Definition 2.1: Formally, a Turing machine is a quadruple $M = (K, \Sigma, \delta, s)$. Here K is a finite set of *states* (these are the instructions alluded to above); $s \in K$ is the *initial state*. Σ is a finite set of *symbols* (we say Σ is the *alphabet* of M). We assume that K and Σ are disjoint sets. Σ always contains the special symbols $\sqcup$ and $\triangleright$: The *blank* and the *first symbol*. Finally, δ is a *transition function*, which maps $K \times \Sigma$ to $(K \cup \{h, \text{"yes"}, \text{"no"}\}) \times \Sigma \times \{\leftarrow, \rightarrow, -\}$. We assume that h (the *halting state*), "yes" (the *accepting state*), "no" (the *rejecting state*), and the *cursor directions* $\leftarrow$ for "left," $\rightarrow$ for "right," and $-$ for "stay," are not in $K \cup \Sigma$. □

Function δ is the "program" of the machine. It specifies, for each combination of current state $q \in K$ and current symbol $\sigma \in \Sigma$, a triple $\delta(q, \sigma) = (p, \rho, D)$. p is the next state, ρ is the symbol to be overwritten on σ, and $D \in \{\leftarrow, \rightarrow, -\}$ is the direction in which the cursor will move. For $\triangleright$ we require that, if for states q and p, $\delta(q, \triangleright) = (p, \rho, D)$, then $\rho = \triangleright$ and $D = \rightarrow$. That is, $\triangleright$ always directs the cursor to the right, and is never erased.

How is the program to start? Initially the state is s. The string is initialized to a $\triangleright$, followed by a finitely long string $x \in (\Sigma - \{\sqcup\})^*$. We say that x is the *input* of the Turing machine. The cursor is pointing to the first symbol, always a $\triangleright$.

From this initial configuration the machine takes a step according to δ, changing its state, printing a symbol, and moving the cursor; then it takes another step, and another. Note that, by our requirement on $\delta(p, \triangleright)$, the string will always start with a $\triangleright$, and thus the cursor will never "fall off" the left end of the string.

Although the cursor will never fall off the left end, it will often wander off the right end of the string. In this case we think that the cursor scans a $\sqcup$, which of course may be overwritten immediately. This is how the string becomes longer—a necessary feature, if we wish our machines to perform general computation. The string never becomes shorter.

Since δ is a completely specified function, and the cursor never falls off the left end, there is only one reason why the machine cannot continue: One of the three halting states h, "yes", and "no" has been reached. If this happens, we say that the machine has *halted*. Furthermore, if state "yes" has been reached, we say the machine *accepts* its input; if "no" has been reached, then it *rejects* its input. If a machine halts on input x, we can define the *output* of the machine M on x, denoted $M(x)$. If M accepts or rejects x, then $M(x) =$ "yes" or "no", respectively. Otherwise, if h was reached, then the output is the string of M at the time of halting. Since the computation has gone on for finitely many steps, the string consists of a $\triangleright$, followed by a finite string y, whose last symbol is not a $\sqcup$, possibly followed by a string of $\sqcup$s (y could be empty). *We consider string y to be the output of the computation*, and write $M(x) = y$. Naturally, it is possible that M will never halt on input x. If this is the case we write $M(x) = \nearrow$.

Taken From -

Papadimitriou, Christos H.. Computational Complexity. United Kingdom, Addison-Wesley, 1993.