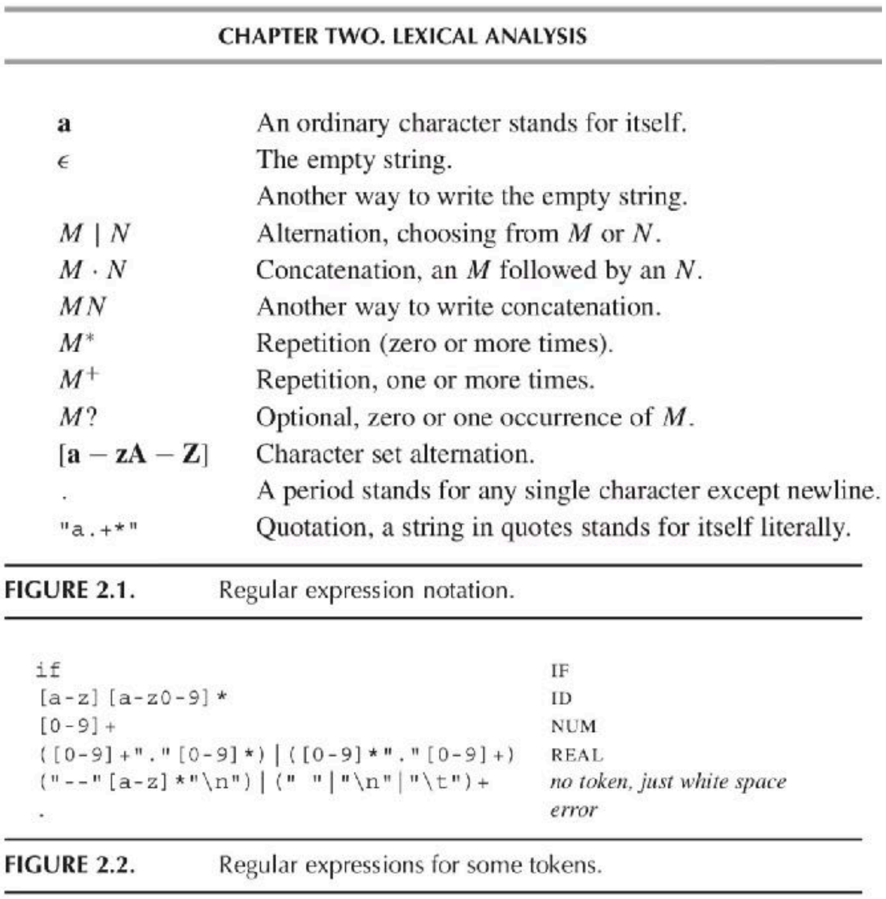
Compilers

CMPT 432

Lab 1

Goals	Making tokens with your Lexer
Notes	We need to figure out how the characters that comprise the source code get turned into tokens that are (hopefully) valid in the language.
Resources	*Crafting a Compiler* • Read chapter 3 • Do exercises 1.11, 3.1 *Dragon* • Read chapter 3 • Do exercises 1.1.4, 1.6.1
Submitting	Use $\LaTeX$ to produce a PDF and commit a PDF of your work to your private GitHub repository and I'll take a look at it.



CHAPTER TWO. LEXICAL ANALYSIS

a	An ordinary character stands for itself.
ϵ	The empty string.
	Another way to write the empty string.
$M \mid N$	Alternation, choosing from M or N.
$M \cdot N$	Concatenation, an M followed by an N.
MN	Another way to write concatenation.
M^*	Repetition (zero or more times).
M^+	Repetition, one or more times.
$M?$	Optional, zero or one occurrence of M.
$[\mathbf{a-zA-Z}]$	Character set alternation.
.	A period stands for any single character except newline.
`"a.+*"`	Quotation, a string in quotes stands for itself literally.

FIGURE 2.1. Regular expression notation.

```
if                                     IF
[a-z][a-z0-9]*                         ID
[0-9]+                                 NUM
([0-9]+"."[0-9]*)|([0-9]*"."[0-9]+)    REAL
("--"[a-z]*"\n")|(" "|"\n"|"\t")+      no token, just white space
.                                      error
```

FIGURE 2.2. Regular expressions for some tokens.

from *Modern Compiler Implementation in Java* by Andrew Appel