COMP161 - Lab 7 & Homework 5

Spring 2015

For this lab you'll work on writing a mutator procedure. For homework you'll work on an output procedure. Both topics are covered in lecture notes 9.

Lab 7

An n-gram letter frequency analysis counts the frequency of n letter sequences in a string. These statistics can then be used for several purposes in natural language processing and cryptography. A common technique to handle the begging and end of a document is to pad both with $(n-1)$ special characters such that the first n gram is the first letter of the text preceded by $(n-1)$ special characters and the last n gram is the last character followed by $(n-1)$ special characters. It is usually important that the padding character is unique and doesn't occur elsewhere in the text as it signifies the beginning and end of the document.

Your task is to design and develop a general purpose STRING MUTATOR called *add_padding* that can be used for this purpose. The procedure should take a string *str*, a positive integer *pad_size*, and a character *pad_char*. If the character *pad_char* doesn't occur in *str*, then it should then modify *str* by adding *pad_size* occurrences of *pad_char* to both the beginning and end of the string. If *pad_char* does occur in *str*, then it should leave *str* unchanged.

In order to complete this procedure, you'll need to make use of the *fill constructor*[1] for the std::string class as well as the std::string method *find*[2]. So, a secondary objective of this lab is to give you practice reading a language reference. As always, your procedure should be fully documented[3] and have a sufficient set of tests in addition to be fully defined. At the end of lab, submit a compilable set of code as *lab7* using the *handin* command.

[1] http://www.cplusplus.com/reference/string/string/string/

[2] http://www.cplusplus.com/reference/string/string/find/

[3] now that we're writing procedures for effect, don't forget post conditions

Homework 5

Due By 8 am, Monday March 2nd

Printing data in an organized, tabular form is a common output task. Tabular output presents data aligned into columns and rows and typically prints each value in a specific way. For this problem you are to write an output procedure called *styleRowOut* that produces *one row* of a table of beer style statistics like those see here:

```
Name           OG      FG      ABV     IBU   SRM
Maibock        1.07    1.016   7.2     29    7
Bock           1.07    1.021   7       25    25
Doppelbock     1.077   1.017   7.3     22    21
Eisbock        1.095   1.028   11.5    30    34
```

The first row you see is the header; your procedure does not print that row. The remaining rows are the data rows and are examples[4] of the kind of output your procedure should produce. Each row contains a string, the name, and five doubles.

- The *OG* and *FG*[5] are measurements of beer taken before and after fermentation. They're always reported to three decimal places and between 1.0 and 2.0.

- The *ABV*[6] measures the strength of the beer. It is always reported to one decimal place. The ABV tends to be between 3 and 15.

- The *IBU*[7] measures the hop character and is reported with zero decimal places. The IBUs of a beer tend to be between 10 and 120.

- The *SRM*[8] is a measure of beer color and is reported with zero decimal places. The SRM of a beer is typically between 5 and 50.

To control the precision of doubles printed in a stream we can use *std::setprecision*[9] from the library *iomanip*[10]. This I/O manipulator works in a similar fashion to *std::boolalpha*, which you've seen before; it controls the output of tokens after it in the stream.

To provide the illusion of a column of data we control the width of each token printed. If you print the number 14 with a width of 10 then the output will contain 8 extra spaces so that the total width is 10, 2 for the digits and 8 spaces. You can set the width using *std::setw*[11] from the *iomanip* library. The trick is to print the values for each column the same width.

- Names should be printed 20 wide.

- All the numerical values should be printed 8 wide.

It should be noted that the example table shown above is not necessarily printed with these width values. It was generated by Mathematica and not a C++ program.