<u>High-level descriptions:</u>

When describing a TM's behavior, we use informal, high-level algorithmic pseudo-code.
We don't worry about what the states are or what the heads are doing (unless explicit in the problem to solve).

TMs = algorithms] Church-Turing thesis

<u>Reasonable</u>
Encodings of finite math objects into (binary) strings:

1. natural numbers, integers
2. strings over arbitrary alphabets (e.g. ASCII $\longrightarrow$ 8 bits per symbol)
3. Lists of encodable things

$$O_1, \ldots, O_n$$
$$\downarrow \qquad \downarrow$$
$$\underbrace{x_1, \ldots, x_n}$$
$$\downarrow$$
$$x_1 \# x_2 \# \cdots \# x_n \quad (\# \text{ does not occur in the } x_i)$$

4. finite sets of encodable objects (rep as a list)
5. trees, graphs
6. DFAs, NFAs, regexes, TMs

What does "reasonable encoding" mean?
Any reasonable basic operation that you would want to perform on the object(s) can be done with a TM on the encoding.

<u>Ex</u>: +, $\times$, ... on integers
lists: find the length
pick out the n'th element (some given n)
graphs: traverse an edge, visit nodes, ...

Given object O, we let $\langle O \rangle$ be the string that encodes it.
Given $O_1, O_2, \ldots, O_n$, let $\langle O_1, \ldots, O_n \rangle$ — encoding of the list of $O_1, \ldots, O_n$ (each encoded)

<u>Ex</u>: Graph reachability
TM M that decides this problem:

M = "On input $\langle G, s, t \rangle$ where G is a graph and s & t are vertices of G:
tacit, optional [0. If input is not of this form, <u>reject</u>.]
1. Do BFS with source s
2. If t ever shows up in this search, then <u>accept</u>.
3. <u>Reject</u>"

A <u>universal TM</u> is one that can simulate the behavior of any TM on any input.

Here is a universal TM:

U := "On input $\langle M, w \rangle$ where M is a TM and w is a string (over M's input alphabet):
1. Run M on input w"

$\underbrace{\text{[and do what M does]}}_{\text{optional}}$

<u>Ex</u>: - If M accepts w, then U accepts $\langle M, w \rangle$
- If M rejects w, then U rejects $\langle M, w \rangle$
- If M loops on input w, then U loops on input $\langle M, w \rangle$ (it must loop!)
- If M computes a function f, then U computes the same function except that U output f(w) (on its own input $\langle M, w \rangle$)
[same output as M on w]

[Inspired the stored program computer]

A timed universal TM:

N := "On input $\langle M, w, t \rangle$ where M is a TM, w is a string, and t is a natural number:
1. Run M on input w for t steps [or until it halts, whichever happens first]
2. If M does not halt in $\le t$ steps, then reject, else do what M does."