1	/12
2	/18
3	/12
4	/17
5	/15
6	/15

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
DEPARTMENT OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE

6.191 Computation Structures
Updated Fall 2022

Quiz #1

Name **Solutions**	*Athena login name*	*Score*

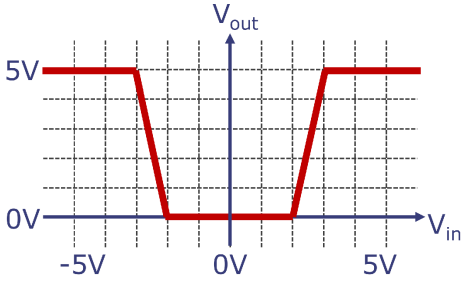
Recitation section

☐ WF 10, 34-301 (Grace)	☐ WF 2, 13-5101 (Frances)	☐ WF 12, 36-155 (Shiqi)
☐ WF 11, 34-301 (Grace)	☐ WF 3, 13-5101 (Frances)	☐ WF 1, 36-155 (Shiqi)
☐ WF 12, 35-310 (Alexandra)	☐ WF 10, 13-4101 (Georgia)	☐ WF 1, 34-303 (Amelia)
☐ WF 1, 35-308 (Alexandra)	☐ WF 11, 13-4101 (Georgia)	☐ WF 2, 34-303 (Amelia)
		☐ opt-out

Please enter your name, Athena login name, and recitation section above. Enter your answers in the spaces provided below. Show your work for potential partial credit. You can use the extra white space and the backs of the pages for scratch work.

Problem 1. The ¯_(ツ)_/¯ Abstraction (12 points)

Device X has the Voltage Transfer Characteristic (VTC) given below:



$$V_{out} = \begin{cases} 5V & V_{in} \le -3V \\ 5(-V_{in} - 2V) & -3V \le V_{in} \le -2V \\ 0V & -2V \le V_{in} \le 2V \\ 5(V_{in} - 2V) & 2V \le V_{in} \le 3V \\ 5V & V_{in} \ge 3V \end{cases}$$

We want to use Device X to build an XOR gate.

(A) (9 points) Consider the circuit shown below. We want to analyze three candidate signaling specifications. We consider a signaling specification *valid* if and only if the circuit behaves like a digital XOR gate. Find whether each signaling specification is valid, briefly explaining why or why not. If the specification is valid, give its noise immunity (smallest noise margin).

For this analysis, assume that input voltages are always between 0V and 5V (otherwise, a very low or very high voltage at one of the inputs could make the circuit misbehave).

	Spec A	**Spec B**	**Spec C**
V_{OL}	**0V**	**0V**	**0V**
V_{IL}	**0.5V**	**1V**	**2V**

V_{IH}	4.5V	4V	3V
V_{OH}	5V	5V	5V

Spec A: Noise immunity or invalid spec? ____0.5V_____
Brief explanation for why valid/invalid:

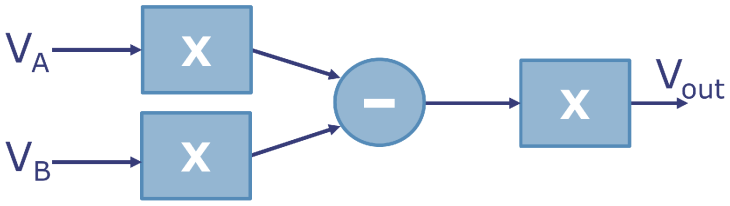
Vin in range (-0.5V, 0.5V) if both inputs are the same, at least 4.5V-0.5V=4V for A=1 B=0, and at most -4V for A=0 B=1

Spec B: Noise immunity or invalid spec? ____1V_____
Brief explanation for why valid/invalid:

Vin in range (-1V, 1V) if both inputs are the same, at least 4V-1V=3V for A=1 B=0, and at most -3V for A = 0 B=1

Spec C: Noise immunity or invalid spec? __INVALID___
Brief explanation for why valid/invalid:
With A=1 B=0, Vin can be as low as 3V-2V=1V, which gives a digital 0 at the output (-> not a XOR). Moreover, valid digital inputs can produce invalid outputs (e.g., VA=4V, VB=1.5V).

(B) (3 points) Consider the circuit shown below. Find the signaling specification that makes this circuit behave like an XOR gate and maximizes noise immunity. As before, **assume that input voltages are always between 0V and 5V**.



Signaling specification: V_{OL} = __0__V V_{IL} = __2.2__V V_{IH} = __2.8__V V_{OH} = __5__V

Noise Immunity: __2.2___ V

With Vin=2.2V (2.8V) an X device produces Vout=1V (4V), which exercise the maximum voltage ranges at the inputs of the circuit from part (A) that make it behave like an XOR gate.

Problem 2. Boolean Algebra (18 points)

(A) (8 points) Simplify the following Boolean expressions by finding a minimal sum-of-products expression for each one. (Note: These expressions can be reduced into a minimal SOP by repeatedly applying the Boolean algebra properties we saw in lecture.)

1. $\overline{(\bar{x}z + \bar{y}z)} = \overline{(\bar{x} + \bar{y})z} = xy + \bar{z}$

2. $x + z\left(y + \overline{(yz)}\right) = x + z(y + \bar{y} + \bar{z}) = x + z$

3. $\bar{x} + \bar{y} + xy = \overline{(xy)} + xy = 1$

4. $yz(\bar{y} + \bar{x}) + \bar{x}y = \bar{y}yz + \bar{x}yz + \bar{x}y = \bar{x}y(z + 1) = \bar{x}y$

(B) (2 points) You are given the truth table for a circuit that takes a 3-bit unsigned binary input (X = ABC), multiplies it by 2 mod 8 and adds 1 mod 8 to it to produce a 3-bit unsigned binary output (Y = A'B'C').

A	B	C	A'	B'	C'
0	0	0	0	0	1
0	0	1	0	1	1
0	1	0	1	0	1
0	1	1	1	1	1
1	0	0	0	0	1
1	0	1	0	1	1
1	1	0	1	0	1
1	1	1	1	1	1

For the above truth table, write out a **minimal sum-of-products** for each function A'(A,B,C), B'(A,B,C), and C'(A,B,C)

Minimal sum-of-products for A'(A,B,C)=_____**B**_____

Minimal sum-of-products for B'(A,B,C)=_____**C**_____

Minimal sum-of-products for C'(A,B,C)=_____**1**_____

(C) (3 points) Now consider a new function G(A, B, C) defined by the truth table below. Find the normal form and a minimal sum-of-products expression for G(A, B, C).

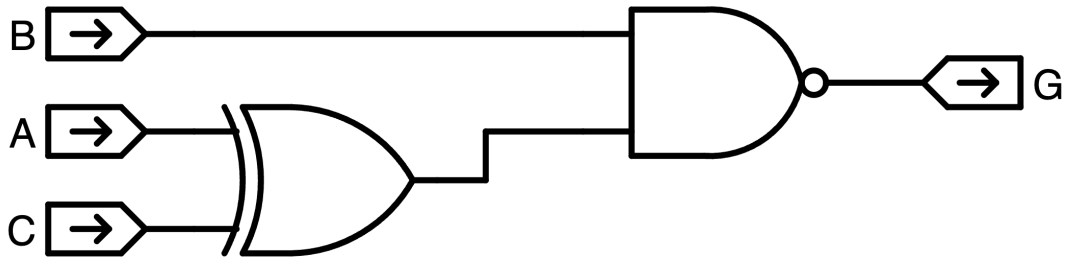
A	B	C	G
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1

1. **Normal form for G =** _____ $\bar{a}\bar{b}\bar{c} + \bar{a}\bar{b}c + \bar{a}b\bar{c} + a\bar{b}\bar{c} + a\bar{b}c + abc$ _____

2. **Minimal sum of products for G =** _____ $\bar{b} + \bar{a}\bar{c} + ac$ _____

$$\bar{a}\bar{b}\bar{c} + \bar{a}\bar{b}c + \bar{a}b\bar{c} + a\bar{b}\bar{c} + a\bar{b}c + abc = \bar{b} + \bar{a}\bar{c} + ac$$

(D) (3 points) Draw the circuit that implements the minimal sum of products you derived for G **using the fewest number of gates**. You may use 2-input OR, NOR, AND, NAND, XOR, and XNOR, and inverters in your circuit.



(E) (2 points) Below you are given the delays for the different gates you were permitted to use in part D above. Compute the propagation delay of your circuit from D.

Gate	**t_{PD} (ns)**
XNOR2	7.0
XOR2	6.5
NOR2	6.0
OR2	5.5
AND2	5.0
NAND2	3.0
INV	2.0

Longest path:
a/c → XOR2 → NAND2 → G
6.5 + 3.0 = 9.5ns

t_{PD} (ns) = _____**9.5**_____

Problem 3. CMOS Logic (12 points)

Ben invented a light-speed spaceship last night, using a CMOS gate as the critical component. He wrote down the truth table for the Boolean expression implemented by this gate. Unfortunately, his nemesis replaced one of the entries in his truth table and Ben can't remember which one it was! Fortunately, you have taken 6.191 and can help Ben reconstruct the truth table.

Below is the truth table for Ben's Boolean expression, $F(a, b, c)$. One entry of the truth table has been modified. **Ben surmises that his nemesis has flipped one of the outputs from a 1 to a 0**.

(A) (3 points) Circle the entry in the truth table which has been modified and explain why it must have been incorrect.

F(0, 1, 0) -> F(1, 1, 0) should follow the "rising inputs lead to falling outputs" rule. However, the truth table has F(0, 1, 0) = 0 and F(1, 1, 0) = 1, which is not possible under this rule.

Since we know Ben's nemesis flipped an output from a 1 to a 0, it must be that F(0, 1, 0) is the incorrect line – its output should really be a 1.

a	b	c	F
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	0

(B) (1 point) Ben thinks he can make the spaceship even faster if he sets F(1, 1, 1) = 1. Can Ben still implement this with a single CMOS gate?

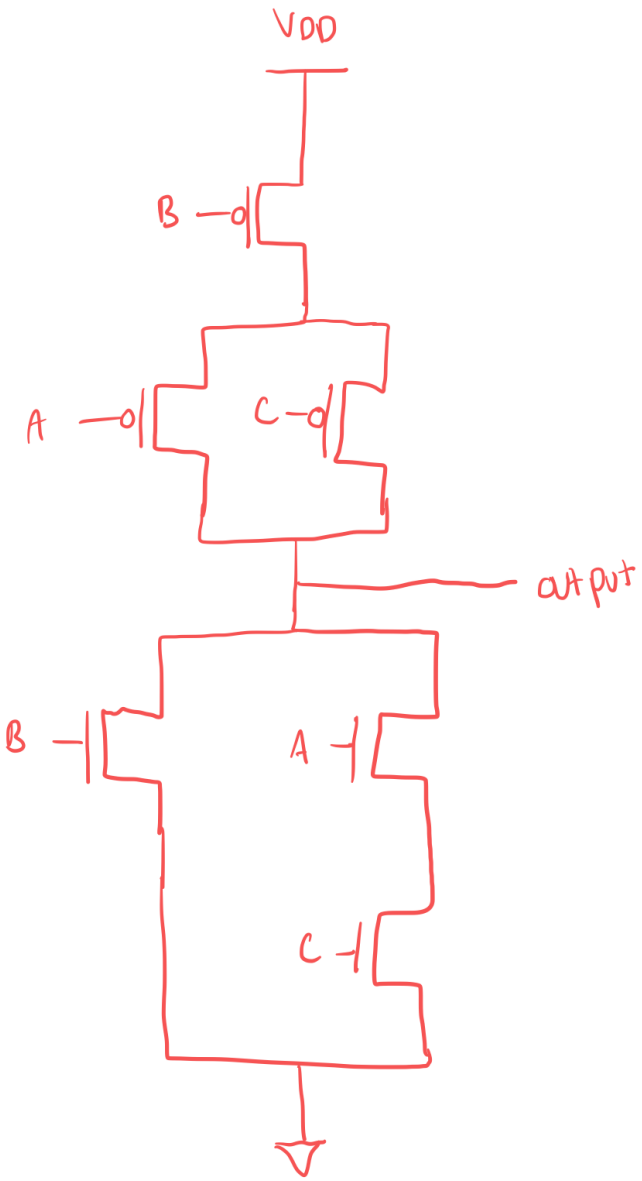
(circle one) Yes No

(C) (4 points) Ben wants to add an input d to his CMOS gate to implement a new function, G. Ben sets G(a, b, c, 0) = F(a, b, c). Given that G can be implemented as a single CMOS gate, what are the following values?

(circle one) G(0, 0, 1, 1) = : 0 … 1 … (can't say)

(circle one) G(1, 0, 1, 1) = : 0 … 1 … (can't say)

(D) (4 points) Alice thinks Ben should use a different design. She proposes using the Boolean expression $\bar{B}(\bar{A} + \bar{C})$. Draw the CMOS gate for Alice's Boolean expression.

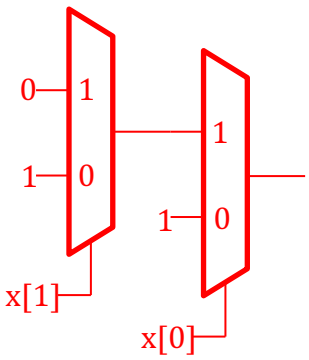


Problem 4. Combinational Circuits (17 points)

A fibbinary number is any number whose binary representation does not contain any sequences of adjacent 1's. For example, 1 (0b001), 2 (0b010), and 5 (0b101) are fibbinary, but 3 (0b011) is not. Our goal is to design some combinational circuits that check whether a number is fibbinary.

(A) (3 points) You first want to implement a simple **2-bit fibbinary checker** in hardware, but Logic Gates 'R' Us is sold out of every logic gate except 2-input muxes! Can we still implement a circuit that will output a 1 if a 2-bit input x is a fibbinary number and 0 otherwise using only 2-input muxes? If yes, draw the circuit using the **minimum** number of muxes. If no, explain why not.

x[1]	x[0]	out
0	0	1
0	1	1
1	0	1
1	1	0



(B) (3 points) Implement the function `isFibbinary3` in Minispec, which returns 1 if its 3-bit input `x` is a fibbinary number and 0 otherwise. For full credit, use only logical operators (NOT (~), AND (&), OR (|), XOR (^)).

```
function Bit#(1) isFibbinary3(Bit#(3) x);
   return ~(x[2] & x[1] | x[1] & x[0]);
endfunction
```

(C) (5 points) Implement `isFibbinary#(n)` in Minispec using only a for loop and logical operators (NOT (~), AND (&), OR (|), XOR (^)). `isFibbinary#(n)` takes in an n-bit input `x` and returns 1 if it is a fibbinary number and 0 otherwise. Assume **n is greater than or equal to 2.** You may declare variables and return values outside of the for loop.

```
function Bit#(1) isFibbinary#(Integer n)(Bit#(n) x);
   Bit#(1) isFib = 1;
   for (_Integer i = 0_; _ i < n - 1_; _ i = i + 1_) begin
       isFib = isFib & ~(x[i] & x[i + 1])
   end
   return isFib;
endfunction
```

(D) (2 points) How does the delay of `isFibbinary#(n)` grow with input width `n`?

The number of iterations through the for loop is proportional to n.

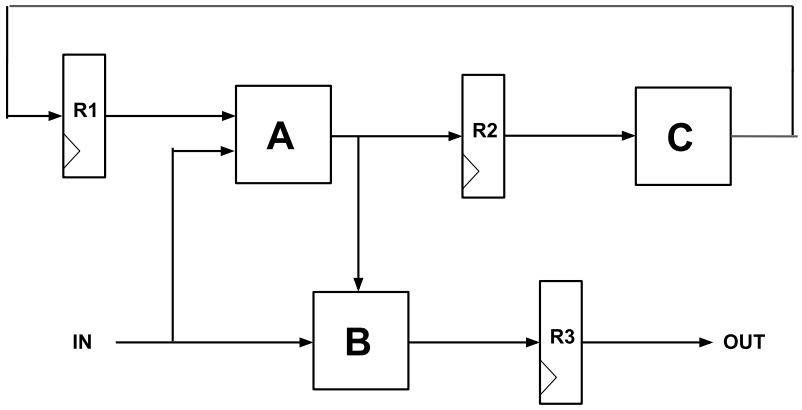
Delay complexity: O(__n___)

(E) (4 points) Your friend points out that there's a faster way to check for a fibbinary number if we don't limit ourselves to using only logical operators. Complete the single-line implementation of `isFibbinary#(n)` using any combination of the following binary operators: &, |, ^, ~, <<, >>, ==, !=. *Hint:* Some shifting is involved.

```
function Bit#(1) isFibbinary#(Integer n)(Bit#(n) x);
   return (__((x << 1) & x) == 0__)? 1'b1: 1'b0;
endfunction
```

Problem 5. Combinational and Sequential Logic Timing (15 Points)

Consider the sequential circuit below, as well as the timing specifications. Registers R1, R2, and R3 are driven by a common clock. **A**, **B**, and **C** are combinational circuits.



	t_{pd}	t_{cd}	t_{setup}	t_{hold}
Register (R1, R2, R3)	5ns	1ns	7ns	2ns
Circuit A	3ns	**?**	--	--
Circuit B	4ns	2ns	--	--
Circuit C	6ns	2.5ns	--	--

(A) (3 points) What are t_{pd} and t_{cd} of this sequential circuit?

t_{pd} (ns): _____5_____

t_{cd} (ns): _____1_____

(B) (3 points) What is the smallest value for t_{cd} of Circuit A that will allow this circuit to operate correctly? Show your work.

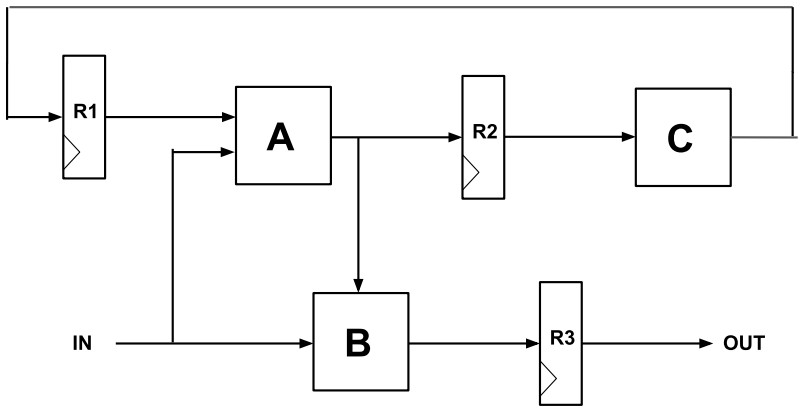
$t_{cd,R1} + t_{cd,A} \geq t_{hold,R2}$
$1 + t_{cd,A} \geq 2$
$t_{cd,A} \geq 1ns$

t_{cd} of Circuit A (ns): _____1_____

(C) (4 points) What is the shortest clock period that can be used to drive all of the registers in the circuit? Show your work.

$t_{clk} \geq t_{pd,R1} + t_{pd,A} + t_{pd,B} + t_{setup,R3}$
$t_{clk} \geq 5 + 3 + 4 + 7$
$t_{clk} \geq 19ns$

t_{clk} of circuit (ns): _____19_____



ORIGINAL	t_{pd}	t_{cd}	t_{setup}	t_{hold}
Register (R1, R2, R3)	5ns	1ns	7ns	2ns
Circuit A	3ns	**?**	--	--
Circuit B	4ns	2ns	--	--
Circuit C	6ns	2.5ns	--	--

The table above shows the original specs of our circuit. We now find that our supplier has the following alternative circuits for A, B and C available with the following specifications.

	t_{pd}	t_{cd}	t_{setup}	t_{hold}
A-New	1ns	0.5ns	--	--
B-New	2.5ns	2ns	--	--
C-New	2ns	1ns	--	--

(D) (5 points) These new circuits aren't cheap, so you may only replace *one circuit* in order to minimize clock period. Please indicate which combinational circuit you are replacing, and the resulting minimum clock period. Explain why your selected component is a better choice than the other two. You should explicitly compare all three components in your explanation.

Combinational circuit replaced: _____B_____

t_{clk} of circuit (ns): _____18_____

Explanation:
To minimize $t_{clk} \geq t_{pd,R1} + t_{pd,A} + t_{pd,B} + t_{setup,R3}$ we want to replace A or B. Can't replace A with A-New because contamination delay of 0.5 would not satisfy the hold time for R2. So, we want to replace B since it is the other component in the critical path. Replacing B reduces the R1-R3 path to 5 + 3 + 2.5 + 7 = 17.5ns. However, the critical path is now the R2-R1 path which is 5 + 6 + 7 = 18.

Problem 6. Finite State Machine (15 points)

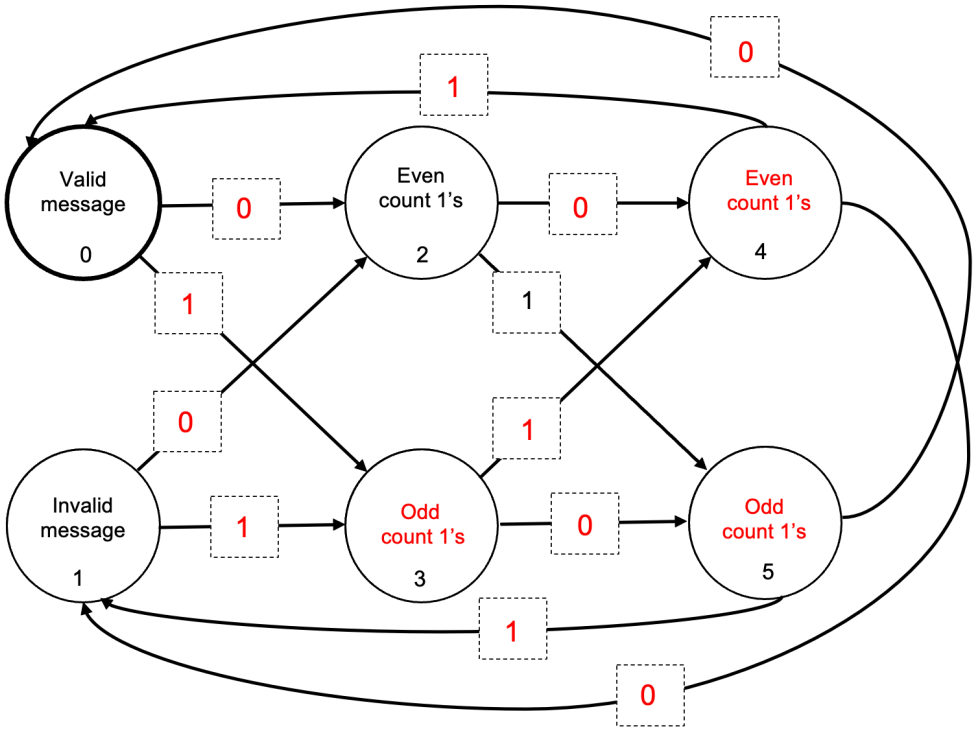
Rettiwt Complaint Hotline Operator is implementing a new message transfer protocol to help him detect corrupted messages:

- Each message will have 3 bits.
- The first two bits are data bits.
- The third bit is a parity bit that should make the total number of 1's (including the parity bit) in the message be odd.
- Messages are sent bit by bit.

With this specification, a message is valid if and only if it contains an odd number of 1's. For example, 111 and 010 are valid messages, but 101 is an invalid message.

The output of the FSM is set to 1 for one clock cycle on the next rising edge of the clock **after an incorrect parity bit is received**. The FSM should output a 0 at all other times. The bit received immediately after the parity is treated as the beginning of a new message regardless of the correctness of the previous message.

(A) (6 points) We provide a partially complete state transition diagram. Each circle corresponds to the state the FSM is currently in. The Hotline Operator provided annotations for some of the states. Fill in the missing inputs corresponding to each of the arrows connecting the states. The FSM begins at state 0 (Valid Message).



(B) (3 points) Fill in the missing data in the following **truth table** for this finite state machine. Use the integers in the FSM states to identify the current and next state.

Current State	Input	Next State	Output
0	0	2	0
0	1	3	0
1	0	2	1
1	1	3	1
2	0	4	0
2	1	5	0
3	0	5	0
3	1	4	0
4	0	1	0
4	1	0	0
5	0	0	0
5	1	1	0

(C) (6 points) Assume your FSM begins at state 0 before processing each of the following messages. What is the state and output of the FSM on the next rising edge of the clock **after receiving the last bit** of each message?

1. 001_011_111_10 Output:_____**0**_____ State:_____**5**_____
2. 110_010_010_110_011 Output:_____**1**_____ State:_____**1**_____
3. 110_110_100_111_0 Output:_____**0**_____ State:_____**2**_____

END OF QUIZ 1!