1. (a) $q_0\text{aaa, } q_1\text{aa, } q_2\text{a}$, $q_3\text{a}$, $q_R$

(b) $q_0\text{aaaa, } q_1\text{aaa, } q_2\text{aa, } q_3\text{a}$, $q_4\text{a}$, $q_5\text{a}$, $q_6\text{a}$, $q_7\text{a}$, $q_8\text{a}$, $q_9\text{a}$, $q_{10}\text{a}$, $q_{11}\text{a}$, $q_{12}\text{a}$

2. (a) Construct a TM $M$ that does the following on input $w$:

1. Scan $w$ for a $b$; if a $b$ is found, mark it, move left to the beginning of the tape, and go to the next step. If no $b$ is found and no $a$ is found, then accept; otherwise, reject.

2. Scan $w$ to find an $a$; mark it if one is found go to the next step; otherwise, reject.

3. Continue scanning $w$; if another $a$ is found, mark it, move left to the beginning of the tape, and go to the first step; otherwise, reject.

(b) Construct a TM $M$ that does the following on input $w$:

1. Scan $w$ for a $b$; if a $b$ is found, mark it, move left to the beginning of the tape, and go to the next step. If no $b$ is found and no $a$ is found, then reject; otherwise, accept.

2. Scan $w$ to find an $a$; mark it if one is found go to the next step; otherwise, accept.

3. Continue scanning $w$; if another $a$ is found, mark it, move left to the beginning of the tape, and go to the first step; otherwise, accept.

3. To show that a PDA with two stacks can simulate a Turing machine, we consider a configuration of a TM $uqv$, where $q$ is the current state, $u$ is the contents of the tape to the left of the tape head, and $v$ is the contents of the tape at the tape head and to the right of the tape head. Then for $u = u_1 \cdots u_k$, we say that the first stack contains $u$ where $u_1$ is the bottom of the stack and for $v = v_1 \cdots v_\ell$, we say that the second stack contains $v_2 \cdots v_\ell$ with $v_1$ on the bottom of the stack.

Since $v_1$ is the symbol in the tape cell under the tape head, which stack it belongs to depends on which direction we want the tape head to move. If we want the tape head to move to the right, then $v_1$ is on the top of the second stack and we pop $v_1$, perform the transition, and push the symbol to be written to the first stack. To simulate a
move to the left, we pop $v_1$ from the first stack, perform the transition, and push the symbol to be written onto the second stack.

Lastly, we need to consider when we reach the bottom of a stack. If there are no elements to pop from the first stack, then we have reached the left end of the tape and we do nothing. If there are no elements to pop from the second stack, then we have reached a blank space and we push a blank symbol onto the left stack.

4. (a) Given Turing machines $M_1$ and $M_2$ that decide languages $L_1$ and $L_2$ respectively, we can construct a Turing machine $M_3$ that decides $L_1 \cap L_2$. $M_3$ operates as follows:

1. On input word $w$, run $M_1$ on $w$. If $M_1$ accepts, then go to the next step. If $M_1$ rejects, then reject.
2. Run $M_2$ on $w$. If $M_2$ accepts, then accept; otherwise, reject.

We can see that $M_3$ will only accept $w$ if both $M_1$ and $M_2$ accept $w$. Also, $M_3$ is guaranteed to halt, since $M_1$ and $M_2$ are both guaranteed to halt. Thus, $L_1 \cap L_2$ is decidable.

(b) Given a Turing machine $M$ that decides a language $L$, we can construct a Turing machine $M'$ which decides $\overline{L}$. $M'$ operates as follows:

1. On input $w$, run $M$ on $w$.
2. If $M$ rejects $w$, then accept; if $M$ accepts $w$, then reject.

Since $L$ is decidable, $M$ is guaranteed to halt. Thus $M'$ decides $\overline{L}$ and $\overline{L}$ is decidable.

(c) Let $M_1$ and $M_2$ be Turing machines that recognize languages $L_1$ and $L_2$, respectively. We can construct a Turing machine $M_3$ that recognizes $L_1 \cap L_2$. $M_3$ operates as follows:

1. On input word $w$, run $M_1$ on $w$. If $M_1$ accepts, then go to the next step. If $M_1$ rejects, then reject.
2. Run $M_2$ on $w$. If $M_2$ accepts, then accept; otherwise, reject.

First, we note that $M_3$ accepts $w$ only if both $M_1$ and $M_2$ accept $w$ and $M_3$ will reject $w$ if at least one of $M_1$ or $M_2$ rejects $w$. However, if either $M_1$ or $M_2$ do not halt, then $M_3$ does not halt. Thus, $M_3$ recognizes $L_1 \cap L_2$.

(d) Let $M_1$ and $M_2$ be Turing machines that recognize languages $L_1$ and $L_2$, respectively. We can construct a Turing machine $M_3$ that recognizes $L_1 \cdot L_2$. $M_3$ operates as follows:

1. On input $w$, nondeterministically split $w$ into two parts $w = w_1 w_2$. 

2
2. Run $M_1$ on $w_1$. If $M_1$ accepts, then go to the next step. If $M_1$ rejects, then reject.

3. Run $M_2$ on $w_2$. If $M_2$ accepts, then accept. If $M_2$ rejects, then reject.

We note that $M_3$ will only accept $w$ if there exists a branch of computation where $w_1$ is accepted by $M_1$ and $w_2$ is accepted by $M_2$. If there is no $w_1$ that is accepted by $M_1$, $M_3$ either halts and rejects or runs forever. The same applies to $M_2$ if there exists some $w_1$ that is accepted by $M_1$ but no suitable $w_2$ is accepted by $M_2$.

5. We construct a Turing machine $M$ that decides $\text{ALL}_{\text{DFA}}$. First, we note that the class of regular languages is closed under complementation and that there is an algorithm that when given a DFA $A$ constructs a DFA that recognizes the language $\overline{L(A)}$. Secondly, we note that if $L(A) = \Sigma^*$, then $\overline{L(A)} = \emptyset$. Let $M_E$ be a Turing machine that decides $E_{\text{DFA}}$. Then $M$ operates as follows:

1. On input $\langle A \rangle$, where $A$ is a DFA, construct a DFA $A'$ such that $L(A') = \overline{L(A)}$.

2. Run $E_{\text{DFA}}$ on $\langle A' \rangle$.

3. If $E_{\text{DFA}}$ accepts $\langle A' \rangle$, then accept; otherwise, reject.

6. To show that a doubly-infinite Turing machine can simulate an ordinary TM, we simply mark the initial tape cell with a special symbol that disallows the machine from moving to the left of the cell.

To show that an ordinary Turing machine can simulate a doubly-infinite Turing machine, instead, we show that a 2-tape TM, which we have shown to be equivalent in power to the ordinary TM, can simulate a doubly-infinite tape. Let $D$ be a doubly-infinite TM and let $M$ be our 2-tape TM. We split the tape of $D$ into two parts and assign each part to a tape of $M$. Tape 1 of $M$ corresponds to the part of the tape of $D$ that contains the input word and everything to the right. Tape 2 of $M$ contains everything on the tape of $D$ to the left of the input word in reverse order.

More formally, let $w_0$ denote the contents of the tape cell that contained the first symbol of the input word at the beginning of the computation of $D$. Then if $D$ has a tape $uw_0v$, Tape 1 of $M$ contains $w_0v$ and Tape 2 of $M$ contains $u^R$. 

3