Lab 3 Report

How it works:

The program parses specific HTML tags and outputs their corresponding Latex commands to a .tex file. For example, “<h1>[^<]*</h1>” would parse an HTML header and everything in between the open and close HTML tags. The corresponding Latex command that is printed is \section{%s}, where %s is the text between the open and close tags.

This simple method works well for non-nested HTML tags. However, some HTML tags were nested. Since it would be very difficult (or perhaps impossible) to write a single regular expression toe capture an HTML tag and any combination of nested sub-tags, Lex states were used. As an example, <p> (paragraph) contains plain text and nested HTML tags that modify the appearance of that text, say, <b> for bold text. When <p> is encountered, the program changes states to state PARAGRAPH. In the PARAGRAPH state, we can parse all of the nested tags with their own regular expressions. For <b>, the regular expression “'<b>'[^b]*'</b>'” was used. Similar expressions were used for small, large, italic, strong, emphasized, superscript, and subscript modifiers.

A second state, LIST, was used to HTML ordered and unordered lists. When <ol> or <ul> tags are encountered from state 0 , the program outputted “begin{enumerate)” or “begin{itemize}”, respectively, and transitioned to state LIST. LIST checks for a list item, <li> or the closing ordered/unordered tags. If <li> is parsed, the program changes states to PARAGRAPH and begins the process described in the previous paragraph.

Problems:

The problem that took me the longest time to debug was in the regular expression that parsed <pre> content. My original expression inside the <pre> and </pre> was simply “.*”. I thought the “.” operator captured all text, including line breaks and tabs. It doesn't. The modified and correct expression is “(.|[\r\n])*”.