COSC – Fall 2017
Programming Assignment 2
html2latex – HTML to Latex converter with Lex

For the second programming assignment, students will learn to use the Lex lexical analyzer generator by constructing a program that will convert a subset of HTML into LaTeX. Begin the project by downloading the PA2 starter directory from the course website. The PA 2 starter directory contains a Make based build system, initial Lex source code for the HTML-to-LaTeX converter, an example input file, and an executable ("mj_html2latex") with a working solution. You can use the solution executable to test whether your program produces the same output as the solution we will use to grade your assignment. If you have never used LaTeX to typeset a document then the open-content LaTeX book may be helpful.

To begin, create the project without making any changes by typing "make" on your terminal after having extracted the source code and after having entered the html2latex directory. This should create an executable named "html2latex" inside of the PA 2 directory. Once you have done this correctly you can attempt to convert the example input file into a PDF output file by typing "make test" on your terminal. This will fail with an error because you have not completed the assignment.

The purpose of this project is to convert the subset of HTML given below into correct LaTeX output that can then be converted into a PDF. The PA2 starter code that you downloaded has an example of how to use Lex to convert the HTML H1 tag and HTML comments into LaTeX. You will complete the project by adding the required additional rules to the Lex source code file "html2latex.l". To receive full points for this project you will need to produce correct output by converting the following HTML tags into LaTeX:

- HTML Comments:
  HTML comments begin with the tag "`<!--`" and end with the tag "`-->`". Any character between these two tags is considered a comment. This includes other HTML tags (i.e. HTML tags appearing inside of an HTML comment are considered part of the comment). The PA2 starter directory contains Lex source code to handle HTML comments appropriately.

- HTML Headings:
  You must convert HTML headers for H1, H2, and H3 into LaTeX source code. The HTML headers begin with the tags "`<h1>`", "`<h2>`", and "`<h3>`" respectively and end with the tags "`</h1>`", "`</h2>`", and "`</h3>`". You will convert an HTML H1 header into a LaTeX "`section`" element, an HTML H2 header into a LaTeX "`subsection`" element, and an HTML H3 header into a LaTeX "`subsubsection`" element. You can assume that the content inside of an HTML heading is plain text (i.e. HTML tags cannot appear between the begin tag and end tag). The PA2 starter directory contains Lex source code to handle HTML H1 headers correctly.

- HTML Pre-formatted Paragraphs:
  HTML pre-formatted paragraphs begin with the tag "`<pre>`" and end with the tag "`</pre>`". Any character between these two tags, including white space, is considered to be part of the pre-formatted text and should be output exactly as seen (i.e. all text between the beginning and ending tags should be preserved exactly in the output). HTML pre-formatted paragraphs should be output using LaTeX "`verbatim`" environments.

- HTML Paragraphs:
  HTML paragraphs begin with the tag "`<p>`" and end with the tag "`</p>`". HTML paragraphs

contain text that can be intermingled with certain HTML tags. The list of tags that you are required to support appear below. You should convert HTML paragraphs into LaTeX paragraphs.

- HTML Ordered and Unordered Lists:
  HTML ordered lists begin with the tag "`<ol>`" and end with the tag "`</ol>`". Likewise, unordered lists begin with "`<ul>`" and end with "`</ul>`". Both types of lists can contain only list items that begin with the tag "`<li>`" and end with the tag "`</li>`". HTML list items contain the same type of text as HTML paragraphs. You should convert HTML unordered lists into LaTeX "`itemized`" environments and HTML ordered lists into LaTeX "`enumerate`" environments.

HTML paragraphs and HTML list items can both contain HTML tags inside of their text region. You are not required to support nesting of these HTML tags so you can assume the text between each of the tags is plain text. You must support the following tags inside of paragraphs and list items:

- Small Texts:
  HTML small text begins with the tag "`<small>`" and ends with the tag "`</small>`". HTML small text should be converted to a LaTeX "`scriptsize`" element.

- Large Text:
  HTML large text begins with the tag "`<big>`" and ends with the tag "`</big>`". HTML large text should be converted to a LaTeX "`Large`" element.

- Bold Text:
  HTML bold text begins with the tag "`<b>`" and ends with the tag "`</b>`". HTML bold text should be converted to a LaTeX "`textbf`" element.

- Italic Text:
  HTML italics text begins with the tag "`<i>`" and ends with the tag "`</i>`". HTML italic text should be converted to a LaTeX "`textit`" element.

- Strong Text:
  HTML strong text begins with the tag "`<strong>`" and ends with the tag "`</strong>`". HTML strong text should be converted to a LaTeX "`textmd`" element.

- Emphasized Text:
  HTML emphasized text begins with the tag "`<em>`" and ends with the tag "`</em>`". HTML emphasized text should be converted to a LaTeX "`emph`" element.

- Superscript Text:
  HTML superscript text begins with the tag "`<sup>`" and ends with the tag "`</sup>`". HTML superscript text should be converted to a LaTeX "`textsuperscript`".

- Subscript Text:
  HTML subscript text begins with the tag "`<sub>`" and ends with the tag "`</sub>`". HTML subscript text should be converted to a LaTeX "`textsubscript`" element.

Note, all of the HTML tags in all of the examples that I will be using are in lower case. **Your HTML-to-LaTeX converter does not need to support upper case tags but it must support upper case letters inside of all of the tags.** Your PA 2 program will be graded on inputs other than the example input given so be sure it handles all of the cases given above.

After you complete your implementation, you should rigorously test your program to ensure its correctness. We will test your code with inputs other than the example input given, so be sure it handles all of the items listed above.

Finally, you will submit a project report. The project report is a short (one to two page, single-spaced) document that describes:

1. (in your own words) the problem you set out to solve,
2. your approach (i.e. design and relevant implementation details) for solving this problem,
3. how you debugged and tested your solution, and
4. any issues you had in completing the assignment.

Please comment your code so that others (i.e. the grader) can understand it. Comments at the top of the file should indicate your name, this course, and the assignment. You should attempt to match your output as closely as possible to our output.

When you have completed your assignment, you should upload a gzipped tar file (created with `tar cvzf ...`) with your source files and a pdf of your report to the Canvas course website before the end of the day, anywhere on earth, on Friday, September 22th. (That is, you must submit your project by 6:59am EDT on Saturday, September 23rd to avoid a late penalty). Partial credits will be given for incomplete efforts. However, a program that does not compile or run will get 0 points. Point breakdown is below:

- HTML 2 Latex conversion (80)
- Code is easy to understand (e.g., contains appropriate comments) (10)
- Project report (10)