A. Let \( A, B \in P \). Let \( M, N \) be deterministic Turing machines that decide \( A, B \) in time \( O(n^k), O(n^\ell) \), respectively. Define a Turing machine \( K \) that, on input \( w = w_1 \cdots w_n \), does the following.

1. For \( i = 0, \ldots, n \):
   (a) \( K \) runs \( M \) on \( w_1 \cdots w_i \).
   (b) \( K \) runs \( N \) on \( w_{i+1} \cdots w_n \).
   (c) If both \( M \) and \( N \) accept, then \( K \) accepts; otherwise, \( K \) continues.

2. If \( K \) has not accepted by now, then \( K \) rejects.

Thus \( K \) accepts a string \( w \) if and only if that string can be expressed as the concatenation of a string in \( A \) and a string in \( B \). The running time of \( K \) on an input of length \( n \) is no worse than

\[
    n \cdot \left( O(n^k) + O(n^\ell) \right) = O(n^{1 + \max(k, \ell)}).
\]

So \( K \) is polynomial-time. Thus \( P \) is closed under concatenation.

B. We know from class and the textbook that \( \text{NP} \subseteq \text{PSPACE} \). So, if any language in \( \text{PSPACE} \) is reducible to \( B \), then any language in \( \text{NP} \) is reducible to \( B \).

(If you don’t remember that \( \text{NP} \subseteq \text{PSPACE} \), here’s how you could figure it out. We know that \( \text{NP} \subseteq \text{NPSPACE} \), because a Turing machine cannot use more space than it uses time. We know that \( \text{NPSPACE} \subseteq \text{PSPACE} \), because of the theorem that says that simulating a nondeterministic Turing machine with a deterministic one causes only a quadratic blowup in the space required.)

C. You’ve proved in your homework that \( A \) is not decidable, probably by mimicking our proof that \( K(x) \) is not computable. Essentially the same proof shows that \( A \) is not recognizable.

Suppose that \( M \) is a recognizer for \( A \). Define a Turing machine \( N \) that, on input \( n \) (regarded as an integer in binary), outputs a string \( x \) such that \( K(x) \geq n \). \( N \) can do this by running \( M \) on all strings of length \( n \), in parallel. At least one of these strings \( x \) is incompressible. Eventually, \( M \) will eventually tell \( N \) that \( x \) is incompressible. At that point, \( N \) stops running \( M \) and outputs \( x \). Let \( m \) be any integer large enough that

\[
    m - \lfloor \log_2 m \rfloor - 1 > |N| + |\#|.
\]

Let \( x = N(m) \). Then \( N\#m \) is a description of \( x \), of length less than \( m \), so \( K(x) < m \). But the definition of \( N \) guarantees that \( K(x) \geq m \). This contradiction shows that \( A \) cannot be recognizable.
D. Let \( D \) be a regular expression matching any digit. Let \( L \) match any letter, and let \( A \) match any character other than carriage returns. Let \( C \) be the carriage return character (usually written \( \backslash n \) or \( \backslash r \) in programming languages). Let \( \cdot \) denote a space. Then an addressee is

\[
AA^*,
\]

(without the comma), a street address is

\[
DD^* \cdot AA^* \cdot
\]

a P.O. box is

\[
PO_{\cdot}Box \cdot DD^*,
\]

and a valid third line is

\[
AA^*, LL^* DDDD(\epsilon \cup −DDDD).
\]

So a valid postal address is

\[
AA^*C(DD^* \cdot AA^* \cup PO_{\cdot}Box \cdot DD^*)CAA^*, LL^* DDDD(\epsilon \cup −DDDD).
\]

E. Assume (for the sake of contradiction) that \( A \) is a CFL. Let \( p \) be the pumping length guaranteed to exist for \( A \) by the pumping lemma for CFLs. Let

\[
s = 1^p0^p \cdot \#1^p0^p.
\]

Then \( s \in A \), so the pumping lemma guarantees that \( s = uvxyz \) for strings \( u, v, x, y, z \) such that \( |vxy| \leq p \), \( |vy| \geq 1 \), and \( uv^ixy^iz \in A \) for all \( i \geq 0 \). There are several cases.

- If \( vxy \) is contained in the \( 1^p0^p \) on the left-hand side, then we can pump up to make the left-hand side longer than the right-hand side. In particular, \( uv^2xy^2z \notin A \).

- Similarly, if \( vxy \) is contained in the \( 1^p0^p \) on the right-hand side, then we can pump down to make the right-hand side shorter than the left-hand side: \( uxz \notin A \).

- The only remaining case has \( \# \in vxy \). Then clearly \( \# \notin x \), or else we could pump \( v \) and \( y \) to wreck the number of \( \# \)'s. Because \( |vxy| \leq p \), we know that \( v = 0^k \) and \( y = 1^\ell \) for some \( k, \ell \leq p \). We must have \( k > 0 \), because otherwise pumping \( v \) and \( y \) would just mean pumping \( y \), and we could pump down to make the right-hand side shorter than the left-hand side. But then, because \( v \) is a nonempty string of 0's, pumping up \( v \) and \( y \) causes the left-hand side to have more 0's than does the right-hand side, so we again leave \( A \).
We have shown that, no matter how $u$, $v$, $x$, $y$, and $z$ are arranged, we can pump to leave the language $A$. This contradiction shows that $A$ cannot be context-free.

F. For any Turing machine $M$ and string $w$, define a Turing machine $N$ that, on input $x$, simply runs $M$ on $w$ and then accepts. So, if $M$ halts on $w$, then $N$ accepts all strings $x$, and $L(N)$ is infinite. On the other hand, if $M$ does not halt on $w$, then $N$ does not halt on any input $x$, so $L(N) = \emptyset$ is finite. Thus the function that takes $(M, w)$ to $N$ is a computable reduction of $\text{HALT}_{TM}$ to $A$. Because $\text{HALT}_{TM}$ is not recognizable, it follows that $A$ is not recognizable.