1. Design deterministic finite automata to recognize the languages below. Test your DFA’s using Prolog on the indicated strings in the language, as well as some strings which are not in the language.

   (a) \( L = \{ w \mid w \in \{0,1\}^*, w \text{ contains no more than one occurrence of the string } 00 \} \). Note that the string 000 should be regarded as having 2 occurrences of 00. Example strings include 00, 1010, and 11100101110.

   (b) \( L = \{ w \mid w \in (0+1)^{++}(0+1)^+, w \text{ represents an even binary number after the addition} \} \). Note that the + is a ‘+’ symbol, not a regular expression union operator. Example strings include 0+0, 1+1, and 1110010+1110.

2. Design nondeterministic finite automata to recognize the languages below. Test your NFA’s using Prolog on the indicated strings in the language, as well as some strings which are not in the language.

   (a) \( L = \{ w \mid w \in \{0,1\}^*, \text{the 3rd symbol from the right of } w \text{ is a } 0 \} \). Example strings include 000, 100001, and 011010.

   (b) \( L = \{ w \mid w \in \{0,1\}^*, w \text{ contains both } 101 \text{ and } 010 \text{ as substrings} \} \). Example strings include 1010, 101010, and 0010011011.

3. Convert the NFA \( M = (\{q_1, q_2, q_3, q_4, q_5, q_6, q_7\}, \{a, b\}, \delta, q_1, \{q_2, q_6\}) \), where \( \delta \) is:

   \[
   \begin{align*}
   \delta(q_1, \epsilon) &= \{q_2, q_3\} \\
   \delta(q_1, a) &= \{q_1\} \\
   \delta(q_2, a) &= \{q_3\} \\
   \delta(q_3, b) &= \{q_4\} \\
   \delta(q_4, b) &= \{q_2\} \\
   \delta(q_5, \epsilon) &= \{q_6\} \\
   \delta(q_5, b) &= \{q_5\} \\
   \delta(q_6, b) &= \{q_7\} \\
   \delta(q_7, a) &= \{q_6\}
   \end{align*}
   \]

   into an equivalent DFA.

4. A deterministic finite transducer is a 6-tuple \( M = (Q, \Sigma, \Delta, \delta, q_0, F) \), where

   1) \( Q \) is a finite set of states.
   2) \( \Sigma \) is a finite input alphabet.
   3) \( \Delta \) is a finite output alphabet.
   4) \( \delta \) is a mapping from \( Q \times (\Sigma \cup \{\epsilon\}) \) to \( Q \times \Delta^* \). Determinism implies that 1) either \( \delta(q, a) \) contains at most one element for each \( a \in \Sigma \), and \( \delta(q, \epsilon) = \emptyset \), or 2) \( \delta(q, \epsilon) \) contains one element, and for all \( a \in \Sigma \), \( \delta(q, a) = \emptyset \). That is, there can be no moves involving a choice between making an \( \epsilon \)-move or consuming an input symbol.
   5) \( q_0 \in Q \) is the initial state.
   6) \( F \subseteq Q \) is the set of final states.
Note that a finite transducer accepts its input when it reaches a final state and outputs a string over $\Delta^*$ for each input string accepted.

(a) Construct a deterministic finite transducer whose input is of the form $(0+1)^*$, and whose output is the same as the input on the even positions but inverted on the odd positions. For example, on input 0, the output is 1, on input 000111, the output is 1010010, and on input 1110, the output is 01011.

(b) Test your DFT using Prolog on the above example strings in the language. Note that a DFT transition of the form $\delta(q_0, a) = (q_1, bc)$ is written in Prolog as:

```prolog
delta(q0, [a | RestOfInput], InitialOutput, FinalOutput) :-
    append(InitialOutput, [b, c], NewOutput),
    delta(q1, RestOfInput, NewOutput, FinalOutput).
```

and a final state is represented by:

```prolog
delta(qf, [], FinalOutput, FinalOutput).
```

To run on the third string above, use the goal:

```prolog
delta(q0, [1, 1, 1, 1, 0], [], Output).
```

The others are similar.