Name ______________________ Student No. ______________________

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

All questions worth 20, for a total of 100.

1. Show that the class of context-free languages is closed under the regular operations, union, concatenation, and star. That is, show that if $L_1, L_2$ are CFLs, then so are $L_1 \cup L_2, L_1 L_2, L_1^*$.

   **Solution:** Suppose that $G_1, G_2$ are the grammars for $L_1, L_2$. For $L_1 \cup L_2$, let $S \to S_1 | S_2$, and for $L_1 L_2$, let $S \to S_1 S_2$, and for $L_1^*$, let $S \to S S_1 | \varepsilon$.

2. Show that the language $L = \{0^n\#0^{2n}\#0^{3n} | n \geq 0\}$ is not context-free.

   **Solution:** Problem 2.30b in the textbook.

3. Let $G$ be a CFG in Chomsky normal form that contains $b$ variables, i.e., $|V| = b$. Show that if $G$ generates some string with a derivation having at least $2^b$ many steps, then $L(G)$ is infinite.

   (**Hint:** Show first that if $|w| = n$, and $G \stackrel{k}{\Rightarrow} w$, with $k \geq 2^b$, then $n \geq \frac{2^b+1}{2} > 2^{b-1}$.)

   **Solution:** We know that if $G$ is in CNF, then if $|w| = n$, $w \in L(G)$, then *any* derivation of $w$ requires exactly $2n - 1$ steps. This is because each rule of the form $A \to BC$ increases the length of the string by 1, so there are $n - 1$ such applications; then, we replace each variable by a terminal, with $n$ more applications.

   So, a derivation with at least $2^b$ many steps must produce a string of length at least $\frac{2^b+1}{2} > 2^{b-1}$.

   Now consider a parse tree for $w$. To generate a string of length at least $2^b$, the parse tree must be of height at least $b + 1$, because each node has at most 2 children. To see this, note that we must have a tree of height $b$ to generate all the variables that will turn into terminals; then we add 1 for the terminals.

   As in the Pumping Lemma, we can conclude that some variable must be repeated on some path in this tree. Now divide $w$ into $uvxyz$ (again, as in the Pumping Lemma), and it follows that $uv^ixy^iz$ must be in the language for all $i \in \mathbb{N}$, and so $L(G)$ must be infinite.

4. Let a $k$-PDA be a pushdown automaton equipped with $k$ stacks. Define the transition function in the obvious way (meaning that on each pair $(q, a)$ we go to $(q, b_1, b_2, \ldots, b_k)$, where $b_i$ denotes the action on the $i$-th stack). A 0-PDA is just an NFA; a 1-PDA is the conventional PDA. Show the following:

   (a) A 1-PDA is more powerful than a 0-PDA.

   (b) A 2-PDA is more powerful than a 1-PDA.

   (c) A $k$-PDA with $k \geq 3$ is just as powerful as a 2-PDA.

   ("more powerful" means recognizes more languages).

   **Solution:** 0-PDAs correspond to regular languages, which are strictly contained in context-free languages which correspond to 1-PDAs.

   A 2-PDA can recognize the language $\{ww | w \in \{0,1\}^*\}$ which we showed (Pumping Lemma) is not context-free; push the first half of an input on the first stack, guess the middle, and then push the second half of the input on the second stack, and then on $\varepsilon$'s pop the two stacks simultaneously and compare.

   In fact, 2-PDAs correspond to Turing machines. We can simulate an arbitrary TM with two stacks as follows: we can use two stacks to keep track of the configurations of the TM. Each configuration is of the form $vqw$, so we keep $v$ on the first stack, and $w$ on the second, and we are in state $q$. We simulate a transition by pushing and popping. We start with $q_0 w$ and halt when we encounter a halting state.

   On the other hand, any $k$-PDAs can be simulated by a TM, since TMs are the most general model of computation.

5. Show that a language $L$ is decidable iff there exists an enumerator $E$ which enumerates all the strings in $L$ in order of length. (That is, $E$ first enumerates all strings of length 1, then all strings of length 2, then all strings of length 3, etc.)

**Solution:** [$\Longrightarrow$] let $M$ be $L$'s decider. We construct $E$ as follows: $E$ examines all the strings in the alphabet in lexicographic order, and runs them on $M$, one by one, and outputs a string iff $M$ accepts it.

[$\Longleftarrow$] let $E$ be the enumerator, and define $M$ as follows: on input $w$, $M$ simulates $E$ for as long as either $E$ outputs $w$, in which case $M$ accepts, or $E$ starts listing strings of length $> |w|$, in which case $M$ rejects. (If $E$ never lists a string of length $> |w|$, then $L$ is finite, in which case it is decidable—in fact regular!)