1. a) xxax

b) abxx

2. a) Base case: 0\cdot25 = 0
Recursion clause: (x + 1)\cdot25 = x\cdot25 + 25
This one line (↑) is a complete answer to the question.

b) The induction goes most smoothly if you do it on the number of concatenations of elements of \{aa\} that form the given member of \{aa\}^*.

**Base case:** The number of concatenations is 0, then x is the empty string. On the empty string, \(F_1\) remains in \(q_0\), which is not a final state of \(F_1\).

**Induction step:**

**Induction assumption:** Say that every string x in \{aa\}^* that is formed by n or fewer concatenations of elements of aa is not accepted by \(F_1\). Say that \(x'\) is formed by the concatenation of n+1 instances of aa.

Since 0 < n + 1, there must be strings \(y \in \{aa\}^*\) and \(y' = aa\), such that \(x' = y \ast y'\). Since y is formed with n concatenations of aa, it is not accepted by \(F_1\), by the induction assumption. If y is the empty string, then \(x' = aa\). \(F_1\) enters \(q_1\) upon reading a, and then enters \(q_2\) upon reading the final a. \(q_2\) is not a final state of \(F_1\), so the computation is unsuccessful.

If y is not the empty string, then \(F_1\) leaves \(q_0\) when it reads the first a and there is no transition that can bring it back to \(q_0\). \(q_1\) is a final state, and since y is not accepted by \(F_1\), \(F_1\) must finish reading y in \(q_2\). Now \(F_1\) has only \(y' = aa\) to read. Since \(F_1\) begins reading aa in \(q_2\), it will begin by reading a and moving into \(q_1\), and then reading a again and moving into \(q_2\), terminating in \(q_2\), which is not a final state.

So \(x'\) is not accepted by \(F_1\). This completes the inductive step, and hence the proof.

3. \{⟨q_0, a, q_1⟩, ⟨q_1, a, q_2⟩, ⟨q_2, a, q_1⟩\} are the transitions, \(q_0\), \(q_1\) and \(q_2\) are the states, with \(q_0\) and \(q_1\) the final states.

To keep the machine from accepting the empty string, don’t make \(q_0\) a final state.

[For this question, it would be acceptable to draw the labelled graph rather than to spell out the tuples.]
c) This machine recognizes strings that are either “b” (just the single letter) or consist of a string $abab \cdots ab$ of some number of “ab”’s concatenated together.