A. Let A be the empty set and C the set of all strings over $\{0,1\}$. These languages are matched by the regular expressions $\emptyset$ and $(0 \cup 1)^*$, and hence are regular. Let $B = \{0^n 1^n : n \geq 0\}$, which is not regular, as we've discussed many times. Notice that $A \subseteq B \subseteq C$, as desired.

B. [Remark: In class we've shown that $\{ww : w \in \{0,1\}^*\}$ is not context-free. The solution below is essentially identical: A PDA is incapable of testing whether two strings are equal. In class we've shown also that $\{a^n b^n c^n : n \geq 0\}$ is not context-free. You can mimic that argument, using the string $s = \texttt{\textbackslash s}^p\texttt{\textbackslash n\textbackslash s}^p\texttt{\textbackslash n\textbackslash s}^p\texttt{\textbackslash n}$, to prove that A is not context-free. In fact, that proof is probably simpler than the one given below.]

Assume for the sake of contradiction that A is context-free. Let p be a pumping length for A, and let

$$s = \texttt{\textbackslash t}^p\texttt{\textbackslash s}^p\texttt{\textbackslash n\textbackslash t}^p\texttt{\textbackslash s}^p\texttt{\textbackslash n}.$$

Because $s \in A$ and $|s| \geq p$, s can be pumped. That is, there exist strings u, v, x, y, z such that $s = uvxyz$, $|vxy| \leq p$, $|vy| \geq 1$, and $uv^i xy^i z \in A$ for all $i \geq 0$. There are several cases, based on the location of vxy within s.

If vxy is a substring of the first line $\texttt{\textbackslash t}^p\texttt{\textbackslash s}^p\texttt{\textbackslash n}$, then there are two subcases. If vxy contains the `\n`, then pumping s to uv^2xy^2z adds a new line to the string, with indentation at most $\texttt{\textbackslash s}^{p-1}$. This indentation does not match the indentation of the last line in the string. If vxy does not contain the `\n`, then pumping v and y does not change the number of lines in the string, but does change the indentation of the first line, so that it no longer matches the indentation of the second line. In either subcase, the pumped string is not in A.

Similarly, if vxy is a substring of the second line $\texttt{\textbackslash t}^p\texttt{\textbackslash s}^p\texttt{\textbackslash n}$, then there are two subcases. In both subcases, s can be pumped to leave A.

The remaining case occurs when vxy is a substring of $\texttt{\textbackslash s}^p\texttt{\textbackslash n\textbackslash t}^p$ that contains at least one `\s` and at least one `\t`. Again there are two subcases. If vy contains the `\n`, then pumping up v and y adds a new line with incorrect indentation, so $uv^2xy^2z \notin A$. Otherwise, x contains the `\n`. Pumping v and y changes the number of `\s`s in the first indentation or the number of `\t`s in the second indentation (or both). In either of these subcases, the indentations no longer match, so that the pumped string is not in A.

In each case, we have shown that s can be pumped to leave the language A. This fact contradicts the pumping lemma. Hence A is not context-free.

C. [Remark: This problem is closely related to Problem 2.26 from our textbook.]

If G is an arbitrary context-free grammar, then P's stack may grow arbitrarily large in the process of accepting a string w. For example, let G be the grammar

$$S \to SS|\epsilon|a$$

over the terminal alphabet $\{a\}$. Then the string $w = a$ is in the language of G, but P may add arbitrarily many Ss to its stack, before collapsing all but one of them to ϵs.

Now suppose that G is in Chomsky normal form. If $w = \epsilon$, then P's stack never exceeds height 2. Henceforth assume that $w \neq \epsilon$. As G derives w from the start variable S, consider how the string of variables and terminals evolves from S to w. Each rule of the form $A \to BC$ increases the number of variables in the string by one. Each rule of the form $A \to a$ replaces a variable with a terminal. Hence, the length of the string can never exceed the length of the final product w. As P simulates G on input w, its stack stores exactly the same deriving string, with an additional \$ symbol to mark the bottom of the stack. Thus P will never have stack height greater than $|w| + 1$.

D. Define a DFA M by taking the product of the DFAs M_1 and M_2. Precisely, let $Q_M = Q_{M_1} \times Q_{M_2}$, $\Sigma_M = \Sigma_{M_1} = \Sigma_{M_2}$, $q_{0M} = (q_{0M_1}, q_{0M_2})$, and $\delta_M(q_1, q_2, a) = (\delta_{M_1}(q_1, a), \delta_{M_2}(q_2, a))$, as always. Define

$$F_M = (F_{M_1} \times (Q_{M_2} - F_{M_2})) \cup ((Q_{M_1} - F_{M_1}) \times F_{M_2}).$$

That is, a state (q_1, q_2) in M is a final state if and only if q_1 is a final state of M_1 or q_2 is a final state of M_2, but not both. Thus the language of M is the *symmetric difference* of the languages of M_1 and M_2:

$$L(M) = (L(M_1) \cup L(M_2)) - (L(M_1) \cap L(M_2)).$$

In other words, M accepts a string w if and only if M_1 and M_2 disagree on w.

Notice that $p_1 p_2$ is the number of states in M and hence a pumping length for $L(M)$. Suppose that $w \in L(M)$ has length $|w| \geq p_1 p_2$. By the pumping lemma, w can be pumped down to a shorter string in $L(M)$. If this shorter string also has length $p_1 p_2$ or greater, then it can be pumped down to a still shorter string in $L(M)$. If we repeat this process, we eventually obtain a string $x \in L(M)$ such that $|x| < p_1 p_2$. Thus, if M_1 and M_2 disagree on any string w such that $|w| \geq p_1 p_2$, then they must disagree on some string x such that $|x| < p_1 p_2$. Taking the contrapositive of that statement, we conclude that if M_1 and M_2 agree on all strings x of length less than $p_1 p_2$, then they agree on all strings of length greater than or equal to $p_1 p_2$, and hence on all strings.