Eric Roberts
CS106B

Handout #9
January 12, 2015

# Section Handout #1—Simple C++

Sections will meet once a week to give you a more intimate environment to discuss course material, work through problems, and raise any questions you have. Each week we will hand out a set of section exercises, which are the problems proposed for section that week. While we will not be collecting or grading these problems, you will receive greater benefit from section if you've looked over the problems in advance and tried to sketch solutions. Your section leader won't necessarily cover every exercise in depth, so be sure to speak up if there are particular problems you'd like to focus on. Solutions to all problems will be given in section so you can work through any remaining exercises on your own. Many of the section problems have been taken from old exams, so they are also an excellent source of study questions when exam time rolls around.

For this week's section, your mission is to solve three problems, which are designed to help you practice the solution techniques you need for Assignment #1.

**Problem 1. String manipulation**

Your first problem is to design and implement a function **`censorString`** that takes two strings as input and eliminates from the first string all characters that are present in the second. Thus

> **`"Stanford University"`** with **`"nt"`** removed becomes **`"Saford Uiversiy"`**
> **`"Llamas like to laugh"`** with **`"la"`** removed becomes **`"Lms ike to ugh"`**
> and so on . . .

Note that the function is case sensitive, so that the uppercase **`L`** in **`Llamas`** stays there.

You could design this function to operate in two different ways. One way is to have the function return a new string, leaving the original string unchanged. A second way is to write a procedure that modifies the original string. For this section, try writing both of these versions. First, write a function that returns a completely new string with the following prototype:

```
string censorString1(string text, string remove);
```

When you complete the first version, write a function that modifies the original string instead. This version should have the following prototype:

```
void censorString2(string & text, string remove);
```

In this class, we're much more likely to write programs that operate functionally in the style of **`censorString1`**. Even so, it is important to be able to read C++ programs that use either style.

**Problem 2. File processing and reference parameters**

When we grade your exams, we typically keep track of various statistics like the minimum, maximum, and mean (i.e., the traditional average) scores. Write a function

```
void readStats(string fname, int & min, int & max, double & mean);
```

that takes the name of an input file and three reference parameters that will hold this statistical information. Your implementation should open the file, read the scores one line at a time, close the file, and then return to the client with the values of **`min`**, **`max`**, and **`mean`** correctly set. For efficiency's sake, your function should make only a single pass over the file. Your implementation should also call **`error`** with an appropriate message if the file does not exist or if any of the scores are not in the range 0 to 100.

**Problem 3. Recursive functions (adapted from Chapter 7, exercise 1, page 344)**

Suppose that you have somehow been transported back to 1777 and the Revolutionary War. You have been assigned a dangerous reconnaissance mission: evaluate the amount of ammunition available to the British for use with their large cannon, which has been shelling the Revolutionary forces. Fortunately for you, the British—being neat and orderly—have stacked the cannonballs into a single pyramid-shaped stack with one cannonball at the top, sitting on top of a square composed of four cannonballs, sitting on top of a square composed of nine cannonballs, and so forth. Unfortunately, however, the Redcoats are also vigilant, and you only have time to count the number of layers in the pyramid before you are able to escape back to your own troops. To make matters worse, computers will not be invented for at least 150 years, but you should not let that detail get in your way. Your mission is to write a recursive function **`cannonball`** that takes as its argument the height of the pyramid and returns the number of cannonballs it contains. Your function must operate recursively and must not use any iterative constructs, such as **`while`** or **`for`**.

For example, in a cannonball stack of height 4, there are 30 (16 + 9 + 4 + 1) arranged in the following layers: