Name ____________________ Student No. ____________________

No aids allowed. Answer all questions on test paper. Use backs of sheets for scratch work.

Total Marks: 50

[10] 1. Let L_{rep} be the following language: $L_{\text{rep}} = \{ww | w \in \{0,1\}^*\}$. Thus, L_{rep} is the language of *repeated* strings. For example, $011011, 0000 \in L_{\text{rep}}$, but $01, 111 \notin L_{\text{rep}}$.

Show that $\overline{L_{\text{rep}}} = \Sigma^* - L_{\text{rep}}$ is a Context-Free Language.

Solution: As was shown in class, you break up producing the strings into two cases: odd and even:

$$S \longrightarrow O|E \tag{1}$$

and

$$O \longrightarrow 00O|01O|10O|11O|0|1 \tag{2}$$

Note that no string of odd length can be in L_{rep}.

Now we deal with even-length strings. To ensure that we produce only v's such that $v \neq ww$, we have to force the following condition: there is an i, $1 \leq i \leq |v|/2$ such that the i-th position of v contains 1 and the $|v|/2 + i$ position contains 0, or vice-versa.

The two sub-cases can be obtained with the rules:

$$\begin{aligned} E &\longrightarrow AB|BA \\ A &\longrightarrow XAX|1 \\ B &\longrightarrow XBX|0 \\ X &\longrightarrow 1|0 \end{aligned} \tag{3}$$

Putting together the productions (1,2,3), we obtain a CFG for the language $\overline{L_{\text{rep}}}$ which shows that it is a CFL.

[10] 2. Show that the language $L_{\mathrm{abc}} = \{a^n b^n c^n | n \geq 1\}$ is *not* a Context-Free Language.

Solution: An application of the Pumping Lemma for CFLs (given in Example 7.19 in the textbook) shows that L_{abc} is not a CFL.

[10] 3. Consider the following two languages:

$$L_1 = \{a^n b^n c^i | n, i \geq 1\} \subseteq \{a, b, c\}^*$$
$$L_2 = \{a^i b^n c^n | n, i \geq 1\} \subseteq \{a, b, c\}^*$$

(a) Show that L_1, L_2 are Context-Free Languages.

Solution: The grammar for L_1 has the following rules:

$$S \longrightarrow PC$$
$$P \longrightarrow aPb|ab$$
$$C \longrightarrow cC|c$$

The grammar for L_2 is analogous. Since L_1, L_2 have CFGs, they are CFLs.

(b) Using questions 2 and 3(a) show that Context-Free Languages are *not* closed under *intersection*.

That is, show that if L_1, L_2 are Context-Free Languages, then $L_1 \cap L_2$ is *not necessarily* a Context-Free Language.

(**Note:** Do this question even if you did not answer 2 and 3(a).)

Solutions: In (a) we showed that L_1, L_2 are CFLs. On the other hand, $L_1 \cap L_2 = \{a^n b^n c^n | n \geq 1\}$ which has been shown not to be a CFL in question 2.

[10] 4. (a) Suppose that $G = (V, T, P, S)$ is a Context-Free Grammar. Define what it means for a symbol in $V \cup T$ to be *generating*, and what it means to be *reachable*.

Solution: A symbol A is generating if $A \overset{*}{\Rightarrow} w$ for some $w \in T^*$. A symbol A is reachable if there exists a derivation $S \overset{*}{\Rightarrow} \alpha A \beta$.

(b) A symbol is *useful* if it is both generating and reachable. Give an algorithm, which on input G (a Context-Free Grammar) finds all the useful symbols of G.

Solution: We find all the generating symbols, and then all the reachable symbols, and those symbols which turn out to be both, are useful.

To find the generating symbols, first mark all the symbols in T as generating. Then repeat the following step, for as long as *new* generating symbols are found: if every symbol in α has been marked as generating, and $A \longrightarrow \alpha$ is a rule, then mark A as generating. This procedure terminates, because there are finitely many symbols in $V \cup T$, so it terminates in at most $|V \cup T|$ many "steps."

To find the reachable symbols, mark S as reachable. Now repeat the following step for as long as *new* reachable symbols are found: if $A \longrightarrow \alpha$ is a rule, and A has been marked as reachable, then mark every symbol in α as reachable. This procedure terminates for the same reason as the above procedure.

[10] 5. Design a Pushdown Automaton (a PDA) for the Context-Free Grammar

$$G_{\text{pal}} = (\{P\}, \{0,1\}, \{P \longrightarrow \varepsilon|0|1|0P0|1P1\}, P).$$

Solution: The PDA P has input symbols $\{0,1\}$, and stack symbols $\{0,1,P\}$, just one state q, it accepts its input by empty stack, and it has the following transition function:

$$\begin{aligned}
&\delta(q,1,1) = \delta(q,0,0) = (q,\varepsilon)\\
&\delta(q,\varepsilon,P) = \{(q,\varepsilon),(q,0),(q,1),(q,0P0),(q,1P1)\}
\end{aligned}$$

Note that $\delta(q,1,0) = \delta(q,0,1) = \emptyset$, which signifies that the next input symbol and the stack do not agree, so this branch of the computation dies.

[5] 6. **BONUS QUESTION**

(You don't have to answer this question to get full marks.)

Consider a modified class of PDAs where we only allow stacks of a fixed depth d. Call this class BDPDAs (for Bounded-Depth PDAs).

If A is a BDPDA, and A has already d symbols in its stack, then A cannot push any more symbols onto the stack.

What class of languages can be described by BDPDAs?

Solution: Regular Languages. The reason is that a finite stack of max depth d can be simulated with $\sum_{i=0}^{d} |\Gamma|^i$ many states; that is, state q_α indicates that α is in the stack. There is a finite number of such αs (given by the summation above). Hence, a BDPDA can be simulated with an ε-NFA with states $Q \times \{q_\alpha | \alpha \in \Gamma^*, |\alpha| \leq d\}$.