Introduction to Systems Programming (Systems I)

Homework #3 (Part B)

Due: Tuesday September 20 2016 before 11:59 PM

Email-based help Cutoff: 5:00 PM on Mon, September 19 2016

Maximum Points: 25

Submission Instructions

This homework assignment must be turned-in electronically via Canvas. Ensure your C++ source code is named *MUid_homework3*.cpp, where *MUid* is your Miami University unique ID. Ensure your program compiles (without any warnings or errors) successfully. Ensure you have tested operations of your program as indicated. Once you have tested your implementation, upload the following onto Canvas:

1. The 1 C++ source file developed for this part of the homework.

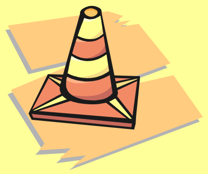
General Note: Upload each file associated with homework (or lab exercises) individually to Canvas. Do not upload archive file formats such as zip/tar/gz/7zip/rar etc.

Objective

The objective of this homework is to:

- Further understand the use of command-line arguments.
- Continue to review the use of file streams for text file I/O
- Continue to gain familiarity with developing C++ program involving simple string manipulation

Grading Rubric:



The program submitted for this homework **must pass necessary base case test(s) in order to qualify for earning any score at all**. Programs that do not meet base case requirements will be assigned zero score!
Program that do not compile, have a method longer than 25 lines, or just some skeleton code will be assigned zero score.

- **-1 Points**: for each warning generated by the compiler
- **NOTE:** Violating CSE programming style guidelines is a compiler error! Your program should not have any style violations reported in NetBeans when you compile it.

Starter Code:

For this homework no starter code is supplied (as it is relatively a straightforward homework). However, a few input and expected-output files are supplied for testing. You need to copy the files to your `NetBeans` project folder for testing.

Homework Exercise

This homework requires development of a relatively straightforward string-processing program that converts specific elements on a standard tweet into HTML format. The file with tweets (1st argument) and the output HTML file (2nd argument) are specified as command-line arguments. Each line in the input file contains 1 tweet. A tweet is 1 line string consisting of 1 or more words separated by 1 (or more) blank spaces. Words in a tweet must be converted to HTML format. The rules for converting tweets and 4 types of words in a tweet into HTML are:

1. Each tweet should be placed within a HTML `div` tag (with CSS class `tweet`), that is between, `<div class="tweet">` and `</div>`.

2. Convert tweet handles of the form `@xyz` to HTML anchors of the form: `<a href="https://www.twitter.com/xyz">@xyz</a>`. For example, the string `"@miamiuniversity"` should be converted to the string `"<a href="https://www.twitter.com/miamiuniversity>@miamiuniversity</a>"`

3. Convert hash tags of the form `#abc` to HTML anchors of the form: `<a href="https://www.twitter.com/hashtag/abc">#abc</a>`. For example, the hash tag `"#LoveAndHonor"` should be converted to the string `"<a href="https://www.twitter.com/hashtag/LoveAndHonor">#LoveAndHonor</a>"`

4. Convert 2 or more blank spaces (single blank spaces remain unchanged) to HTML entity `"&nbsp;"`. Note that blank spaces may occur at the beginning, between words, or end of tweets. For example, the string `"   "` (3 blank spaces) is converted to the string `" &nbsp;&nbsp;"`.

5. Convert the following 4 emoticons to HTML `img` (image) tags (with suitable URLs) as tabulated below:

Emoticon	HTML tag
:-)	`<img src="http://ceclnx01.cec.miamioh.edu/~raodm/hw3/smile.png">`
:-(	`<img src="http://ceclnx01.cec.miamioh.edu/~raodm/hw3/frown.png">`
:-|	`<img src="http://ceclnx01.cec.miamioh.edu/~raodm/hw3/normal.png">`
:++	`<img src="http://ceclnx01.cec.miamioh.edu/~raodm/hw3/cpp.png">`

Additional HTML formatting

In addition to formatting each tweet as an HTML, the following fixed HTML header and footer needs to be added (similar to how it was done in previous exercises) to the output file generated by your program is a valid HTML:

HTML header

```
<!DOCTYPE html>
<html>
<head>
<link type="text/css" rel="stylesheet"
href="http://ceclnx01.cec.miamiOH.edu/~raodm/hw3/tweets.css"/>
</head>
<body>
```

HTML footer

```
</body>
</html>
```

Example of Tweets converted to HTML

Here are a few examples of tweets (*i.e.*, strings) and corresponding HTML versions (*i.e.*, strings) of them. The examples actually show contents of input files and contents of output files.

Contents of input file:

```
One simple tweet
```

Contents of output file:

```
<!DOCTYPE html>
<html>
<head>
<link type="text/css" rel="stylesheet"
href="http://ceclnx01.cec.miamiOH.edu/~raodm/hw3/tweets.css"/>
</head>
<body>
<div class="tweet">
One simple tweet
</div>
</body>
</html>
```

Contents of input file (`base_inputs.txt`):

```
@miamiuniversity
#LoveAndHonor
:++
3 space   test
```

Contents of output file:

```
<!DOCTYPE html>
<html>
<head>
<link type="text/css" rel="stylesheet"
href="http://ceclnx01.cec.miamiOH.edu/~raodm/hw3/tweets.css"/>
</head>
```

```
<body>
<div class="tweet">
<a href="https://www.twitter.com/miamiuniversity">@miamiuniversity</a>
</div>
<div class="tweet">
<a href="https://www.twitter.com/hashtag/LoveAndHonor">#LoveAndHonor</a>
</div>
<div class="tweet">
<img src="http://ceclnx01.cec.miamiOH.edu/~raodm/hw3/cpp.png">
</div>
<div class="tweet">
3 space &nbsp;&nbsp;test
</div>
</body>
</html>
```

You may also view the output files in your web-browser by placing the output files in your `public_html` folder. Here are sample URLs that you can use to view two generated output files in your web-browser:

- http://ceclnx01.cec.miamioh.edu/~raodm/hw3/expected_base_output.html
- http://ceclnx01.cec.miamioh.edu/~raodm/hw3/expected_tweet_output.html

Tips

Here are a few tips/suggestions (you can ignore any/all of them if your solution does not need them) to help you develop this program:

- This is a relatively straightforward program. So don't over complicate it.
- However, as with all programs you will need 2 or 3 tries to streamline the code (in case you are under the wrong impression that most people code programs just once).
- Organize your program into methods, one for processing each type of word, and one that processes 1 tweet, and 1 that reads tweets line-by-line.
- The string processing straightforward. The methods like `find_first_of`, `find_first_not_of`, etc. will come in handy. Review documentation online and memorize the methods.
- Save every 2 or 3 lines of code you type, compile, and resolve compiler errors. **Typing 7 or 10 lines of C++ and then compiling is a bad idea for beginners**.

Functional Testing

You can verify correct operation of your implementation by comparing or `diff`'ing your program's output against the supplied expected output files from a `NetBeans` terminal using the commands shown below:

! Base case – program must generate the expected outputs in this case to earn any credit for this homework:

```
$ ./homework3 base_inputs.txt base_output.html
$ diff base_output.html expected_base_output.html
```

```
$ ./homework3 tweet_inputs.txt tweet_output.html
$ diff tweet_output.html expected_tweet_output.html
```

☞ **Note**: The above `diff` commands should not generate any output indicating that your outputs are exactly the same as the expected outputs.

Turn-in:

This homework assignment must be turned-in electronically via Canvas. Ensure your C++ source code is named `MUid_homework3.cpp`, where `MUid` is your Miami University unique ID. Ensure your program compiles (without any warnings or errors) successfully. Ensure you have tested operations of your program as indicated. Once you have tested your implementation, upload the following onto Canvas:

- The single C++ source file developed as part of this homework.

Upload all the necessary C++ source files to onto Canvas. Do not submit zip/7zip/tar/gzip files. Upload each file independently.