# label\_detection\_results/chapter3\_12\_6\_19/test\_12\_6\_19\_ch3\_bert.json

# atom 13

\begin{enumerate}[label=(\alph\*)]   
 \item Below, all missing transitions go to a rejecting sink state.  
 \begin{center}  
 \includegraphics[width=10cm]{dfa1.png}  
 \end{center}  
 \item Take the DFA above and flip the acceptingaccepting and rejecting statesrejecting states.  
 \item \  
 \begin{center}  
 \includegraphics[width=10cm]{dfa2.png}  
 \end{center}  
 \item Below, all missing transitions go to a rejecting sink state.  
 \begin{center}  
 \includegraphics[width=10cm]{dfa3.png}  
 \end{center}  
 \item \  
 \begin{center}  
 \includegraphics[width=10cm]{dfa4.png}  
 \end{center}  
 \end{enumerate}

# atom 14

Let $L$ be a finite language, i.e., it contains a finite number of words . Show that there is a DFA recognizing $L$.

# atom 15

Sorry, we currently do not have a solution for this exercise. But we are more than happy to discuss it with you during office hours.

# atom 16

A language $L \subseteq \Sigma^\*$ is called a \def{regular language} if there is a deterministic finite automaton $M$ such that $L = L(M)$.

# atom 17

All the languages in Exercise~\ref{exercise:Draw-DFAs} are regular languages.

# atom 18

Is the language  
\[  
 \{w \in \{\s{0},\s{1}\}^\* : w \text{ has an equal number of occurrences of $\s{01}$ and $\s{10}$ as substrings}\}  
\]  
regular?

# atom 19

The answer is yes because the language is exactly same as the language in Exercise~\ref{exercise:Draw-DFAs}, part (c).

# atom 20

Let $\Sigma = \{\s{0},\s{1}\}$. The language $L = \{\s{0}^n \s{1}^n: n \in \mathbb{N}\}$ is \textbf{not} regular.

# atom 21

Our goal is to show that $L = \{\s{0}^n \s{1}^n: n \in \mathbb{N}\}$ is not regular. The proof is by contradiction. So let's assume that $L$ is regular.   
  
Since $L$ is regular, by definition, there is some deterministic finite automaton $M$ that recognizes $L$. Let $k$ denote the number of states of $M$. For $n \in \mathbb{N}$, let $r\_n$ denote the state that $M$ reaches after reading $\s{0}^n$ (i.e., $r\_n = \delta(q\_0, \s{0}^n)$). By the pigeonhole principle,\footn{The \emph{pigeonhole principle} states that if $n$ items are put inside $m$ containers, and $n > m$, then there must be at least one container with more than one item. The name \emph{pigeonhole principle} comes from thinking of the items as pigeons, and the containers as holes. The pigeonhole principle is often abbreviated as PHP.} we know that there must be a repeat among $r\_0, r\_1,\ldots, r\_k$ (a sequence of $k+1$ states). In other words, there are indices $i, j \in \{0,1,\ldots,k\}$ with $i \neq j$ such that $r\_i = r\_j$. This means that the string $\s{0}^i$ and the string $\s{0}^j$ end up in the same state in $M$. Therefore $\s{0}^iw$ and $\s{0}^jw$, \emph{for any} string $w \in \{\s{0},\s{1}\}^\*$, end up in the same state in $M$. We'll now reach a contradiction, and conclude the proof, by considering a particular $w$ such that $\s{0}^iw$ and $\s{0}^jw$ end up in different states.   
  
Consider the string $w = \s{1}^i$. Then since $M$ recognizes $L$, we know $\s{0}^iw = \s{0}^i\s{1}^i$ must end up in an accepting stateacceptingacceptingaccepting stateaccepting stateaccepting state. On the other hand, since $i \neq j$, $\s{0}^jw = \s{0}^j\s{1}^i$ is not in the language, and therefore cannot end up in an accepting stateend up in an accepting stateacceptingacceptingaccepting stateaccepting stateaccepting state. This is the desired contradiction.