Department of Computer Science and Engineering

Programming Languages

II Semester 2022-23

Instructor: [S. Arun-Kumar]

Assignment: MDT: A translator from Markdown-- to HTML text with tables

Introduction

HTML is often too verbose and complicated for most static pages like this one. Most writers of web-pages require only the simplest HTML mark-ups with the fewest possible options and even fewer special symbols. For such simple web-pages it suffices to use a few simple symbols to format the text written in a text editor (such as *gedit*, *Emacs* or even *notepad*) as an HTML document.

These considerations led to the design of *markdown* which is a simple program that converts text into HTML with a minimum number of "markup" symbols. There are now several versions with various sophisticated features that can do most things such as convert the markdown file into HTML or LaTex or even directly into pdf.

The original markdown did not support HTML tables. So we add that as an essential feature with its own markdown syntax along with some more simplifications by stripping down and changing some functionalities.

Markdown--: Essentials

1. This HTML file `mdtab-2023.md.html` was produced by running markdown on the text file `mdtab-2023.md`. Use the text file to test your own program.
2. To know about markdown go to the markdown syntax page maintained by John Gruber. Many of the simplifications listed below also refer to Gruber's document.
3. Recognize and leave all inline HTML tags unchanged.
4. **Headings.** Do not implement the underlining feature of markdown. Implement only the headings defined by hashes. Hence you must implement the 6 levels of headings which are the levels shown in the centered lines at the beginning of this document. The number of hashes at the beginning of the line indicates the heading level.
5. Implement the following for pure text elements -- Use of double asterisks for **bolding** and single asterisks for *italics*. There is nothing in markdown for underlining -- under-scores are used for italics and bolding as an alternative to asterisks. Instead implement underlining as follows. Example: "_underlined_text_" produces underlined text).
6. **Horizontal Ruling**. Implement only the code corresponding to the use of at least 3 consecutive hypens (---) to produce a horizontal line.
7. **Block quotes.** Implement only the email-style block-quoting (i.e. every line in a block quote should begin with >, otherwise it is not a block quote) along with the nested block-quoting.
8. **Links.** Implement the direct href produced by the using the "[text](http://url/of/the/link)" syntax. Also implement the automatic links feature e.g.

   ```
   <http://www.cse.iitd.ac.in/~sak>
   ```

   renders as http://www.cse.iitd.ac.in/~sak which provides the URL and the link to the URL.
9. **Lists.** Implement
   - *ordered lists* -- as given by John Gruber. Pay particular attention to his 1986 example relating to escape sequencing the "." in case 1986 comes at the beginning of a line.
   - *unordered lists* -- implement only the hyphen (as done here).
10. Implement tables from csv format. Make the simplifying assumption that a HTML table is always centered. For example the following is

    ```
    <<
    1|2|3|4|9
    5|6|7|8|10
    11|12|13|14|15
    >>
    ```

    a possible syntax for an HTML table that looks like

1	2	3	4	9
5	6	7	8	10
11	12	13	14	15

What you need to do.

Write a SML function `mdt2html` which takes a single input text file *filename*`.mdt` and translates it into a HTML file *filename*`.html` that may be viewed on any browser. Your SML file should be called `mdt2html.sml`

You may use all the functionalities given in the `TextIO` and the `String` modules of SML. While this requires using some of the impure features of SML, your primary focus should be on not using the imperative constructs of SML -- such as loops and assignment -- unless absolutely essential.

Note:

1. Since the above description is as clear as I can possibly make it, If you still have some doubts, make reasonable design decisions and explicitly state them in a `README.mdt` to be submitted along with your code.
2. Upload and submit a single zip file called *entryno*`.zip` where *entryno* is your entry number. This file should contain the two files `mdt2html.sml` and `README.mdt`.
3. You are *not* allowed to change any of the names or types given in the specification/signature. You are not even allowed to change upper-case letters to lower-case letters or vice-versa.
4. The evaluator may use automatic scripts to evaluate the assignments (especially when the number of submissions is large) and penalise you for deviating from the instructions.
5. You may define any new auxiliary functions/predicates you like in your code besides those mentioned in the specification.
6. Your program should implement the given specifications/signature.
7. You need to think of the *most efficient* way of implementing the various functions in a *functional* style given in the specification/signature so that the function results satisfy their definitions and properties.
8. In a large class or in a large assignment, it is not always possible to specify every single design detail and clear each and every doubt. So you need to submit a `README.mdt` file containing
   - all the decisions (they could be design decisions or resolution of ambiguities present in the assignment) that you have taken in order to solve the problem. Whether your decision is ``reasonable'' will be evaluated by the evaluator of the assignment.
   - a description of which code you have ``borrowed'' from various sources, along with the identity of the source.
   - all sources that you have consulted in solving the assignment.
   - name changes or signature changes if any, along with a full justification of why that was necessary.
9. The evaluator may look at your source code before evaluating it, you must explain your algorithms in the form of comments, so that the evaluator can understand what you have implemented.
10. Do *not* add any more decorations or functions or user-interfaces in order to impress the evaluator of the program. Nobody is going to be impressed by it.
11. There is a serious penalty for code similarity (similarity goes much deeper than variable names, indentation and line numbering). If it is felt that there is too much similarity in the code between any two persons, then both are going to be penalized equally. So please set permissions on your directories, so that others have no access to your programs.
12. To reduce penalties, a clear section called **Acknowledgements** giving a detailed list of what you copied from where or whom may be included in the `README.mdt` file.