1. A string over the alphabet $\Sigma$ is a finite sequence of characters from $\Sigma$.

(a) These are almost exactly like strings used in programs, just using characters from a specific given alphabet $\Sigma$.

E.g., if $\Sigma = \{a, b, c\}$, and $abc$ and $abbacc$ are strings over $\Sigma$.

(b) $\Sigma$ stands for the empty string, which contains no characters.

(c) $|x|$ denotes the length of the string $x$, so $|\epsilon| = 0$.

2. The product $xy$ of two strings $x$ and $y$ is simply their...

(a) Concatenation, i.e., the sequence of characters from $x$ followed by those from $y$.

(b) The string product is associative: $x(yz) = (xy)z$.

But not commutative.

(e.g., if $x = abc$ and $y = bba$, then $xy = abcbba$ and $yx = bbaabc$)

(c) As with numbers, we can define powers of strings using induction:

\[ z^n = \epsilon \text{, and } z^{n+1} = z^n \cdot z \text{ for } n \geq 0. \]

\[ z^n = \epsilon, z^1 = z, z^2 = z \cdot z, z^3 = z^2 \cdot z, \text{ etc.} \]

(d) $|xy| = |x| + |y|$ and $|x^n| = n |x|$.

Length of concatenation = sum of lengths.

Length of a power of $x$ = number of copies times length of $x$.

3. For a set $W$ of strings, we can define its asterate $W^*$ inductively:

Let $W^0 = \{ \epsilon \}$ and $W^{n+1} = W^n \cdot W$ for $n \geq 0$; then $W^* = \bigcup W^n$.

(a) $W^*$ is the set of all strings that can be built as concatenations of some finite number of strings chosen from $W$ (with replacement — strings can be used more than once!).

E.g., if $W = \{ab, c, da\}$, then $W^*$ contains $\epsilon, abc, ccc, ab, da, abc, \ldots$

(b) A few key properties are that $(W^*)^* = W^*$ and $\epsilon \cdot W^* = W^* \cdot W^*$ for any set $W$.

(c) $\Sigma^*$ is the set of all [finite] strings over $\Sigma$ — so $x \in \Sigma^*$ simply means that $x$ is a string over $\Sigma$. 

4. If \( x \) and \( y \) are strings, their formal sum \( x + y \) represents the set \( \{x, y, x+y\} \) of both strings; we can think of it as collecting a union of some strings on sets of strings.

This \( "+" \) has most of the properties we're used to for sums:

- It's associative: \( (x+y)+z = x+(y+z) \) --- both are \( \{x, y, z, y+z, x+z, x+y+z\} \)
- Commutative: \( x + y = y + x \) --- both are \( \{x, y, x+y\} \)
- With 0 as its identity: \( x + 0 = x \) --- all are \( \{x\} \)

It also has some properties as a union: \( x + x = x \) --- both just have x

1. Products distribute over sums: \( x(\{x^2\}) = xy + xz \) --- concatenating \( x \) and \( \{y, z\} \)
   and
   \( (x+y)z = xz + yz \) --- concatenating \( \{y, z\} \) and \( \{x, y\} \) and \( \{z\} \)

5. A **regular expression** over \( \Sigma \) is any expression built from:
   - Words over \( \Sigma \)
   - Concatenation \( (\cdot) \)
   - Sums \( (+) \)
   - Asterates \( (*) \)
   - And \( (\cdot) \) for concatenation

   Each regular expression generates some set of strings over \( \Sigma \).

6. \( \cdot \) is the empty string, an element of \( \Sigma^* \)

   \( \cdot \) is the empty set, a subset of \( \Sigma^* \) containing no elements

   \( \{\cdot\} \) is a set containing just the empty string; a subset of \( \Sigma^* \) with one element, \( \cdot \)

7. \( \Sigma^* \) is every string that can be built from the strings of \( \Sigma \) except \( \cdot \).

   In other words, it's the nonempty strings we can build from \( \Sigma \).

   If \( \Sigma \neq \emptyset \), this is denoted by \( \Sigma^+ \) (technically, \( \Sigma^+ = \Sigma \cdot \Sigma^* \) for any set \( \Sigma \), but if \( \Sigma \neq \emptyset \), this contains \( \cdot \))
8. (a) \[ a^2 \cdot (bca)^2 = a \cdot a \cdot b \cdot c \cdot a \cdot b \cdot c \cdot a \]

(b) \[ \frac{1}{a + (b + c)} \cdot \omega = \frac{1}{a + b + c} \cdot \omega = a \omega + ab \omega + ac \omega \]

(c) \[ \frac{1}{a + b + c} = \frac{1}{a + (b + c)} = \frac{1}{a + b + c} \]

\[ = b + c \]

(d) \[ (a + xy)^3 = (a + xy)^2 \cdot (a + xy) = (a + xy)^2 \cdot a + (a + xy)^2 \cdot xy \]

\[ = \frac{1}{a + xy + (xy)^2} = a + xy + (xy)^2 \]

\[ = (a + xy) + (a + xy) + (a + xy) + (a + xy) + (a + xy) \]

\[ = (a + xy) + (a + xy) + (a + xy) + (a + xy) + (a + xy) \]

9. (a) \[ 1^n \] gives all strings of 1's, i.e., \[ \{ 1^n : n \geq 0 \} \]

(b) \[ 0^1 \] gives the strings above with a preceding 0, i.e., \[ \{ 01^n : n \geq 0 \} \]

(c) \[ 0^n1 \] gives all strings of 0's with a 1 at the end, i.e., \[ \{ 0^n1 : n \geq 0 \} \]

(d) \[ (01)^n \] gives all powers of 01, i.e., \[ \{ (01)^n : n \geq 0 \} \]

(e) \[ 0^n1 \] gives the concatenations of all strings of 0's with all strings of 1's, i.e., \[ \{ 0^n1^m : n, m \geq 0 \} \]

(c) Note that the # of 0's need not match the # of 1's!

10. (a) This is just the concatenation of 01 with \[ \sum^n = (0+1)^n : \text{concatenate!} \]

(b) Similar to (a), but now we could have any string at the start, as well:

\[ (0+1)^n \text{, } 000 \text{, } 0+1 \text{, } \]

(c) This is just the asterisk of 001: \[ (001)^* \]

(d) A little trick!

This means that we have all of our 1's at the end (any # of them, so 1^n), and thus all 0's at the start (any # of them, so 0^x).

Concatenating, we get \[ 0^x1^n \]