1. Exercise 4.1.1

(a) We trace the computation of the given Turing machine $M$ from configuration $(q_0, \triangleright abbba)$:

$(q_0, \triangleright abbba) \vdash (q_1, \triangleright bbbba) \vdash (q_0, \triangleright bbbba)$

$(q_1, \triangleright bbabba) \vdash (q_0, \triangleright babbb)$

$(q_1, \triangleright baabba) \vdash (q_0, \triangleright baabba)$

$(q_1, \triangleright baabba) \vdash (q_0, \triangleright baabba)$

$(q_1, \triangleright baabba) \vdash (q_0, \triangleright baabba)$

$(q_1, \triangleright baabba) \vdash (q_0, \triangleright baabba)$

$(q_1, \triangleright baabba) \vdash (q_0, \triangleright baabba)$

$(h, \triangleright baabba)$

(b) The machine will scan from the current square to the right until it finds a blank, along the way converting each $a$ to $b$ and each $b$ to $a$.

2. Exercise 4.1.2

(a) We trace the computation of the given Turing machine $M$ from configuration $(q_0, \triangleright abb \sqcup bb \sqcup \sqcup aba)$:

$(q_0, \triangleright abb \sqcup bb \sqcup \sqcup \sqcup aba) \vdash (q_1, \triangleright abb \sqcup bb \sqcup \sqcup \sqcup aba) \vdash (q_0, \triangleright abb \sqcup bb \sqcup \sqcup \sqcup aba)$

$(q_1, \triangleright abb \sqcup bb \sqcup \sqcup \sqcup aba) \vdash (q_1, \triangleright abb \sqcup bb \sqcup \sqcup \sqcup aba)$

$(q_2, \triangleright abb \sqcup bb \sqcup \sqcup \sqcup aba) \vdash (h, \triangleright abb \sqcup bb \sqcup \sqcup \sqcup aba)$

(b) The machine will search for a string of blanks separating a $b$ (on the left) and an $a$ (on the right), where the $a$ must occur at or to the right of the current square. The machine does not halt, unless such a string can be found. If the search succeeds, the head will in the end be positioned at the leftmost of these blanks.
3. **Exercise 4.1.4**

When the given Turing machine is started in a configuration \((q_0, \triangleright a^n a)\), it will scan to the left, erasing every other \(a\). If \(n\) is even, the machine will halt in configuration \((h, \sqcap a u)\), where \(u\) is the string \((\sqcup a)^{n/2}\); if \(n\) is odd, the machine does not halt.

4. **Exercise 4.1.7**

A suitable Turing machine is \((K, \Sigma, \delta, s, \{h\})\), where \(K = \{s, q, h\}\), \(\Sigma = \{a, b, \sqcap, \triangleright\}\), and the transition function \(\delta\) is given by the following table:

<table>
<thead>
<tr>
<th>(q)</th>
<th>(\sigma)</th>
<th>(\delta(q, \sigma))</th>
</tr>
</thead>
<tbody>
<tr>
<td>(s)</td>
<td>(a)</td>
<td>((q, \rightarrow))</td>
</tr>
<tr>
<td>(s)</td>
<td>(b)</td>
<td>((s, \rightarrow))</td>
</tr>
<tr>
<td>(s)</td>
<td>(\sqcap)</td>
<td>((s, \rightarrow))</td>
</tr>
<tr>
<td>(s)</td>
<td>(\triangleright)</td>
<td>((s, \rightarrow))</td>
</tr>
<tr>
<td>(q)</td>
<td>(a)</td>
<td>((h, a))</td>
</tr>
<tr>
<td>(q)</td>
<td>(b)</td>
<td>((s, \rightarrow))</td>
</tr>
<tr>
<td>(q)</td>
<td>(\sqcap)</td>
<td>((s, \rightarrow))</td>
</tr>
<tr>
<td>(q)</td>
<td>(\triangleright)</td>
<td>((s, \rightarrow))</td>
</tr>
</tbody>
</table>

5. **Exercise 4.1.8**

(a) The expression \(LL\) denotes a Turing machine \((K, \Sigma, \delta, s_1, \{h_2\})\), where \(K = \{s_1, h_1, s_2, h_2\}\) and the transition function \(\delta\) is defined by:

<table>
<thead>
<tr>
<th>(q)</th>
<th>(\sigma)</th>
<th>(\delta(q, \sigma))</th>
</tr>
</thead>
<tbody>
<tr>
<td>(s_1)</td>
<td>(x)</td>
<td>((h_1, \leftarrow))</td>
</tr>
<tr>
<td>(s_1)</td>
<td>(\triangleright)</td>
<td>((s_1, \rightarrow))</td>
</tr>
<tr>
<td>(h_1)</td>
<td>(x)</td>
<td>((s_1, x))</td>
</tr>
<tr>
<td>(h_1)</td>
<td>(\triangleright)</td>
<td>((s_1, \rightarrow))</td>
</tr>
<tr>
<td>(s_2)</td>
<td>(x)</td>
<td>((h_2, \leftarrow))</td>
</tr>
<tr>
<td>(s_2)</td>
<td>(\triangleright)</td>
<td>((s_2, \rightarrow))</td>
</tr>
</tbody>
</table>

where \(x\) ranges over the set \(\Sigma - \{\triangleright\}\).

6. **Exercise 4.1.9**

The machines \(LR\) and \(RL\) do not always accomplish the same thing. For example, if started in a configuration \(\triangleright a_1 a_2 w\), the machine \(LR\) will yield \(\triangleright a_1 a_2 w\), whereas \(RL\) will yield \(\triangleright a_1 a_2 w\).
7. Exercise 4.2.1

The following Turing machine computes the function \( f : \{a, b\}^* \rightarrow \{a, b\}^* \) with \( f(w) = ww^R \).

The machine transforms a configuration \( \sqcap \sqcup w \) to \( \sqcup \sqcap w w^R \). The variable \( x \) in the diagram is meant to range over \( \{a, b\} \).

8. Exercise 4.5.1

(a) Let \( M = (K, \Sigma, \Delta, q_0, \{h\}) \) be a nondeterministic Turing machine, where \( K = \{q_0, q_1, q_2, q_3, q_4, q_5, h\} \), \( \Sigma = \{a, b, \sqcup, \sqcap\} \), and

\[
\Delta = \{(q_0, \sqcap), (q_1, \rightarrow)\}, \\
\{(q_1, a), (q_1, \rightarrow)\}, \\
\{(q_1, a), (q_2, \rightarrow)\}, \\
\{(q_2, b), (q_3, \rightarrow)\}, \\
\{(q_3, b), (q_3, \rightarrow)\}, \\
\{(q_3, b), (q_4, \rightarrow)\}, \\
\{(q_4, a), (q_5, \rightarrow)\}, \\
\{(q_5, a), (q_5, \rightarrow)\}, \\
\{(q_5, \sqcup), (h, \sqcap)\}\}.
\]

This Turing machine semidecides the regular language \( a^*abb^*baa^* \).

(b) We outline a nondeterministic Turing machine \( M \) that accepts strings of the form \( ww^R uu^R \), where \( w \) and \( u \) are strings in \( \{a, b\}^* \). A suitable machine can be obtained as a combination of simpler machines. The desired machine should accept the empty string. If the input is a nonempty string \( ww^R uu^R \), we may assume without loss of generality that \( u \) is a nonempty string.

(i) First, let \( M_1 \) be a machine that nondeterministically selects a position in its input. The machine \( M_1 \) has two states, an initial
state $s$ and a halting state $h$, and transitions $((s, \sigma), (s, \rightarrow))$ and $((s, \sigma), (h, \rightarrow))$, for all tape symbols $\sigma$. If $u$ is a string $avv$, this machine can transform a starting configuration $\triangleright_L wwR uuR$ to $\triangleright_L wR avvR a$.

(ii) The next part of $M$ works in a similar way as a right-shifting machine by shifting the string beginning at the current position one square to the right and inserting a blank symbol. Thus $\triangleright_L wR avvR a$ is transformed to $\triangleright_L wR \equiv avvR a$.

(iii) Then the machine determines whether the tape contains a string from $\{uuR : u \in \{a, b\}^*\}$ beginning on the square to the right of its head. If that test is successful, the machine moves the head to the leftmost blank square and repeats the same test, halting only if it is successful.

9. Exercise 4.6.2

(c) The grammar $G = (V, \{a\}, R, S)$ is designed to generate the language $\{a^n^2 : n \geq 0\}$, where $V = \{S, L, R, T, [, ], a\}$ and

$$R = \{ \begin{array}{ll}
S \rightarrow e, & S \rightarrow [R], \\
RA \rightarrow AR, & R] \rightarrow LA], \\
TA \rightarrow aT, & T] \rightarrow e, \\
[L \rightarrow T, & L] \rightarrow [R, \\
AL \rightarrow LAaa \\
aA \rightarrow Aa, & a] \rightarrow [a \} \\
\end{array} \}
$$

The rules can be used to generate strings of the form $[A^kR]a^{k^2-k}$, for any $k \geq 0$. From $[A^kR]a^{k^2-k}$ one can derive, first, $[L(Aaa)^kA]a^{k^2-k}$ and, after rearranging the order of symbols, $[L(a(k+1)^2-(k+1)].$ (Note that $(k+1)^2 - (k+1) = k^2 + 2k + 1 - (k + 1) = (k^2 - k) + 2k.$) The variable $L$ can then be converted either to $R$, to induce another iteration of this process, or to $T$ to terminate the process and change all occurrences of $A$ to $a$.

10. Exercise 5.2.1

(a) Let $M = (K, \Sigma, \delta, s, \{h\})$ be the Turing machine in Example 4.1.1.
States and tape symbols are encoded as follows:

<table>
<thead>
<tr>
<th>state/symbol</th>
<th>encoding</th>
</tr>
</thead>
<tbody>
<tr>
<td>$q_0$</td>
<td>$q00$</td>
</tr>
<tr>
<td>$q_1$</td>
<td>$q01$</td>
</tr>
<tr>
<td>$h$</td>
<td>$q11$</td>
</tr>
<tr>
<td>$\square$</td>
<td>$a000$</td>
</tr>
<tr>
<td>$\triangleright$</td>
<td>$a001$</td>
</tr>
<tr>
<td>$\leftarrow$</td>
<td>$a010$</td>
</tr>
<tr>
<td>$\rightarrow$</td>
<td>$a011$</td>
</tr>
<tr>
<td>$a$</td>
<td>$a100$</td>
</tr>
</tbody>
</table>

The Turing machine $M$ is represented by the string

\[
\text{“}M''\text{”} = (q00, a100, q01, a000), \\
(q00, a000, q11, a000), \\
(q00, a001, q00, a011), \\
(q01, a100, q00, a100), \\
(q01, a000, q00, a011), \\
(q01, a001, q01, a011).
\]

(b) The string $aaa$ is encoded as $a100a100a100$.

(c) The machine $U'$ has to initialize its three tapes so as to enable simulation of $M$. At the beginning of the simulation, the first tape contains the string $\triangleright \sqcup a001a000a100a100a100$, the second tape contains $\triangleright \sqcup \text{“}M''\text{”}$, and the third tape contains $\triangleright \sqcup q00$.

If the machine $M$ is started in state $q_0$ and initial tape $\triangleright \sqcup aaa$, it will immediately go to a halting state. If it is started in state $q_1$ with the same initial tape, then at the beginning of the simulation of the third step of $M$, the first tape will contain the string $\triangleright \sqcup a001a000a000a100a100a100$ (with the head positioned as indicated) and the third tape the string $\triangleright \sqcup q01$. (The contents of the second tape do not change.)

11. **Exercise 5.4.2**

(a) Let $H_Q$ be the language

\[
\{ \text{“}M''\text{”} q'' w'' : M \text{ reaches state } q \text{ when started on input } w \text{ from its initial state} \}.
\]
We show how to reduce the halting problem to $H_Q$.

Let $M = (K, \Sigma, \delta, s, H)$ be a Turing machine. By $M_h$ we denote the Turing machine $(K \cup \{h\}, \Sigma, \delta \cup \delta_h, s, \{h\})$, where $h \notin K$ and $\delta_h(q, \sigma) = (h, \rightarrow)$, for all states $q \in H$. In other words, $M_h$ performs essentially the same computations as $M$ but has single halting state $h$.

We define a function $\tau$ by: $\tau(x) = "M_h""w", \text{ if } x \text{ is a string of the form } "M""w", \text{ and } \tau(x) = e, \text{ otherwise.}$ This function is a reduction from the set $H$ to $H_Q$, as a machine $M$ halts on input $w$ if and only if $M_h$ reaches state $h$ for input $w$. We may conclude that the set $H_Q$ is not recursive.

(e) The problem of determining, given a Turing machine $M$, whether $M$ ever writes a nonblank symbol when started on the empty tape, is decidable.

Let $M$ be a given Turing machine and $k$ be the number of nonhalting states of $M$. When $M$ is started from its initial state on an empty tape, precisely one of the following situations will occur after at most $k$ steps:

(i) $M$ has written a nonblank symbol.
(ii) $M$ has reached a halting state without writing a nonblank symbol.
(iii) $M$ has never written a nonblank symbol and has reached some nonhalting state $q$ repeatedly.

In cases (i) and (ii) the answer to the stated problem is obvious. In case (iii) the machine will repeat a cycle ad infinitum without ever writing a nonblank symbol.

(g) Let $H_C$ be the language

\[ \{ "M''"M'': M_1 \text{ semidecides the complement of } L(M_2) \}. \]

We show that $H_C$ is undecidable by giving a reduction to $H_C$ from the set

\[ H_4 = \{ "M'": M \text{ halts on all inputs} \}. \]

Let $N$ be a Turing machine that halts for no input. Thus, $L(N)$ is the empty set, whereas its complement is $\Sigma^*$. We define a function $\tau$ by: $\tau(x) = "M''", \text{ if } x \text{ is a string of the form } "M''"N'', \text{ and } \tau(x) = e, \text{ otherwise.}$ This function is a reduction from $H_4$ to $H_C$: 
a string $x$ is an element of $H_4$ if and only if $x = "M"$ for some Turing machine $M$ that halts on all inputs; and

$\tau("M") = "M""N"$ is an element of $H_C$ if and only if $M$ accepts all strings in the complement of $L(N)$, i.e., halts on all inputs.

(h) Let $H_I$ be the language

$\{"M_1""M_2" : \text{there is a string for which both } M_1 \text{ and } M_2 \text{ halt}\}$.

Define a function $\tau$ by: $\tau(x) = "M""M", \text{ if } x \text{ is a string of the form } "M", \text{ and } \tau(x) = e, \text{ otherwise.}$ This function is a reduction from the set

$H_3 = \{"M" : M \text{ halts on some input}\}$

to $H_I$. Since $H_3$ is not recursive, $e$ may conclude that $H_I$ is not recursive either.