COP701 - Software Systems Lab
Assignment 1 - HTML to LaTeX Converter

Nilaksh Agarwal - 2015PH10813

September 1, 2019

Objective

To convert a HTML document to an equivalent LaTeX document.

1. Learn about HTML and LaTeX in brief.
2. Write a lexer i.e to do a lexical analysis of your HTML code and generate a string of tokens. Programs that you can use: flex, jflex
3. Do not use any available libraries to parse the html.
4. Parse the sequence of tokens using parser such as yacc, CUP, ANTLR, bison (C++ or Java)
5. Generate an AST(Abstract Syntax Tree) of your HTML code
6. Map it to an equivalent AST of LaTeX.
7. Generate the equivalent LaTeX code which can be compiled to a PDF using TexMaker

1 Lexer (ply.lex)

In this part, we created lex commands using regular-expressions to tokenize the input HTML document.

```
tokens = [’STARTTAG’,
          ’ENDTAG’,
          ’EMPTYTAG’,
          ’COMMENT’,
          ’TEXT’,
          ’DOCTYPE’,
          ’FILLER’,
          ’ENDFILE’]

t_STARTTAG = r’<[^\/!][^<>]*>’
t_ENDTAG = r’<\/[^<>\!\/]*>’
t_EMPTYTAG = r’<[^\/!][^<>]*\/>’
t_TEXT = r’(?=[^<>]+)[^\n<>]+’
t_COMMENT = r’<\!--[^!]+-->’
t_DOCTYPE = r’<\![^-][^<>*]+>’
t_FILLER = r’[ ]{2,}|[\n]+’
t_ENDFILE = r’<\/[Hh][Tt][Mm][Ll]>’
def t_error(t):
   raise TypeError("Unknown text ’%s’" % (t.value))
};
```

Here, each token is defined as follows:

1. **STARTTAG:** All open HTML tags $< >$

2. **ENDTAG:** All closing HTML tags $< / >$

3. **EMPTYTAG:** All self-closed HTML tags, or tags that don’t require close tags (e.g. $< br > < img/ >$)

4. **COMMENT:** All comments $<! - - - - >$

5. **TEXT:** All text in the HTML document

6. **FILLER:** Filler values like multiple consecutive spaces or newline characters

7. **ENDFILE:** End of file indicator $< /html >$

Parser (ply.yacc)

After this, we implement a parser to parse this HTML document and generate the Abstract Syntax Tree (AST)

```
def p_document(p):
  '''document :
    | FILLER document
    | COMMENT document
    | DOCTYPE document
    | start document
    | end document
    | empty document
    | text document
    | ENDFILE
    '''
def p_start(p):
  '''start : STARTTAG'''
def p_end(p):
  '''end : ENDTAG'''
def p_empty(p):
  '''empty : EMPTYTAG'''
def p_text(p):
  '''text : TEXT'''
def p_error(p):
  print("Syntax error")
```

Here, we parse this document and create a AST with each token (except comment/fillers) as a node. This tree gives us the structure of the HTML document

A node of the tree is as follows:

```
class Node:
def __init__(self,type=None,parent=None,children=None
            ,value=None,attr=[],values=[]):
  self.type = type
  if children:
    self.children = children
  else:
    self.children = []
  if parent:
    self.parent = parent
  else:
    self.parent = None
  self.attr = attr
  self.values = values
  self.value = value
```

This is the AST generated from the HTML sample1.html, generated from this lexer and parser.

```
Tree
└── HTML
    ├── HEAD
    │   └── TITLE
    │       └── Sample document
    └── BODY
        ├── H1
        │   └── CSS
        ├── P
        │   ├── Lorem ipsum dolor sit amet, consectetuer...
        │   ├──  Nulla ut lectus id velit aliquet semper...
        │   ├──  vel nisl.Duis lobortis mi at lorem. Eti...
        │   ├── sem, adipiscing at,porttitor vitae, inte...
        │   └──  tincidunt eget , euismod ac, molestie q...
        ├── P
        │   ├── Lorem ipsum dolor sit amet, consectetuer...
        │   ├──  Nulla ut lectus id velit aliquet semper...
        │   ├──  vel nisl. Duis lobortis mi at lorem. Et...
        │   ├── sem, adipiscing at, porttitor vitae, int...
        │   └──  tincidunt eget, euismod ac, molestie qu...
        ├── P
        │   ├── Lorem ipsum dolor sit amet, consectetuer...
        │   ├──  Nulla ut lectus id velit aliquet semper...
        │   ├──  vel nisl. Duis lobortis mi at lorem. Et...
        │   ├── sem, adipiscing at, porttitor vitae, int...
        │   ├──  tincidunt eget, euismod ac, molestie qu...
        │   └──  hendrerit semper, accumsan ac, consequa...
        ├── P
        │   ├── Lorem ipsum dolor sit amet, consectetuer...
        │   ├──  Nulla ut lectus id velit aliquet semper...
        │   ├──  vel nisl. Duis lobortis mi at lorem. Et...
        │   └── sem, adipiscing at, porttitor vitae, int...
        ├── DIV
        │   └── P
        │       └── Div line 1.
        ├── DIV
        │   └── P
        │       └── Div line 2.
        ├── DIV
        │   └── P
        │       └── Div line 3.
        ├── H1
        │   └── Special symbols
        ├── H2
        │   └── Greek symbols
        ├── P
        │   ├── TT
        │   │   └── Alpha
        │   └── : &Alpha; , &Gamma; &Theta; &Xi; &Pi;
        ├── H2
        │   └── LaTeX chars
        ├── P
        │   └── { } _ ^ @ $ \ % ~ #
        ├── H1
        │   └── LaTeX commands in HTML
        ├── P
        │   └── It's easy to include LaTeX commands in H...
        ├── H1
        │   └── Different font styles and sizes.
        ├── P
        │   ├── Lorem ipsum
        │   ├── FONT --> (size,7)
        │   │   └── dolor
        │   ├──  sit amet,
        │   ├── I
        │   │   └── consectetuer
        │   ├──  adipiscing elit.
        │   ├──  Nulla ut
        │   ├── STRONG
        │   │   └── lectus
        │   ├──  id velit aliquet semper.
        │   ├── TT
        │   │   └── Proin vitae
        │   ├──  erat. Duis metus. Nam
        │   ├──  vel nisl. Duis
        │   ├── SMALL
        │   │   └── lobortis
        │   ├──  mi at
        │   ├── FONT --> (size,1)
        │   │   └── lorem
        │   └── .
        ├── A --> (name,img)
        ├── H1
        │   └── Images
        ├── P
        │   └──  supports only JPG and PNG images.
        ├── P
        │   └── CENTER
        │       └── IMG --> (src,marley.jpg)
        ├── P
        │   └── IMG --> (src,logo.png)
        ├── H1
        │   └── Tables
        ├── TABLE --> (border,1)
        │   ├── TR
        │   │   ├── TD
        │   │   │   └── 1 1
        │   │   ├── TD
        │   │   │   └── 1 hgf2
        │   │   └── TD
        │   │       └── hgfhf1 3
        │   ├── TR
        │   │   ├── TD
        │   │   │   └── 2 1
        │   │   ├── TD
        │   │   │   └── 2 2
        │   │   └── TD
        │   │       └── 2 3
        │   └── TR
        │       ├── TD
        │       │   └── 3 1
        │       ├── TD
        │       │   └── 3 2
        │       └── TD
        │           └── 3 3
        ├── BR
        ├── TABLE --> (border,0)
        │   ├── TR
        │   │   ├── TD
        │   │   │   └── Sparta Praha
        │   │   └── TD
        │   │       └── 28
        │   ├── TR
        │   │   ├── TD
        │   │   │   └── Slovan Liberec
        │   │   └── TD
        │   │       └── 25
        │   ├── TR
        │   │   ├── TD
        │   │   │   └── Dukla Praha
        │   │   └── TD
        │   │       └── 24
        │   └── TR
        │       ├── TD
        │       │   └── Slavia Praha
        │       └── TD
        │           └── 20
        ├── H1
        │   └── Subscript, superscript
        ├── P
        │   ├── H
        │   ├── SUB
        │   │   └── 2
        │   ├── O, E = mc
        │   └── SUP
        │       └── 2
        ├── H1
        │   └── Hyperlinks
        ├── P
        │   ├── I study at
        │   ├── A --> (href,http://www.mff.cuni.cz)(title,MFF)
        │   │   └── UK MFF
        │   ├── . And
        │   ├──     what about
        │   ├── A --> (href,#img)
        │   │   └── images
        │   └── ?
        ├── H1
        │   └── Some texts
        ├── P
        │   └── CENTER
        │       ├── They went in single file, running like h...
        │       ├── and an eager light was in their eyes. Ne...
        │       ├── swath of the marching
        │       ├── SMALL
        │       │   └── Orcs tramped
        │       ├──  its ugly slot; the sweet grass
        │       └── of Rohan had been bruised and blackened ...
        ├── P
        │   └── John said, I saw Lucy at lunch, she told
        ├── H1
        │   └── Lists and definitions
        ├── DL
        │   ├── DT
        │   │   └── Dweeb
        │   ├── DD
        │   │   ├── young excitable person who may mature
        │   │   ├──     into a
        │   │   ├── EM
        │   │   │   └── Nerd
        │   │   ├──  or
        │   │   └── EM
        │   │       └──  Geek
        │   ├── DT
        │   │   └── Hacker
        │   ├── DD
        │   │   └── a clever programmer
        │   ├── DT
        │   │   └── Nerd
        │   └── DD
        │       └── technically bright but socially inept pe...
        ├── P
        │   ├── In this section, we discuss the lesser k...
        │   └── ...this section continues...
        ├── H2
        │   └── Habitat
        ├── P
        │   ├── Forest elephants do not live in trees bu...
        │   └── ...this subsection continues...
        ├── H3
        │   └── Habitat
        ├── P
```

```
│   ├── Forest elephants do not live in trees bu...
│   ├── ...this subsection continues...
│   └── STRONG
│       └──  AND A LINE FOLLOWS
├── H2
│   └── List
└── UL
    ├── LI
    │   └──  ... Level one, number one...
    ├── OL
    │   ├── LI
    │   │   └──  ... Level two, number one...
    │   ├── LI
    │   │   └──  ... Level two, number two...
    │   ├── OL
    │   │   └── LI
    │   │       └──  ... Level three, number one...
    │   └── LI
    │       └──  ... Level two, number three...
    └── LI
        └──  ... Level one, number two...
```

Mapping AST to LaTeX

We perform a pre-order traversal of this AST, and map the tags to their LaTeXequivalent ones. For specific special tags like $< a >$, $< table >$, $< img >$, $< font >$ we define special functions to take care of the attributes.

```
html2latexdict = {
"HTML" : "",
"HEAD" : "",
"TITLE" : "\\title",
"BODY" : "\\begin{document}",
"H1" : "\\section*",
"H2" : "\\subsection*",
"H3" : "\\subsubsection*",
"H4" : "\\paragraph*",
"P" : "\\par",
"DIV" : "",
"FONT" : "",
"U" : "\\underline",
"B" : "\\textbf",
"I" : "\\textit",
"EM" : "\\emph",
"TT" : "\\texttt",
"STRONG" : "\\textbf",
"SMALL" : "\\small",
"SUB" : "\\textsubscript",
"SUP" : "\\textsuperscript",
"A" : "",
"CENTER" : "\\begin{center}",
"IMG" : "\\includegraphics",
"FIGURE" : "\\begin{figure}",
"FIGCAPTION" : "\\caption*",
"TABLE" : "",
"CAPTION" : "\\caption*",
"TH" : "",
"TR" : "",
"TD" : "",
"BR" : "\\\\",
"DL" : "\\begin{description}",
"DT" : "\\item[",
"DD" : "] \\hfill \\\\ ",
"UL" : "\\begin{itemize}",
"LI" : "\\item",
"OL" : "\\begin{enumerate}"}
```

A brief summary of how some special attributes were handled:

- Font Size: HTML default font size is 3. Any size less than that (upto 1) used a smaller LaTeXcommand and any size bigger (upto 7) used a larger LaTeXcommand.

```
font_size = {
  1:'tiny ',
  2:'small ',
  3:'normalsize ',
  4:'large ',
  5:'Large ',
  6:'LARGE ',
  7:'huge ',
  8:'HUGE '}
```

- LaTeXspecial characters were replaced in the string using a dictionary.

```
spec_char = {
  "\\" : "\\textbackslash",
  "$" : "\\$",
  "%" : "\\%",
  "{" : "\\{",
  "}" : "\\}",
  "_" : "\\_",
  "-" : "\\textendash",
  "#" : "\\#",
  "&" : "\\&",
  "^" : "\\^",
  "" :  "
     ",
  "~" : "\\textasciitilde"}
  ...
for x in spec_char.keys():
    text = text.replace(x,spec_char[x])
```

- Handling of empty LaTeX tags vs \begin{ }, \end{ } tags
  Here we define a "end" variable for each node, containing the closing string for the this node. In case of empty LaTeX tags it is "}" and otherwise it is \end { }

```
    if(end == None):
    print("}", file=file,end="")
  else:
    print(end, file=file,end="")
```

Bonus Work

In addition to the tags/attributes mentioned in the assignment, I have implemented the following features.

- Anchor tags to labels within documents

```
    HTML
     <a name="img"></a>
     ...
     <a href="#img">images</a>
```

  Code in python:

```
def handle_anchor(Node,file):
  for x,y in zip(Node.attr,Node.values):
    if(x.upper()=="HREF"):
      link = y
      if(link[0]!='#'):
        print("\\href{"+link+"}{",file=file)
        return "}"
      else:
        link = link.replace('#',"")
        print("\\hyperref["+link+"]{",file=file,end="")
        return "}"
    if(x.upper()=="NAME"):
      link = y
      print("\\label{"+link,file=file,end="")
      return "}"
```

- Table Borders

```
    HTML
    <table border=1>
    ...
```

  Code in python:

```
  border = 0
  if(x.upper() == "BORDER"):
    border = int(y)
  row_str = ['||']
  for x in range(cols):
    row_str.append('c||')
  row_str = "".join(row_str)
  if(border==0):
    row_str = row_str.replace("||"," ")
  print("\\begin{tabular}{"+row_str+"}",file=file)
  if(border==1):
    print("\\hline",file=file)
    print("\\hline",file=file)
  for x in Node.children:
    if(x.value=="TR"):
        ...
    if(border==1):
      print("\\hline",file=file)
      print("\\hline",file=file)
      ...
```

- Special characters
  I defined a dictionary with the special characters (greek, unicode, etc.) from HTML to LATEX

```
spec_char = {
  "&Alpha;" : "$A$",
  "&Beta;" :  "$B$",
  "&Gamma;" : "$\\Gamma$",
  "&Delta;" : "$\\Delta$",
  "&Epsilon;" : "$E$",
  "&Zeta;" :  "$Z$",
  "&Eta;" : "$E$",
  "&Theta;" : "$\\Theta$",
  "&Iota;" :  "$I$",
  "&Kappa;" : "$K$",
  "&Lambda;" :  "$\\Lambda$",
  "&Mu;" :  "$M$",
  "&Nu;" :  "$N$",
  "&Xi;" :  "$\\Xi$",
  "&Omicron;" : "$O$",
  "&Pi;" :  "$\\Pi$",
  "&Rho;" : "$R$",
  "&Sigma;" : "$\\Sigma$",
  "&Tau;" : "$T$",
  "&Upsilon;" : "$\\Upsilon$",
  "&Phi;" : "$\\Phi$",
  "&Chi;" : "$X$",
  "&Psi;" : "$\\Psi$",
  "&Omega;" : "$\\Omega$",
  "&alpha;" : "$\\alpha$",
  "&beta;" :  "$\\beta$",
  "&gamma;" : "$\\gamma$",
  "&delta;" : "$\\delta$",
  "&epsilon;" : "$\\epsilon$",
  "&zeta;" :  "$\\zeta$",
  "&eta;" : "$\\eta$",
  "&theta;" : "$\\theta$",
  "&iota;" :  "$\\iota$",
  "&kappa;" : "$\\kappa$",
  "&lambda;" :  "$\\lambda$",
  "&mu;" :  "$\\mu$",
  "&nu;" :  "$\\nu$",
  "&xi;" :  "$\\xi$",
  "&omicron;" : "$o$",
  "&pi;" :  "$\\pi$",
  "&rho;" : "$\\rho$",
  "&sigma;" : "$\\sigma$",
  "&tau;" : "$\\tau$",
  "&upsilon;" : "$\\upsilon$",
  "&phi;" : "$\\phi$",
  "&chi;" : "$\\chi$",
  "&psi;" : "$\\psi$",
  "&omega;" : "$\\omega$",
  "&thetasym;" :  "$\\vartheta$"
     ,
  "&nbsp;" : "\&",
  "&lt;" : "\\textless",
  "&gt;" : "\\textgreater",
  "&quot;" : "",
  "&apos;" : "",
  "&cent;" : "",
  "&pound;" : "\\pounds",
  "&copy;" : "\\copyright",
  "&reg;" : "\\textregistered"}
```

- Extra table cells
  If number of <th >or <td >are not the same across all table rows in HTML. To handle this, we found the max number of columns across all rows, and created a table of that size. Afterwards, we appended empty cells to the rows which had inadequate number of columns.

```
rows = len(Node.children)
cols = 0
for x in Node.children:
    cols = max(cols,len(x.children))
for x in range(cols):
  row_str.append(’c||’)
  empty_list.append(" ")
row_str = "".join(row_str)
...
    for i in range(len(x.children),cols):
      print("& ",file=file,end="")
```