1. The proof is the same as the one given in Theorem 5.3, except that in step 1 of the instructions for $M_2$, “has the form $0^n1^n2^n$” is replaced by “has the form $0^n1^n2^n$”. Now if $M$ accepts $w$, then $L(M)$ is the context-free language $\Sigma^*$ and if $M$ does not accept $w$, then $L(M)$ is the non-context-free language $\{0^n1^n2^n|n \geq 0\}$.

2. Let $\text{NONREGULAR}_{TM} = \{\langle M \rangle | M \text{ is a Turing machine, and } L(M) \text{ is not a regular language}\}$. Suppose that you want to reduce $A_{TM}$ to $\text{NONREGULAR}_{TM}$ by transforming $\langle M, w \rangle$ to $\langle M_2 \rangle$. (So if $\langle M, w \rangle$ is in $A_{TM}$, then $\langle M_2 \rangle$ is in $\text{NONREGULAR}_{TM}$, and if $\langle M, w \rangle$ is not in $A_{TM}$, then $\langle M_2 \rangle$ is not in $\text{NONREGULAR}_{TM}$.)

(a) Fill in the blanks in the following two statements in a way that states what you have to do to make the reduction work. Make your statements as general as possible. (In both cases you will be writing down something about the behavior of the Turing machine $M_2$.)

- If $M$ accepts $w$, then $L(M)$ is not regular.
- If $M$ does not accept $w$, then $L(M)$ is regular.

(b) Give the definition of the desired Turing machine $M_2$, given $M$ and $w$.

$M_2 = \text{"On input } x$

1. If $x$ does not have the form $a^n b^n$ for some $n$, reject.
2. If $x = a^n b^n$ for some $n \geq 0$, run $M$ on $w$.
3. If $M$ accepts $w$, accept. If $M$ rejects $w$, reject.

If $M$ accepts $w$, then $L(M) = \{a^n b^n | n \geq 0\}$, so $L(M)$ is not regular. If $M$ does not accept $w$, then $L(M) = \emptyset$, so $L(M)$ is regular.

3. **Problem 5.12**

Let $D = \{\langle M \rangle | M \text{ is a single tape Turing machine which writes a blank symbol over a nonblank symbol when it is run on some input}\}$. We will reduce $A_{TM}$ to $D$ to show that $D$ is undecidable. Given a Turing machine $M$ and an input $w$, we define a single-tape TM $M_1$. On input $x$, $M_1$ moves beyond the input, marks the end of the input with a $*$ and then begins simulating $M$ on $w$. $M_1$ treats $*$ as the left end of the tape in its simulation, so it never moves back over the original input $x$. When $M$ would print a blank, $M_1$ uses another symbol that it has reserved to use in place of the blank symbol, so during the simulation, $M_1$ never prints a blank. If $M$ accepts $w$, then $M_1$ prints a nonblank symbol and then prints
a (real) blank over the nonblank symbol. Thus, if \( M \) accepts \( w \), \( M_1 \) prints a blank over a nonblank symbol no matter what input \( x \) it is run on, while if \( M \) does not accept \( w \), \( M_1 \) never prints a blank over a nonblank symbol no matter what input \( x \) it is run on. This means that \( \langle M, w \rangle \in A_{TM} \) if and only if \( \langle M_1 \rangle \in D \).

Now suppose that \( R \) is a TM that decides \( D \). We define a TM \( S \) that decides \( A_{TM} \) as follows.

\[ S = \text{"On input } \langle M, w \rangle \text{ where } M \text{ is a TM and } w \text{ is an input,} \]

1. Construct the TM \( M_1 \) described above.
2. Run \( R \) on \( \langle M_1 \rangle \).
3. If \( R \) accepts, then \text{accept}. If \( R \) rejects, then \text{reject}.”

Since no such TM \( S \) can exist, \( R \) does not exist and \( D \) is undecidable.

4. Problem 5.15

The corresponding language is \( MOVELEFT_{TM} = \{ \langle M, w \rangle | M \text{ is a Turing machine and on input } w M \text{ attempts to move its head left at some point in the computation} \} \).

A Turing machine to decide \( MOVELEFT_{TM} \) implements the following algorithm. Given \( M \) and \( w \), simulate \( M \) on \( w \) until either a) \( M \) attempts to move left, b) \( M \) halts without ever trying to move left, or c) \( M \) moves right beyond the original input and then repeats some state without ever trying to move left. One of these three things has to happen eventually because if \( M \) never tries to move left and never halts, it will always move right and eventually move past the original input. Since there are only finitely many states, eventually a state will be repeated after \( M \) has gone past its original input. If a) happens, then \( \langle M, w \rangle \in MOVELEFT_{TM} \). If b) happens, then \( \langle M, w \rangle \notin MOVELEFT_{TM} \). If c) happens, then we also have \( \langle M, w \rangle \notin MOVELEFT_{TM} \) because if \( M \) goes from a configuration \( up\) to a configuration \( uvp\) without ever moving left, then \( M \)'s computation will continue as \( uvvp\), \( vvvp\), \ldots \) without ever trying to move left. Since a Turing machine can implement this algorithm, \( MOVELEFT_{TM} \) is decidable.

5. Problem 27

Let \( EQ_{2DIM-DFA} = \{ \langle A, B \rangle | A \text{ and } B \text{ are 2DIM-DFA and } L(A) = L(B) \} \). We show that \( EQ_{2DIM-DFA} \) is undecidable.

Let \( E_{2DIM-DFA} = \{ \langle A \rangle | A \text{ is a 2DIM-DFA and } L(A) = \emptyset \} \). \( E_{2DIM-DFA} \) is \( m \)-reducible to \( EQ_{2DIM-DFA} \) in the same way that \( E_{TM} \) is \( m \)-reducible to \( EQ_{TM} \), so if we show that \( E_{2DIM-DFA} \) is undecidable, this will show that \( EQ_{2DIM-DFA} \) is undecidable.

To show that \( E_{2DIM-DFA} \) is undecidable, we will \( m \)-reduce \( A_{TM} \) to \( E_{2DIM-DFA} \) by a mapping which takes \( \langle M, w \rangle \) to \( \langle B \rangle \). This means that
if $M$ accepts $w$, then we want $L(B)$ to be non-empty and if $M$ does not accept $w$, we want $L(B)$ to be empty. We accomplish this by making $L(B)$ be the set of accepting computation histories of $M$ on $w$. A computation history $C_1, \ldots, C_k$ is presented to $B$ as a rectangle with $C_1$ in the first row, $C_2$ in the second row, etc. (Short configurations are padded with blanks on the right end.) Given an input rectangle, $B$ checks that the first row is the initial configuration of $M$ on $w$, that the last row is an accepting configuration, and that each row follows from the previous row by the rules of $M$. ($B$ can’t write, but because the configurations are one on top of the other, $B$ can still check that one configuration $C_{i+1}$ follows correctly from $C_i$ — the only entries in $C_{i+1}$ that could be different from the entries in $C_i$ are the ones within one cell of where the state is in $C_i$.) If the rectangle passes all these tests, then $B$ accepts, else $B$ rejects.

If $M$ accepts $w$, then there is an accepting configuration history of $M$ on $w$ and $L(B) \neq \emptyset$. If $M$ does not accept $w$, then there is no accepting computation history of $M$ on $w$ and $L(B) = \emptyset$. Thus, we have $m$-reduced $A_{TM}$ to $E_{2DIM-DFA}$ and $E_{2DIM-DFA}$ is undecidable. It follows that $EQ_{2DIM-DFA}$ is undecidable.

6. **Exercise 5.4**

No. For example, let $A = \{0^n1^n|n \geq 0\}$ and $B = \{0^n1^m|n, m \geq 0\}$. We will show that $A \leq_m B$ even though $A$ is not regular and $B$ is regular. A mapping reduction $f$ from $A$ to $B$ is computed by the Turing machine $F$ given by

$$F = \text{"On input } w \in \{0, 1\}^*,$$

1. Determine if $w$ belongs to $A$. (Since $A$ is decidable, $F$ can do this with no further information.)
2. If $w$ belongs to $A$, then output 01. If $w$ does not belong to $A$, then output 10."

(In fact, the same argument shows that if $A$ is any decidable language and $B$ is any language other than $\emptyset$ and $\Sigma^*$, then $A \leq_m B$.)

7. **Problem 5.22**

$A$ is Turing-recognizable if and only if $A \leq_m A_{TM}$.

**Proof:** First suppose that $A$ is Turing-recognizable and let $M$ be a Turing machine that recognizes $A$. The function $f$ defined by $f(w) = \langle M, w \rangle$ is a reduction from $A$ to $A_{TM}$ because it is obviously computable and we have

$$w \in A \iff M \text{ accepts } w \iff \langle M, w \rangle \in A_{TM} \iff f(w) \in A_{TM}.$$ 

Now suppose that $A \leq_m A_{TM}$. We know that $A_{TM}$ is Turing-recognizable, so by Theorem 5.28, $A$ is Turing-recognizable.
8. **Problem 5.23**

A is decidable if and only if $A \leq_m 0^*1^*$.  
**Proof:** First suppose that $A$ is decidable. Define $f$ by $f(x) = 01$ if $x \in A$ and $f(x) = 10$ if $x \notin A$. Since $A$ is decidable, $f$ is computable and $x \in A$ if and only if $f(x) \in 0^*1^*$, so $A \leq_m 0^*1^*$.

Conversely, suppose that $A \leq_m 0^*1^*$. Since $0^*1^*$ is decidable, $A$ is decidable by Theorem 5.22.

9. **Problem 5.24**

The set $\overline{A_{TM}}$ is mapping reduced to $J$ by the function $f(y) = 1y$. Thus, $J$ is not Turing-recognizable. The set $A_{TM}$ is mapping reduced to $J$ by the function $g(x) = 0x$. This shows that $\overline{A_{TM}}$ is mapping reducible to $J$ and hence that $J$ is not Turing-recognizable.

10. **Problem 5.25**

Consider the set $J$ of Problem 5.24. According to that problem, $J$ is not Turing-recognizable, so $J$ is not decidable. We will show that $J \leq_m \overline{J}$, so $B = J$ is a solution to the problem.

First note that $\overline{J} = \{ w | w = 0x \text{ for some } x \in \overline{A_{TM}} \text{ or } w = 1y \text{ for some } y \in A_{TM} \text{ or } w = \varepsilon \text{ or } w \text{ begins with a symbol other than 0 or 1} \}$. Let $z_0$ be some fixed string in $J$, for example, $z_0$ could be $0x_0$ for some particular $x_0$ in $A_{TM}$. Define $f : \Sigma^* \rightarrow \Sigma^*$ by

$$f(w) = \begin{cases} 1x & \text{if } w = 0x \\ 0y & \text{if } w = 1y \\ z_0 & \text{if } w = \varepsilon \text{ or } w \text{ starts with a symbol other than 0 or 1.} \end{cases}$$

Then, it is clear that $f$ is computable. To see that $f$ mapping reduces $J$ to $\overline{J}$, suppose first that $w \in J$. We must show that $f(w) \in \overline{J}$. If $w \in J$, there are two possibilities. If $w = 0x$ with $x \in A_{TM}$, then $f(w) = 1x$ with $x \in A_{TM}$, so $f(w) \in \overline{J}$. If $w = 1y$ with $y \in A_{TM}$, then $f(w) = 0y$ with $y \in A_{TM}$, so $f(w) \in \overline{J}$. Thus, if $w \in J$, then $f(w) \in \overline{J}$.

Now suppose that $w \notin J$. We must show that $f(w) \notin \overline{J}$. There are four possibilities to consider. If $w = 0x$ with $x \in A_{TM}$, then $f(w) = 1x$, so $f(w) \notin \overline{J}$. If $w = 1y$ with $y \in A_{TM}$, then $f(w) = 0y$, so $f(w) \notin \overline{J}$. If $w = \varepsilon$ or $w$ starts with a symbol other than 0 or 1, then $f(w) = z_0$, so $f(w) \notin \overline{J}$. Thus, if $w \notin J$, then $f(w) \notin \overline{J}$.

This shows that $f$ is a mapping reduction of $J$ to $\overline{J}$.

11. It is not possible to m-reduce $E_{LBA}$ to $A_{LBA}$.

**Proof:** Suppose that $E_{LBA} \leq_m A_{LBA}$. By Theorem 5.9, $A_{LBA}$ is decidable, so by Theorem 5.22, $E_{LBA}$ is decidable. This contradicts Theorem 5.10, so the $m$-reduction is not possible.

12. It is not possible to m-reduce $\overline{EQ_{TM}}$ to $A_{TM}$.

**Proof:** If $\overline{EQ_{TM}} \leq_m A_{TM}$, then by Theorem 5.28, and the fact that $A_{TM}$
is Turing-recognizable, $\overline{\text{EQ}_{TM}}$ is Turing recognizable. This contradicts Theorem 5.30.