/hw/week3/employees.txt

The input should be provided to your program by redirecting the input, just as in the week 3 lab.

Your program must illustrate the following:

• Use of structs and vectors
• Logical division of the program into functions with proper parameter passing
• Splitting source code across different files, with use of header files.

None of these are particularly long or difficult for this example. For the struct containing the employees, you may have a member variable for the type of employee, for the annual salary, for the hourly wage and for the number of hours worked. These will be in addition to member variables common to both types of employees. Don’t worry about wasting space for unused member variables. Later in the semester we will see how to avoid the waste here using what’s known as a class hierarchy.