1.1)
The machine recognizes any binary string of minimum length 2.
At least two `1` must be input.
Therefore, strings that are not recognized in the alphabet {0,1}* are:
(three dots means repetition of last symbol)
0...
0...1
0...10...

1.2)
\+ means the literal plus symbol, not the “repeat at least once”
As a regex:
(- | ε)[0-9]+(.[0-9]+)((\+|-)[0-9]+(.[0-9]+)i)

As “dragon-convention” (the book):
letter -> i
digit -> [0-9]
prefix -> (- | ε)

real -> digit+(. digit+)
midoperator -> (\+|-)
imaginary -> real[i]

complex_number -> (-)?real (midoperator imaginary)?

1.3)
A regular language can be accepted by DFAs.
DFA have a finite amount of states and no memory.
Lets assume our DFA has n states, n >= 0.

Regular expressions have properties of Dyck Language: an opening bracket (or parenthesis) must have exactly one matching closing bracket.

Assuming our input is larger than the number of states in our DFA, this means that there exists a state which is reached twice.

Since (^n (opening parenthesis n times) followed by)^n is a valid regular expression, this must mean that the DFA has a loop on opening parenthesis.

However, if we input (^n+k (opening parenthesis n+k, k >= 1 times), this will still be recognized by the DFA, but since it has no memory, it does not detect that there is an invalid number of parenthesis.

Thus, if a DFA for regular languages existed, it must accept invalid parenthesis matches, which is a contraction. Thus, regular expressions are not regular languages.