1. Give an algorithm that, given a DFA, constructs a Turing Machine that decides the same language. In other words, give an algorithm that “converts” a DFA to a decider TM.

Illustrate your algorithm on the following example:

We use the states in the DFA along with two extra states $q_{accept}$ and $q_{reject}$. The initial state of the DFA will be the initial state of the TM. For every symbol $c$ in the input alphabet, change all transitions labelled $c$ to $c \rightarrow R$. For every accepting state in the DFA, add a transition labelled $\square \rightarrow R$ to $q_{accept}$. For every non-accepting state in the DFA, add a transition $\square \rightarrow R$ to $q_{reject}$.
2. Exhibit a derivation of the string $aabcbc$ in the following grammar:

$$
\begin{align*}
S & \to ABCS \mid \epsilon \\
CB & \to BC \\
CA & \to AC \\
BA & \to AB \\
A & \to a \\
B & \to b \\
C & \to c
\end{align*}
$$

What language does this grammar generate?

$$
S \Rightarrow ABCS \Rightarrow ABCABC \Rightarrow AABCB \Rightarrow AABCB \Rightarrow \ast \ aabcbc
$$

The grammar generates a proper subset of $\{ w \in \{a,b,c\}^* \mid \#_a(w) = \#_b(w) = \#_c(w) \}$. Every string $w$ in $\mathcal{L}(G)$ has the following additional property:

$$
\forall w' : \text{if } w' \text{ is a prefix of } w, \text{ then } \#_a(w') \geq \#_b(w') \geq \#_c(w')
$$

Thus

$$
L(G) = \{ w \in \{a,b,c\}^* \mid \#_a(w) = \#_b(w) = \#_c(w) \land \forall \text{ prefixes } w' \text{ of } w : \#_a(w') \geq \#_b(w') \geq \#_c(w') \}
$$