2.4 Give context-free grammars that generate the following languages. In all parts, the alphabet $\sum$ is $\{0, 1\}$.

b. $\{w|w$ starts and ends with the same symbol$\}$

Solution. $S \to 0T0|1T1$
$T \to TT|0|1|\epsilon$ ■

c. $\{w|$ the length of w is odd$\}$

Solution. $S \to ST|0|1$
$T \to 00|01|10|11|\epsilon$ ■

2.9 Give a context-free grammar that generates the language
$A = \{a^i b^j c^k | i = j \text{ or } j = k \text{ where } i, j, k \geq 0\}$.
Is your grammar ambiguous? Why or why not?

Solution. We can have a CFG $G = (\{S, T, U, V, W\}, \{a, b, c\}, R, S)$ with these rules:

$$S \to UV|TW$$
$$T \to aT|\epsilon$$
$$U \to aUb|\epsilon$$
$$V \to cV|\epsilon$$
$$W \to bWc|\epsilon$$

This grammar is ambiguous because if we choose to have the string abc, we have two derivations:
$S \to UV \to aUbV \to abV \to abcV \to abc$
and
$S \to TW \to aTW \to aW \to abWc \to abc$ ■

2.10 Give an informal description of a pushdown automaton that recognizes the language A in Exercise 2.9.

Solution. When we begin the pushdown automaton, we start by pushing \$ into our stack and the automaton will make a decision on whether we need to push a into the stack or not. The decision then leads to two branches that will check either if the number of a's and b's are equal or the number

of b's and c's are equal.
Assume we are inputting n number of a's, which means that we need to input the same n number of b's. In our stack, we would have n number of a's being pushed into the stack. We then proceed to inputting b's, popping a's out of the stack depending on the number of times we input b. Lastly, to check if we can have an accepted state, we check to see if we can pop \$ out the stack. If \$ is on top of the stack, we can pop it out and have an accepted state, stating that we have an equal number of a's and b's. If \$ is not on top of the stack, this would mean that we do not have an equal number of a's and b's and we do not have an accepted state. Assuming we do have an equal number of a's and b's, we do not need to do anything with the stack when inputting c's since we satisfied the language's rules.
Assume we want to have the same number of b's and c's in this scenario. We can ignore pushing a's into the stack since we are not accounting for that in this scenario. We do push b's into the stack the same number of times we take b's as input and when we input c's, we would pop the b's out of the stack according to the number of times we input c's. Same as before, if we are able to pop \$ at the end, then the state would be accepted, and if we cannot pop \$, the state would not be accepted.
■

2.14 Convert the following CFG into an equivalent CFG in Chomsky normal form, using the procedure given in Theorem 2.9.
$A \to BAB|B|\epsilon$
$B \to 00|\epsilon$

Solution. Add a new start variable S and the rule $S \to A$ into the CFG:

$$\begin{gathered} S \to A \\ A \to BAB|B|\epsilon \\ B \to 00|\epsilon \end{gathered}$$

Next is to remove the ϵ rules:
Removing $B \to \epsilon$ gives us:

$$\begin{gathered} S \to A \\ A \to BAB|B|AB|BA|A|\epsilon \\ B \to 00 \end{gathered}$$

Removing $A \to \epsilon$ give us:

$$\begin{gathered} S \to A|\epsilon \\ A \to BAB|B|AB|BA|A|BB \\ B \to 00 \end{gathered}$$

Next is to remove unit rules:
Removing $A \to A$ gives us:

$$\begin{gathered} S \to A|\epsilon \\ A \to BAB|B|AB|BA|BB \\ B \to 00 \end{gathered}$$

Removing $S \to B$ gives us:

$$S \to A|\epsilon$$
$$A \to BAB|00|AB|BA|BB$$
$$B \to 00$$

Removing $S \to A$ gives us:

$$S \to BAB|00|AB|BA|BB|\epsilon$$
$$A \to BAB|00|AB|BA|BB$$
$$B \to 00$$

Next is to convert the remaining rules into proper form:

$$S \to BD|CC|AB|BA|BB|\epsilon$$
$$A \to BD|CC|AB|BA|BB$$
$$B \to CC$$
$$C \to 0$$
$$D \to AB$$

■