1. a) No. M does not accept bbaa.
   b) Yes. M accepts baaab.
   c) No. Input is incorrect format. (Input should encode a DFA and a string.)
   d) No. Input is incorrect format. (An input for Arex should encode a regular expression and a string.)
   e) No. \( L(M) \neq \emptyset \)
   f) No. Input encodes only one DFA and is incorrect format.
   g) Yes. \( L(M) = L(M) \)
2. A decider for $\text{ALL}_{\text{NFA}}$:

$Q = "\text{On input } \langle M \rangle \text{ where } M \text{ is a DFA}
1. Convert } M \text{ to an equivalent DFA } M_D.
2. Let } M_0 \text{ be a one-state DFA recognizing } \Sigma^*.
3. Run the decider } F \text{ for } \text{EQ}_{\text{DFA}} \text{ (Theorem 4.5)}
on input } \langle M_D, M_0 \rangle.
\text{If } F \text{ accepts, accept; and else reject.}"
3. (a) i) \( f \) is not onto. There is no value \( x \)
 such that \( f(x) = -1 \).

ii) \( f \) is not one-to-one because \( f(3) = f(-1) \) = 4.

iii) No because it is neither onto nor one-to-one.

b) The following list contains all finite languages:

Stage 1. List all languages of cardinality at most one
where the length of the longest string is at most one.

Stage k. List all languages of cardinality at most \( k \) where
the length of the longest string is at most \( k \).

Omit languages that have been listed in stage 1, ..., \( k-1 \).

The construction contains all finite languages because
any finite language \( L \) gets listed in stage \( m \) where
\( m \) is the maximum of the cardinality of \( L \) and the
length of the longest string in \( L \).
4. Turing machine deciding $A$.

$Q$ = "On input $\langle R \rangle$ where $R$ is a regular expression:

1. Construct a DFA $M_1$ that is equivalent to $R$.

2. Construct a DFA $M_2$ that recognizes the complement of $\Sigma^*00000\Sigma^* + \Sigma^*111\Sigma^*$. This is possible due to closure properties of regular languages.

3. Construct a DFA $M_3$ such that $L(M_3) = L(M_1) \cap L(M_2)$. This is possible because regular languages are closed under intersection.

4. Decide whether $L(M_3) = \emptyset$ using Theorem 4.4.

If yes, accept. If no, reject."
5. We reduce $A_{TM}$ to $B$.

Suppose TM $P$ decides $B$. Construct following decider for $A_{TM}$:

Q = "On input $\langle M, w \rangle$ where $M$ is a TM and $w$ a string:

1. Construct TM $M_w$:

   $M_w = " \text{On input string } v$

   1. Check whether $v = w$. If not, reject. Otherwise continue.

   2. Simulate $M$ on $v$ and answer what $M$ answers."

2. Run decider $P$ on input $\langle M_w, w \rangle$. If $P$ accepts, accept. If $P$ rejects, reject."

Why this works: $L(M_w) = \{w^k \mid k \geq 1 \text{ and } M \text{ accepts } w \}$ if $M$ accepts $w$.

Thus $M_w$ accepts some power of $w$ if and only if $M$ accepts $w$. 
6. \( A_{us} = \{ \langle M \rangle \mid M \text{ is a TM and } M \text{ contains one or more useless states} \} \)

We reduce \( A_{TM} \) to \( A_{us} \):

Assume that TM \( N \) decides \( A_{us} \). We construct a TM \( P \) that will decide \( A_{TM} \).

Definition of \( P \):

On input \( \langle M, w \rangle \) (where \( M \) is a TM and \( w \) an input for it)

\( P \) constructs a TM \( M' \) that operates as follows:

1. If the input for \( M' \) is \( \langle M, w \rangle \) (= the fixed TM \( M \) and string \( w \)), simulate the computation of \( M \) on \( w \)
2. Else if the input is not \( \langle M, w \rangle \), go to state reject

After this \( P \) runs \( N \) on input \( \langle M' \rangle \) and accepts if \( N \) rejects, and rejects if \( N \) accepts.
6. (continued)

Explanation why the construction works:

(i) If $M$ accepts $w$, then the computation of $M'$ on $<M, w>$ goes through all the states except $q_{accept}$. The computation of $M'$ on any other input uses $q_{accept}$. Hence $M'$ does not have useless states.

(ii) If $M$ does not accept $w$, then $M'$ does not accept any input and $q_{accept}$ is a useless state of $M'$.

Hence $M$ accepts $w$ iff (if and only if) $M'$ does not have useless states.