1-Show how the function \( f(w_1, w_2, w_3) = \sum m(0,2,3,4,5,7) \) can be implemented using a 3-to-8 binary decoder and an OR gate.

2-Braille is a system of raised dots that can be read by a blind person. You are asked to design an encoder circuit that converts Binary Coded Decimal (BCD) numbers to Braille. The Braille patterns for the BCD numbers are shown below in Fig. 1.

![Braille patterns](image)

Fig. 1. Braille representation of digits 0 to 9.

Derive a minimum sum-of-product form representation for each of the four Braille dot outputs \( X, Y, W, Z \) in terms of a 4-bit BCD number. Denote the 4-bit BCD number as \( ABCD \) where \( A \) is the most significant bit, and \( D \) the least significant bit.

3- Draw the schematic and truth table of “Ring Counter”. Also represent the output.
4- Draw the schematic and truth table of “Johnson Counter”. Also represent the output.
5- Drive a 16*1 multiplexer using:
   A) 8*1 multiplexers.
   B) 16*1 multiplexers.
6- Implement these gates with: I) NAND gate. II) NOR gate.
   A) 2-input AND
   B) 2-input OR
   C) NOT
   D) 2-input XOR
   E) 2- input XNOR

Redo this problem with 3-input gates.

7- Derive the state diagram for an FSM that has an input \( w \) and an output \( z \). The machine has to generate \( z=1 \) when the previous four values of \( w \) were 1001 or 1111; otherwise, \( z=0 \). Overlapping input patterns are allowed. Example:

\[
\begin{align*}
  w: & 0101110110011111 \\
  z: & 00000100100010011
\end{align*}
\]

8- Drive the circuit that realizes the FSM of problem above, using:
   I) SR Flip-Flop
   II) D Flip-Flop
   III) JK Flip-Flop
   IV) T Flip-Flop

9- A sequential circuit has two inputs, \( w_1, w_2 \), and an output, \( z \). Its function is to compare the input sequences on the two inputs. If \( w_1 = w_2 \) during any four consecutive clock cycles, the circuit produces \( z=1 \); otherwise, \( z=0 \). Example:

\[
\begin{align*}
  w_1: & 0110111001110 \\
  w_2: & 1110101000111
\end{align*}
\]
10- Implement a half-adder with multiplexer. (2-4*1 mux)
11- Implement a full-adder with multiplexer. (2-8*1 mux)
12- Implement a four-bit-adder with result of problem 11.