We construct a PDA $P$ that recognizes $C$. First it nondeterministically branches to check either of two cases: that $x$ and $y$ differ in length or that they have the same length but differ in some position. Handling the first case is straightforward. To handle the second case, it operates by guessing corresponding positions on which the strings $x$ and $y$ differ, as follows. It reads the input at the same time as it pushes some symbols, say 1s, onto the stack. At some point it nondeterministically guesses a position in $x$ and it records the symbol it is currently reading there in its finite memory and skips to the #. Then it pops the stack while reading symbols from the input until the stack is empty and checks that the symbol it is now currently reading is different from the symbol it had recorded. If so, it accepts.

Here is a more detailed description of $P$'s algorithm. If something goes wrong, for example, popping when the stack is empty, or getting to the end of the input prematurely, $P$ rejects on that branch of the computation.

1. Nondeterministically jump to either 2 or 4.
2. Read and push these symbols until read #. Reject if # never found.
3. Read and pop symbols until the end of the tape. Reject if another # is read or if the stack empties at the same time the end of the input is reached. Otherwise accept.
4. Read next input symbol and push 1 onto stack.
5. Nondeterministically jump to either 4 or 6.
6. Record the current input symbol $a$ in the finite control.
7. Read input symbols until # is read.
8. Read the next symbol and pop the stack.
9. If stack is empty, go to 10, otherwise go to 8.
10. Accept if the current input symbol isn't $a$. Otherwise reject.

Let $M_A$ be a DFA that recognizes $A$, and $M_B$ be a DFA that recognizes $B$. We construct a PDA recognizing $A \circ B$. This PDA simulates $M_A$ on the first part of the string pushing every symbol it reads on the stack until it guesses that it has reached the middle of the input. After that it simulates $M_B$ on the remaining part of the string popping the stack for every symbol it reads. If the stack is empty at the end of the input and both $M_A$ and $M_B$ accepted, the PDA accepts. If something goes wrong, for example, popping when the stack is empty, or getting to the end of the input prematurely, the PDA rejects on that branch of the computation.

Informal description of a PDA that recognizes the CFG in Exercise 2.3:

1. Place the marker symbol $\$ and the start variable $R$ on the stack.
2. Repeat the following steps forever.
3. If the top of stack is the variable $R$, pop it and nondeterministically push either $XRX$ or $S$ into the stack.
4. If the top of stack is the variable $S$, pop it and nondeterministically push either $aTb$ or $bTa$ into the stack.
5. If the top of stack is the variable $T$, pop it and nondeterministically push either $XTX$, $X$, or $\varepsilon$ into the stack.
6. If the top of stack is the variable $X$, pop it and nondeterministically push either $a$ or $b$ into the stack.
7. If the top of stack is a terminal symbol, read the next symbol from the input and compare it to the terminal symbol in the stack. If they match, repeat. If they do not match, reject on this branch of the nondeterminism.
8. If the top of stack is the symbol $\$, enter the accept state. Doing so accepts the input if it has all been read.
2.31 Assume $B$ is context-free and get its pumping length $p$ from the pumping lemma. Let $s = 0^p1^p0^p$. Because $s \in B$, it can be split $s = uvxyz$ satisfying the conditions of the lemma. We consider several cases.

i) If both $v$ and $y$ contain only 0's (or only 1's), then $uv^2xy^2z$ has unequal numbers of 0's and 1's and hence won't be in $B$.

ii) If $v$ contains only 0's and $y$ contains only 1's, or vice versa, then $uv^2xy^2z$ isn't a palindrome and hence won't be in $B$.

iii) If both $v$ and $y$ contain both 0's and 1's, condition 3 is violated so this case cannot occur.

iv) If one of $v$ and $y$ contain both 0's and 1's, then $uv^2xy^2z$ isn't a palindrome and hence won't be in $B$.

Hence $s$ cannot be pumped and contradiction is reached. Therefore $B$ isn't context-free.

2.32 Assume $C$ is context-free and get its pumping length $p$ from the pumping lemma. Let $s = 1^p3^p2^p4^p$. Because $s \in C$, it can be split $s = uvxyz$ satisfying the conditions of the lemma. By condition 3, $vxy$ cannot contain both 1's and 2's, and cannot contain both 3's and 4's. Hence $uv^2xy^2z$ doesn't have equal number of 1's and 2's or of 3's and 4's, and therefore won't be a member of $C$, so $s$ cannot be pumped and contradiction is reached. Therefore $C$ isn't context-free.

2.42 Assume $Y$ is a CFL and let $p$ be the pumping length given by the pumping lemma. Let $s = 1^{p+1}#1^{p+2}# \cdots #1^{3p}$. String $s$ is in $Y$ but we show it cannot be pumped. Let $s = uvxyz$ satisfying the three conditions of the lemma. Consider several cases.

i) If either $v$ or $y$ contain $\#$, the string $uv^3xy^3z$ has two consecutive $t$'s which are equal to each other. Hence that string is not a member of $Y$.

ii) If both $v$ and $y$ contain only 1's, these strings must either lie in the same run of 1's or in consecutive runs of 1's within $s$, by virtue of condition 3. If $v$ lies within the runs from $1^{p+1}$ to $1^{3p}$ then $uv^2xy^2z$ adds at most $p$ 1's to that run so that it will contain the same number of 1's in a higher run. Therefore the resulting string will not be a member of $Y$. If $v$ lies within the runs from $1^{3p+1}$ to $1^{5p}$ then $uv^0xy^0z$ subtracts at most $p$ 1's to that run so that it will contain the same number of 1's in a lower run. Therefore the resulting string will not be a member of $Y$.

The string $s$ isn't pumpable and therefore doesn't satisfy the conditions of the pumping lemma, so a contradiction has occurred. Hence $Y$ is not a CFL.