编译原理 - 作业(1)：词法分析

截至时间：2022.3.15/周二 上课前（14:20）
提交方式：超算习堂（https://easyhpc.net/course/144）

Q1: (p114, Exercise 3.1.2) Tagged languages like HTML or XML are different from conventional programming languages in that the punctuation (tags) are either very numerous (as in HTML) or a user-defined set (as in XML). Further, tags can often have parameters. Suggest how to divide the following HTML document:

Here is a photo of <B>my house</B>:
<P><IMG SRC = “house.gif”><BR>
See <A HREF = “morePix.html”>More Pictures</A> if you liked that one.<P>

into appropriate lexemes. Which lexemes should get associated lexical values, and what should those values be?

<text, "Here is a photo of"> <nodestart, b> <text, "my house"> <nodeend, b>
<nodestart, p> <selfendnode, img> <selfendnode, br>
<text, "see"> <nodestart, a> <text, "More Picture"> <nodeend, a>
<text, "if you liked that one."> <nodeend, p>

Q2: (p125, Exercises 3.3.2) Describe the languages denoted by the following regular expressions:

1) a(a|b)*a
   String of a's and b's that start and end with a.
2) ((ε|a)b*)*
   String of a's and b's.
3) (a|b)*a(a|b)(a|b)
   String of a's and b's that the character third from the last is a.
4) a*ba*ba*ba*
   String of a's and b's that only contains three b.
5) (aa|bb)*((ab|ba)(aa|bb)*(ab|ba)(aa|bb)*)*
   String of a's and b's that has a even number of a and b.

Q3: Write regular expressions for the following languages, or indicate that there exists no such expression:

1) Strings over the alphabet {a, b, c} in which no a’s appear after the first b (if one exists).
   [ab]*(b[bc]*)?
2) Binary numbers that are multiples of 4.
   [01]*00
3) All strings over the alphabet {x, y} that contain no consecutive x’s (including the empty string).
   (x?y)*x?
4) Strings over the alphabet {x, y} that have exactly as many x’s as y’s.
   None exists.

5) Identifiers that start with an uppercase letter and then have one or more alphanumeric characters, ending in a number.
   [a-z][a-zA-Z0-9]*[0-9]

Q4: Consider the following regular expression over the alphabet {a, b}: a*b | aa

1) Use M-Y-T algorithm to make an NFA from the regular expression (show it as a state diagram).
2) Use subset construction to create a DFA equivalent to the NFA you gave for part 1). Show the construction steps and final transition table.
3) Is the DFA in 2) minimized? If yes, explain; otherwise, do the minimization.
   See final_practice Q3. [Note that *a* and *b* are switched.]