1. (a) The language $L_1$ is context free but not regular. To see it is context free, consider the NPDA that first pushes a’s onto the stack until it sees the first b, then pops a’s from the stack as it reads b’s, then when the stack becomes empty, it pushes b’s onto the stack as it reads b’s, and finally, it pops b’s from the stack as it reads c’s.

To see it is not regular, we use the pumping lemma: let $w = a^p b^{2p} c^p$, and consider the ways $w$ can be written as $w = xyz$. If $y$ straddles the boundary between characters, then pumping results in an out-of-order string (not in the language). And, if $y$ is within any single type of character, then pumping on it results in a string not in the language.

(b) The language $L_2$ is not context free. We use the CFL pumping lemma: let $w = a^p b^{2p} c^p$, and consider the ways $w$ can be written as $w = uvxyz$. If either $v$ or $y$ straddles the boundary between characters, then pumping results in an out-of-order string (not in the language). Also, since $|vxy| \leq p$, it must be that either $v$ and $y$ are in the a’s and b’s, or $v$ and $y$ are in the b’s and c’s. If both are in the a’s or both in the c’s, then clearly pumping results in a string not in the language. If $v$ is in the a’s and $y$ is in the b’s then after pumping $i + 1$ times, we get $a^{p + v}|i| b^{2p + |v||i|} c^p$. And indeed, $(p + |v||i|)p = p^2 + |y||i|$ implies $p|v|i = |y||i|$ which implies $|y| = p|v|$, which would make $|vxy| > p$. A symmetric argument holds for the case of $v$ and $y$ being in the b’s and c’s.

(c) Language $L_3$ is regular. Let $T$ be the regular expression $\Sigma^{3001}(\Sigma \cup \epsilon)^*$ (which are those strings of length greater than 3000). We know that regular languages are closed under intersection, so $L'_3 = a^* b^* c^* \cap L(T)$ is regular. Thus the language of all strings in $a^* b^* c^*$ with at least one of each character, and length greater than 3000, is regular. By the pigeonhole principle, any such string must have more than 1000 of one of the three characters. Since it also has at least one of each character, we have that $ijk > 1000$ for these strings.

Now, we observe that $L_3$ is nothing more than $L'_3$ plus a finite number of additional strings, namely some subset of the strings $a^i b^i c^i$ for $i = 0, 1, 2, \ldots, 1000$ and some other strings of length at most 3000. The union of the regular language $L'_3$ with a finite number of strings is still regular (since regular languages are closed under union).

2. (a) Decidable. We will reduce this problem to $E_{CFG}$ (emptiness of context-free-grammars), which we saw in lecture was decidable. Given $E$ we first build the DFA that recognizes language $A$, the complement of $L(E)$. This is possible because regular languages are closed under complement. We also know how to construct a NPDA that recognizes the language $B = L(G) \cap A$ (from the hint: Sipser problem 2.18). We now check if $B$ is empty. From lecture we know that emptiness of CFGs is decidable. Moreover the complementation step and the intersection step are all computable transformations. Finally, note that $B$ is empty iff $L(G) \subseteq L(E)$, so the language CFG-IN-REG is decidable.
(b) Undecidable. We reduce ALLCFG to REG-IN-CFG. Set \( E = \Sigma^* \). Given an instance \( G \) of ALLCFG, we produce the pair \((E, G)\). If \( G = \Sigma^* \) then clearly \( L(E) \subseteq L(G) \); if \( G \neq \Sigma^* \) then \( L(E) \not\subseteq L(G) \). Therefore we have reduced ALLCFG to REG-IN-CFG, and we know from lecture that ALLCFG is undecidable.

3. Suppose there exists a decidable language \( D \) such that \( L_1 \cap D = \emptyset \) and \( L_2 \subseteq D \), with a corresponding TM \( M_D \). Then considering \( M_D(\langle M_D \rangle) \) we come to a contradiction as follows.

Suppose \( M_D(\langle M_D \rangle) \) accepts; i.e. \( \langle M_D \rangle \) is in the language \( D \). Then by the definition of \( L_1 \), \( \langle M_D \rangle \) is in the language \( L_1 \), which contradicts the fact that \( L_1 \cap D = \emptyset \).

Suppose \( M_D(\langle M_D \rangle) \) rejects; i.e. \( \langle M_D \rangle \) is not in the language \( D \). Then by the definition of \( L_2 \), \( \langle M_D \rangle \) is in the language \( L_2 \), which contradicts the fact that \( L_2 \subseteq D \).

4. (a) Let \( G \) be a right-linear CFG. We will construct a NFA \( M \) recognizing \( L(G) \). Our machine \( M \) will have a single state for each non-terminal in the grammar, a distinguished “accept” state, and other states. The start state of \( M \) is the state corresponding to the start symbol in the grammar. For each transition of the form:

\[ A \rightarrow x_1x_2 \ldots x_nB \]

we add \( n - 1 \) states \( s_1, s_2, \ldots, s_{n-1} \) “linking” \( A \) to \( B \), with a transition from \( A \) to \( s_1 \) labelled \( x_1 \), a transition from \( s_1 \) to \( s_2 \) labelled \( x_2 \), etc..., and a transition from \( s_{n-1} \) to \( B \) labelled \( x_n \).

For each transition of the form:

\[ A \rightarrow x_1x_2 \ldots x_n \]

we add \( n - 1 \) states \( s_1, s_2, \ldots, s_{n-1} \) “linking” \( A \) to the accept state, with a transition from \( A \) to \( s_1 \) labelled \( x_1 \), a transition from \( s_1 \) to \( s_2 \) labelled \( x_2 \), etc..., and a transition from \( s_{n-1} \) to the accept state labelled \( x_n \).

Now, if \( M \) accepts a string \( w \), then the sequence of “non-terminal” states it traverses to reach the accept state dictates a derivation of \( w \) in the grammar. In the other direction, if \( w \) has a derivation in the grammar, then it must arise from applying a sequence of rules of the first type, followed by a single application of a rule of the second type. This derivation dictates a path from the start state of \( M \) to the accept state, and thus \( M \) accepts \( w \).

(b) Given a FA \( M \), we construct a right-linear CFG \( G \) as follows. The non-terminals of \( G \) are exactly the states of \( M \). The start symbol of \( G \) is the start state of \( M \). For each transition in \( M \) from state \( A \) to state \( B \), labelled with the symbol \( x \), we add the following rule: \( A \rightarrow xB \). For each transition from state \( A \) to an accept state \( B \), labelled with the symbol \( x \), we add the rule \( A \rightarrow x \).

If \( M \) accepts a string \( w \), then the sequence of states traversed from the start state to an accept state dictates a derivation of \( w \) in the grammar. In the other direction, if \( w \) has a derivation in the grammar, then this derivation dictates a path from the start state of \( M \) to an accept state (since it must end with a rule of the second type).
(c) Consider the following linear CFG $G$:

$$
S \rightarrow 0T | \epsilon \\
T \rightarrow S1
$$

We claim that $L(G) = \{0^n1^n : n \geq 0\}$ (which is not regular as seen in class, so the “linear” constraint on CFGs is not sufficient to force the language to be regular). We first show that all strings of this form are generated by $G$. We prove this by induction on $n$: assume all strings in $L$ of length $< n$ are derivable; the base case with $n = 0$ is trivially true, and then to derive string $0^n1^n$, we use the derivation $S \Rightarrow 0T \Rightarrow 0S1 \Rightarrow^* 00^n1^{n-1}1$, where the last step is possible by induction.

In the other direction, we prove by induction on the length of the derivation strings derivable from $S$ are in $L$. Our induction hypothesis is the stronger statement: all strings derivable in $< n$ steps from $S$ are in $L$, and all strings derivable in $< n$ steps from $T$ are of the form $0^{m-1}1^m$. Clearly the base case (derivation of length 1, which can only derive $\epsilon$) is true. Now consider a derivation of length $n$. If the first step is $S \rightarrow 0T$, then we know by induction that in the remainder of the derivation $0T \Rightarrow^* 00^{m-1}1^m$ for some $m$, so $S$ derives a string in the language as required. If the first step is $T \rightarrow S1$, then we know by induction that in the remainder of the derivation $S1 \Rightarrow^* 0^m1^m1$ for some $m$, so $T$ derives a string of the required form.

5. Let $M$ be a recognizer for $L$. We are given an input $x_1#x_2# \cdots #x_k#$ for some $k \geq 0$. We simulate $M$ on each $x_i$ in parallel, and accept as soon as more than $k/2$ of these simulations accept. Specifically, we do the following for $j = 1, 2, 3, \ldots$: simulate $M$ on each $x_i$ for $j$ steps, and if more than $k/2$ of the simulations halt and accept, then we halt and accept.

Now, it is clear that if a majority of the $x_i$ are in $L$, then we will accept by the time $j$ is the maximum of the number of steps required for $M$ to accept the $x_i$ that are in $L$ (or before). Otherwise, we will never experience more than $k/2$ acceptances among the simulations, for any $j$, so we will never accept.