# students at that grade

Average: 23.5
3/4 >= 22
1/2 >= 25
1/4 >= 26
1.) For the following diagram, use the laws of Boolean Algebra to minimize this circuit. You do not have to name the laws used, but each step must be obvious to someone who knows the laws. Your solution should be in the form of a Boolean Equation.
2.) Simplify the following K-maps, and write the corresponding Boolean Equation. You should create the simplest Sum of Products form possible.
<table>
<thead>
<tr>
<th>A</th>
<th>1</th>
<th>0</th>
<th>1</th>
<th>X</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>C</td>
<td>X</td>
<td>0</td>
<td>X</td>
<td>0</td>
</tr>
<tr>
<td>D</td>
<td>0</td>
<td>X</td>
<td>1</td>
<td>X</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A</th>
<th>X</th>
<th>0</th>
<th>0</th>
<th>X</th>
</tr>
</thead>
<tbody>
<tr>
<td>B</td>
<td>1</td>
<td>X</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>X</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>D</td>
<td>0</td>
<td>X</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
3.) Draw the state diagram for the following circuit. YOU DO NOT HAVE TO DO THE STATE TABLE. YOU DO NOT HAVE TO IMPLEMENT THE CIRCUIT.

Design a circuit that has a 1-bit input. It outputs 1 every time when the last three input values seen (2 previous + the current) are the same, and 0 otherwise. At startup assume that the two previous input values were 1’s.
4.) For the following state diagram, implement the circuit. The state encoding is given to you in the parentheses in each state. You may use premade D-flipflops and any other basic gates. Your implementation should be as simple as possible.

\[
\begin{array}{c|c|c}
\text{PS} & \text{IN} & \text{NS} \\
0 & 0 & 1 \\
0 & 1 & 0 \\
1 & 0 & 0 \\
1 & 1 & 1 \\
\end{array}
\]

\[\text{NS} = \text{PS} + \text{IN}\]

\[\text{OUT} = \overline{\text{PS IN}}\]
5.) For the Verilog given below, draw the corresponding finite state machine. DO NOT IMPLEMENT THE CIRCUIT. The code below is correct, legal Verilog.

```verilog
module machine (clk, reset, in, out);
  input logic clk, reset, in;
  output logic out;

  logic [1:0] ps;
  logic [1:0] ns;

  always_comb begin
    ns[0] = (ps[0] & ps[1]) | in;
    out = ps[1];
  end

  always_ff @(posedge clk) begin
    if (reset)
      ps <= 2'b11;
    else
      ps <= ns;
  end
endmodule
```

**State Diagram:**

![State Diagram Image]
(and back)
Approach: From decimal to binary numbers

Missing: Way to implement numbers in binary
Inputs: Switches
Display: Seven segment displays
Need: Display, address & subtractors, inputs
Problem: Implement simple pocket calculator

Readings: 3.3.3, 3.3.3, 3.3.3

Number Systems
Decimals (Base 10) Numbers

Alternate view: Digit Position I from the right = 0

Digit * 10\(^0\) (rightmost is position 0) = 0

2534 = 2*1000 + 5*100 + 3*10 + 4*1

Headecimal / Headex (6)

Base 2: Binary

2534 = 2*103 + 5*102 + 3*10 + 4*100
Base R Numbers

D_3 D_2 D_1 D_0 \text{(base R)} = D_3 R^3 + D_2 R^2 + D_1 R^1 + D_0 R^0

Digit position \( I = \text{Digit} \times R^I \)

\[
\begin{align*}
E & = 15 \\
D & = 14 \\
C & = 13 \\
B & = 12 \\
A & = 10 \\
\end{align*}
\]

Each digit in range 0..(R-1)

... 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F
Hexadecimal: (E32)₁₀

46₁₀

46 = 2 + 4 + 8 + 32

6 = 2 × 1 + 3 × 16 + 1 × 256

3639₁₀

= 2 × 1 + 3 × 16 + 1 × 256

= 2 × 1 + 3 × 16 + 14 × 16 + 1 × 16

Binary: (101110)₂

Number System (Conversion to Decimal)
Hexadecimal: (A6)_{16} = 6 + 10 \times 16^0

Binary: (110101)_2 = 1 \times 4 + 16 + 32

Conversion from Base R to Decimal
$$\begin{align*}
2_{3} & = 8 \\
2_{2} & = 4 \\
2_{1} & = 2 \\
2_{0} & = 1 \\
2_{8} & = 2^{5} \\
2_{7} & = 2^{12} \\
2_{6} & = 2^{6} \\
2_{5} & = 2^{3} \\
2_{4} & = 2^{1} \\
2_{13} & = 8^{192} (8K) \\
2_{12} & = 4^{96} (4K) \\
2_{11} & = 2048 (2K) \\
2_{10} & = 1024 (1K) \\
2_{9} & = 512 \\
2_{8} & = 256 \\
2_{7} & = 128 \\
2_{6} & = 64 \\
2_{5} & = 32 \\
2_{4} & = 16 \\
2_{3} & = 8 \\
\end{align*}$$

corresponding digit position

less than the number from the value. Put a 1 in the
Successfully subtract the greatest power of two

For positive, unsigned numbers

Convesion of Decimal to Binary (Method 1)
Convert (289)\text{10} to binary

\[
\begin{array}{c|c}
2^0 & 1 \\
2^1 & 0 \\
2^2 & 0 \\
2^3 & 1 \\
2^4 & 0 \\
\hline
\end{array}
\]

Convert (2578)\text{10} to binary

\[
\begin{array}{c|c}
2^0 & 0 \\
2^1 & 8 \\
2^2 & 1 \\
2^3 & 7 \\
2^4 & 8 \\
2^5 & 0 \\
\hline
\end{array}
\]

Decimal to Binary Method I
1010 0011 1111 1111 0110 1010
A 3 FE 2 A

Convert (A3FE2A)₁₆ to binary

E 35 D3₁₆

Convert (11100011010111001111)₂ to hex

E 3 5 D 3

1 hex digit = 4 binary digits

Converting Binary to Hexadecimal
Convert (198)\textsubscript{10} to Hexadecimal

Convert to binary, then to Hex

Converting Decimal to Hex