CSS 342 Programming Assignment #1 Due: Wednesday, April 9, 2014 (from syllabus) **Spring 2014**

Part 1: Getting started and following directions

Send me (zander@u.washington.edu) email from any account -- school, work, or home account. Please use 342 in the subject of the email. In the email body, the first line has a 4 digit number (to be used as an anonymous ID), one blank, first name followed by one blank, then last name. For example (of course, use a number of your choice and your name):

1234 Carol Zander

After this, tell me about the two programming classes you took to get into CSS.
- Where did you go for the two programming courses required for CSS? What language was used in those courses?
- Briefly describe any programming experience you have beyond those courses (e.g., VB at school or work, etc.)
- Something interesting, either about yourself or anything

Let me know if you have concerns about 342. Note that I will not answer your email unless you ask a specific question.

Part 2: Programming, Review structs, arrays, functions, etc. (not object-oriented)

You are given names, last name then first name (each at most 30 characters long), and one integer (represents grade). You expect no more than 100 records. Sort these using an insertion sort (sorted by last name, then by first name when the last names are the same) and display. Also display a histogram and class average (using integer arithmetic).

SAMPLE INPUT:

```
frog freddie 60
duck donald 65
mouse minnie 95
mouse mickey 73
error check -10
error check 101
ghost casper 71
mouse abby 85
```

SAMPLE OUTPUT (yours must be identical):

```
List of names sorted:
 65  duck donald
 60  frog freddie
 71  ghost casper
 85  mouse abby
 73  mouse mickey
 95  mouse minnie

Histogram of grades:
   0-->   9:
  10-->  19:
  20-->  29:
  30-->  39:
  40-->  49:
  50-->  59:
  60-->  69: **
  70-->  79: **
  80-->  89: *
  90-->  99: *
 100--> 100:

Average grade: 74
```

Part 2 NOTES

- For practice, use char arrays and the functions found in `string.h` (including `iostream` is usually sufficient). Abstractly, arrays of char are similar to strings.

  A function prototype having a parameter such as (`char *s1`) is the same as (`char s1[]`). The `*` is a pointer, a pointer to a char which is the same as an address, which is how arrays are defined. Note that all char arrays used as strings must have `'\0'` as their last character so that output routines work properly. The >> operator does this for you.

  There is no =, so to copy, use `strcpy` function to copy (assign) strings (really char arrays, but called strings). For example:

```
char name1[MAX], name2[MAX];
strcpy(name1, name2);
```

  The content of the array name2 is copied to name1 (if it were an assignment, would be like: `name1 = name2;`).

  There is no ==, so to compare, use **`strcmp`** function (uses ascii values) to see if two *strings* (char arrays) are equal (function returns zero), less than (returns a value < 0), or greater than (returns a value > 0) each other. For example:

```
if (strcmp(name1, name2) < 0) {
    cout << name1 << " is alphabetically less than " << name2;
}
```

– You need two strings (char arrays) for each person, for last name and first name, and an int. Use an array of structs

```
struct StudentType {                          // will be a global definition
    int grade;
    char last[MAXLENGTH];
    char first[MAXLENGTH];
};
StudentType students[MAXSIZE];                // will be a local declaration
```

A struct is similar to a class, but typically without methods and all data is public. Note that *StudentType* is a global definition, but the variable *students* is local to main. A global definition is outside of any classes or functions and can be used anywhere. The arrays above have statically allocated memory (allocated at compile time), fixed in the given size.

Other things you can do, assuming you have the declaration: `StudentType temp;`
You can do things like

```
infile >> temp.last >> temp.first;      // read whole thing like with string
strcpy(students[i].last, temp.last);    // acts like an assignment into the array
students[i] = temp;                     // assigning a struct copies correctly
```

– Do this entire program in one .cpp file using only main and other functions (do not use classes). (Java students: this means to pretend main and other functions used are wrapped in a class, similar to the class containing main in your Java programs. Other functions in this program would be like additional methods of that class.) All variables that would have been in the class will be local to main and must be passed to other functions as parameters. Functions still have local vars as needed.

– For the insertion sort, read one line of data into a temporary location of type StudentType and then insert it into its correct position in the array. Continue until all are sorted. The reading and sorting happens at the same time; before the call to sortInput, the array has no items in it. (Reading into the array is not in a separate loop like in the bubble sort program.) You must code it this way. The algorithm (pseudocode) to insert one item:

```
assume name and grade are in a temporary location
loop from the end of the filled part of the array down to beginning {
    if (new last name < current last name) {
        copy current name and grade down
    }
    else if (new/current last names are equal and new first name < current first name) {
        copy/assign current name and grade down
    }
    else {
        found right place, break out of loop
    }
}
now that you've made room, insert (copy) new data into correct sorted position
```

As always expected when programming, comment clearly and thoroughly. Clearly state any assumptions you make in the beginning comment block of the appropriate place, e.g., function definition. For example, you can assume the data is properly formatted as described, and document this when you input the data. Pre and post conditions are recommended, but not required. Functions must be separated using comment separators as in my examples.

Non object-oriented programs are function driven (as opposed to data driven). Functions perform actions instead of objects invoking methods. To demonstrate, main for your lab1 is given (see website). You must use it.

Turn in - find turn-in instructions on the website.

Part 3: Questions (must be handwritten on 8.5 by 11 inch paper)

1. What happens if you turn in an assignment electronically a few minutes after the due time?

2. What happens if you turn in an assignment electronically after that (before it closes)?

3. What happens if you email me about turning in your assignment late?

4. What happens if you email your assignment code?