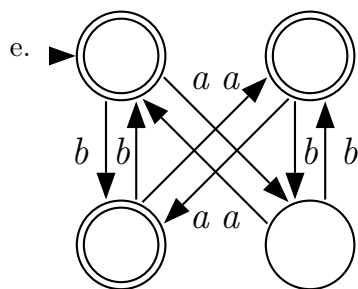
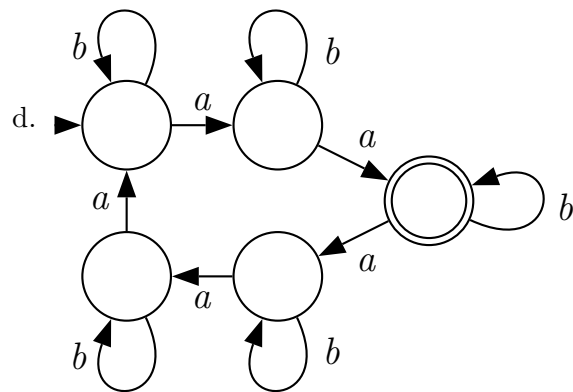
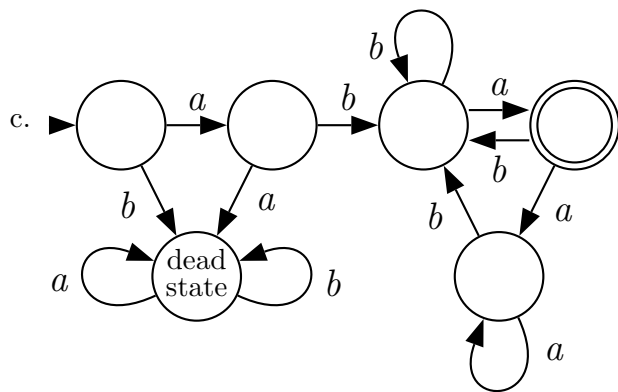
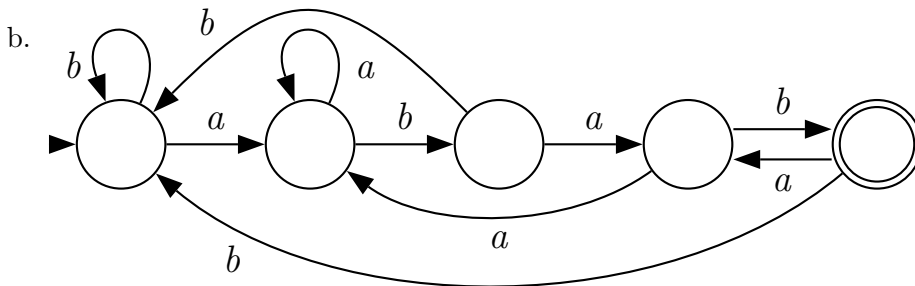
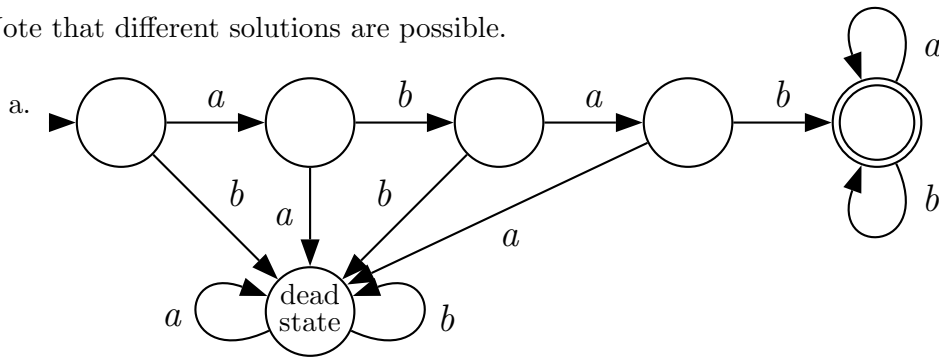


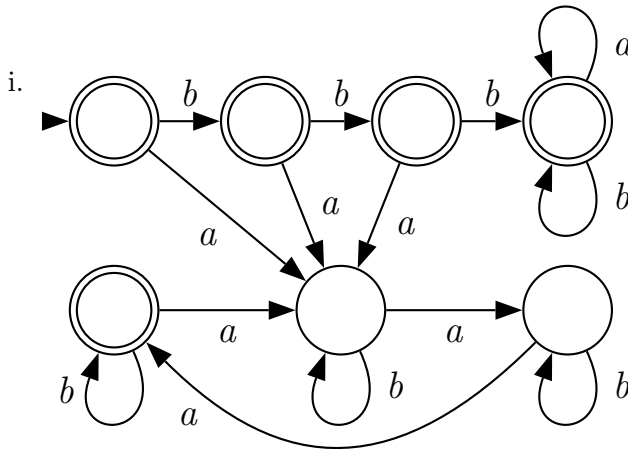
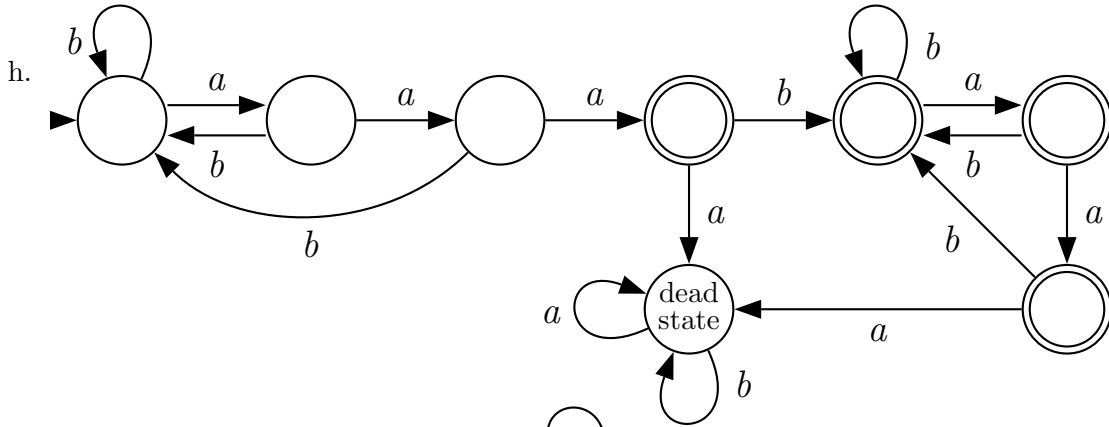
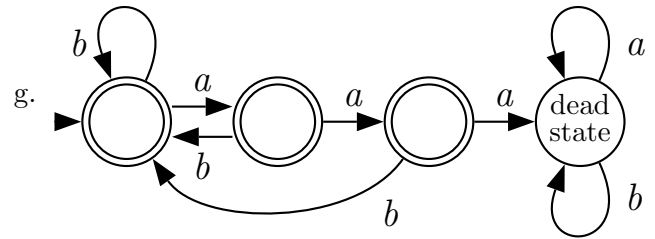
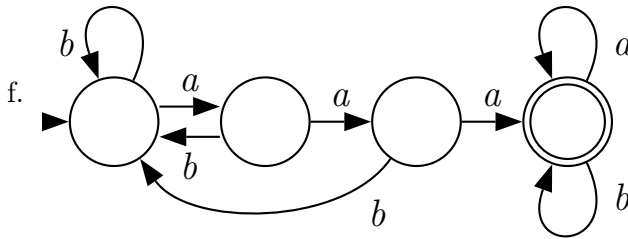
1. Note that different solutions are possible.



Note that when a language is expressed as a conjunction or disjunction of properties, as this one is, it is often convenient to construct a DFA by identifying all of the possibilities for each property and then creating a separate state corresponding to each combination of the properties. Then for each state you need to determine for each symbol of the input alphabet which other state a transition should lead to. Lastly you have to identify which of the states represent final states and which is the start state.

In this DFA, the states represent the following combinations of properties:

- The upper-left state represents strings with an even number of a 's and which have even lengths.
- The lower-left state represents strings with an even number of a 's and which have odd lengths.
- The upper-right state represents strings with an odd number of a 's and which have even lengths.
- The lower-right state represents strings with an odd number of a 's and which have odd lengths.



Below is a regular expression based directly on the DFA.

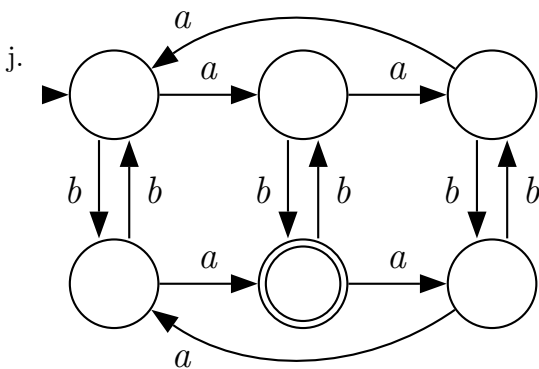
Note that every subsidiary clause in the regular expression corresponds to some path in the DFA from the start state to a final state, with Kleene closure corresponding to a cycle in the DFA.

This regular expression isn't as short as it could be. I wrote this version this way as one example illustration of the correspondence between DFAs and regular expressions.

$(\epsilon \mid b \mid bb \mid bbb(a|b)^* \mid ab^*ab^*ab^*(ab^*ab^*ab^*)^* \mid bab^*ab^*ab^*(ab^*ab^*ab^*)^* \mid bbab^*ab^*ab^*(ab^*ab^*ab^*)^*)$

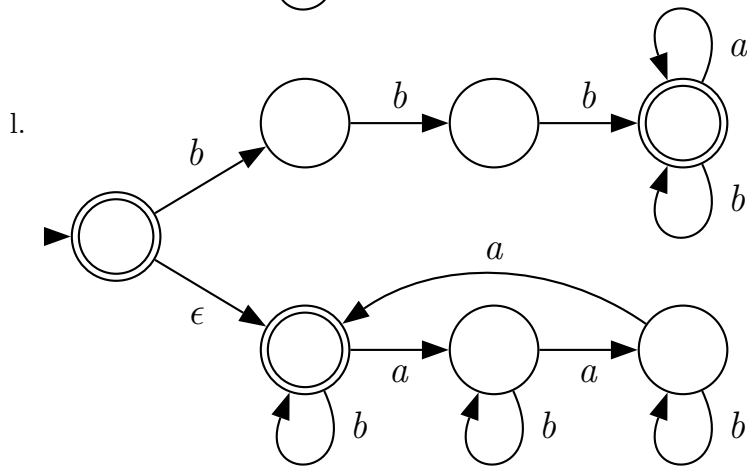
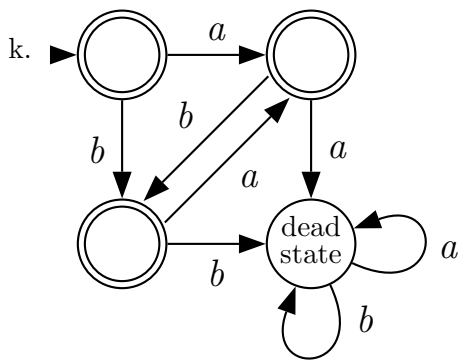
Here's a shorter version which also recognizes or accepts the same language as the DFA does:

$(b \mid bb \mid bbb(a|b)^* \mid (b^*ab^*ab^*ab^*)^*)$



This DFA can again be constructed by creating a state corresponding to each possible combination of the two properties given in the definition of the language. The top row of states represents strings which have an even number of b 's and the bottom row represents strings which have an odd number of b 's. The left column of states represents strings w for which $\#a(w) \equiv 0 \pmod{3}$, while the middle column of states represents strings w for which $\#a(w) \equiv 1 \pmod{3}$, and the rightmost column of states represents strings w for which $\#a(w) \equiv 2 \pmod{3}$.

After determining each such state, the transitions between them should be relatively easy to identify.



Note that not only does this NFA have an ϵ -transition, but some of its states don't have transitions for every alphabet symbol. There are some paths for some strings in the language (such as $baaa$) which don't lead to a final state, but there is a path which does lead to a final state for each string in the language.

2. Similar to Homework #1, you may have noticed in a few cases in this problem that even making a small change in a DFA can have a big effect on the language it accepts.
 - a. This DFA is wrong. Strings like 01011 for example are in the language, so the DFA should recognize or accept them, but it does not. Furthermore, some strings like 10 and 01 for example are not in the language, so the DFA should not recognize or accept them, but it does.
 - b. This DFA is wrong. Strings like 010, 100, and 0011 for example are in the language, so the DFA should recognize or accept them, but it does not.
 - c. This DFA is wrong. Strings like 1010 for example are in the language, so the DFA should recognize or accept them, but it does not. Furthermore, a string like 01 for example is not in the language, so the DFA should not recognize or accept it, but it does.
 - d. This DFA is wrong. Strings like ϵ , 100, and 010, and 0101 for example are in the language, so the DFA should recognize or accept them, but it does not.
 - e. This DFA is wrong. A string like 10011 for example is in the language, so the DFA should recognize or accept it, but it does not.
 - f. This DFA is wrong. A string like 0101010 for example is in the language, so the DFA should recognize or accept it, but it does not.
 - g. This DFA is wrong. Some strings like 110 and 0010 for example are not in the language, so the DFA should not recognize or accept them, but it does.
 - h. This DFA is correct.