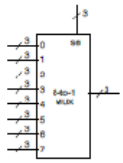
1. (30 pts) A k-bit incrementer circuit takes in a k-bit value $X = X_{k-1}X_{k-2}\cdots X_1X_0$ and (unsigned binary) adds 1 to it to produce $Y = Y_{k-1}Y_{k-2}\cdots Y_1Y_0$. For instance, for $k = 8$, if $X = 00110111$, the incremented value would be $Y = 00111000$. If X is all 1's, then Y should be all 0's (we don't worry about overflow).

   (a) (15 pts) Describe the value of the ith output bit algebraically for any $i > 1$ as a function of the input bits $\{X_i\}$.

   (b) (15 pts) Build a 3-bit incrementer using a 3-bit 8-to-1 MUX, as pictured. Show how to increment an input $X = X_2X_1X_0$.

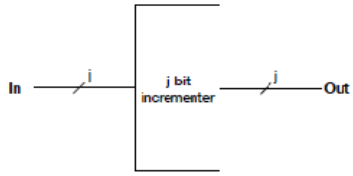


2. (30 pts) $X = X_{k-1}X_{k-2}\cdots X_1X_0$ is a k-bit number provided in 1's complement representation. In this problem, you are to convert the number to 2-s complement representation.

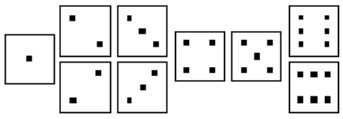
   (a) (10 pts) Describe the algorithm (i.e., in simple pseudocode) where you can reference the entire input X as a variable or the individual bits $\{X_i\}$ in your pseudocode. Demonstrate the correctness of your solution when $k = 4$ for two cases: $X = \quad 1$ and $X = 1$.

   (b) (10 pts) Design a circuit that converts X into 2's-complement form. You may use any of the basic gates (AND, OR, NOT, XOR) or standard circuitry (MUX,Decoder, Enabler, half-adder, full-adder, ripple-carry adder) needed to design the conversion circuit.

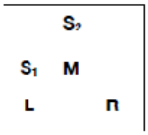
   (c) (10 pts) Implement the converter using a multi-bit incrementer (i.e., described in problem 1, and depicted below as a j-bit incrementer), a multi-bit 2-to-1 MUX, and any basic gates (AND, OR, NOT, XOR) you additionally require.



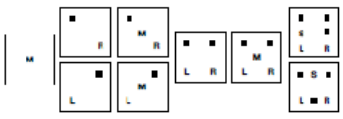
3. (35 pts) Professor Rubenstein's continuing obsession with dice games has him (making you) build a device that can scan the number of dots on the top of a 6-sided die to determine its current rolled value. The scanner is placed parallel to the die, such that the possible orientations are possible for each value. 2, 3 and 6 have two possible orientations depending on the rotation, and other values have only a single orientation.



To read the die, 5 scanners are employed. Each scanner views a particular part of the upward face, and outputs a 1 when the portion of the face viewed contains a dot. These five outputs are labeled M (for middle scan), L (for bottom left scan), R (for bottom right scan), and S_1 and S_2 (for scans that can pick up when the die shows a six). The values provided to the circuit designer are M, L, R, and $S = S_1 + S_2$.



You are to help design a circuit that takes inputs M, L, R and S, and outputs a 3-bit unsigned binary, ABC, equal to the value of the current roll. Each configuration above produces a unique assignment of M, L, R and S. For instance, if only M is 1, then the value depicted on the die must be 1. If only $M = 1$ and $L = 1$, the die value is 3. If only $M = 1$ and $R = 1$, the die value is also 3, just rotated the other way. The figure below shows, for each orientation listed in the top figure, which input values equate to 1 for that orientation.



Provide simplified (in SoP form) expressions of the 3-bit binary die value in terms of these 4 scanner values M, L, R, S.

(35 pts) Professor Rubenstein wants to start a gaming company called FunForTwoHands: automation for games involving 2 people and fun things they can do together, with one hand each.

Are you thinking what I'm thinking? Yes - the initial product will implement the 2-player hand game of Rock-Paper-Scissors, helping to identify the winner of each round.

The game of Rock-Paper-Scissors is played by two players (player 0 and player 1), where each round, the two players each simultaneously select one of the three options: Rock, Paper or Scissors. If both players choose the same object, the players tie. Otherwise, the winner is chosen as follows:

- Paper covers Rock (player choosing paper beats player choosing rock)
- Scissors cuts paper (player choosing scissors beats player choosing paper)
- Rock smashes scissors (player choosing rock beats player choosing scissors)

To automate the process, player 0's choice is described by the two-bit input AB, and player 1's choice by CD, where the 2-bit input indicates the following choice:

2-bit input		choice indicated
0	0	Rock
0	1	Paper
1	0	Scissors
1	1	(Unused)

Given inputs AB and CD that select one of the three 2-bit options above (i.e., you **should** assume that $AB = 11$ or $CD = 11$ will never be provided as an input), 2 outputs can be generated:

- T: outputs 1 when there is a tie, 0 otherwise.
- W: identifies the winner when the selections do not result in a tie (i.e., equals 0 when player 0 wins, 1 when player 1 wins).

For instance, if $ABCD = 0001$, then $T = 0$ and $W = 1$ since player 0 chose rock, and player 1 chose paper, and paper covers rock, hence player 1 wins.

Note that in the event of a tie, the W output is meaningless, since there is no winner.

As the circuit designer for Prof. Rubenstein, you need to provide **simplified SoP algebraic equations** for T and W as a function of the 4 inputs, A, B, C, D. Make sure your equations will produce circuits that, considered individually, are as simple as possible (in SoP form).

2. (35 pts) A k-UnsignedGreaterThan circuit (UGT_k for short) takes two k-bit inputs, $B = B_{k-1}B_{k-2}\cdots B_1B_0$ and $A = A_{k-1}A_{k-2}\cdots A_1A_0$, and returns 1 (TRUE) when $B > A$ where both B and A represent *unsigned k-bit binary values*. The circuit can be built recursively via a combination of circuitry and UGT_{k-1} circuits.

   (a) (5 pts) Show a simplified design for the base case: a UGT_1 circuit, built using only AND, OR, and NOT gates, that returns TRUE when $B_0 > A_0$ (note, we are requiring strictly greater here).

   (b) (8 pts) Now show how to build a UGT_k circuit for $k > 1$ recursively by using one or more UGT_{k-1} circuits and additional basic gates (i.e., AND, OR, NOT gates). Make sure your design clearly indicates which inputs feed in where. You should represent a UGT_{k-1} circuit as shown in the figure above when using it within your UGT_k circuit.

   (c) (7 pts) Now show how to build a general UGT_k recursively with one or more UGT_{k-1} circuits and 4 to 1 MUXes. Remember to clearly indicate how the inputs are feeding into the MUXes and the UGT_{k-1} circuits.

   (d) (8 pts) Suppose instead we wish to build a circuit that treats k-bit inputs A and B as signed 2s complement values. Show how to modify the design in part 2c for this purpose. Make sure you clearly indicate for which i the UGT_i can stay the same and for which it must be modified (as well as showing the modification).

   (e) (7 pts) Returning back to part (2b), suppose the circuit is represented in sum-of-products form (without further simplification). What circuit depth is achievable for the above design, where each AND, NOT, or OR gate used in series increases depth, and where AND and OR gates can take at most 2 inputs? Explain (in one or two sentences) why it has the depth you are claiming.
   (Note: if you are finding part (e) difficult, realize it's only 7 points. Perhaps just move onto the next question, and come back after that one. Partial credit can also be received for answers that are approximately right, i.e., having the right intuition).

2. (20 pts) After scanning a roll of 2 dice, the robot generates a 4-bit input A, B, C, D which indicates in unsigned binary the value of the roll (i.e., the sum of two dice addends. For instance, if the roll produces die values of 5 and 6, then $ABCD = 1011$ represents a roll that totals to 11. Note the 4-bit input $ABCD$ will always be in the range of 2 through 12.

   Let's first build circuits that evaluate the first roll of these two dice, and determine whether that first roll would end the game and if so, whether the gambler won. These circuits should be SoP-simplified, meaning that if written algebraically, they are in a simplified Sum-of-products form.

   (a) (10 pts) **DONE** circuit: Design a SoP-simplified circuit that takes $ABCD$ as the input value of the first roll, and returns 1 if the game is done (i.e., roll was 2,3,7,11 or 12), and returns 0 when the roll is something else (i.e., roll was 4,5,6,8,9,10). You don't need to draw the circuit, your solution may be written algebraically.

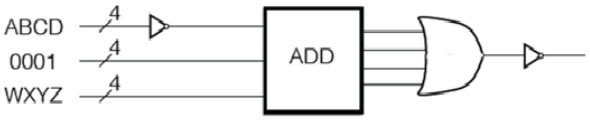
   (b) (10 pts) **NOT-NATURAL** circuit: Design a SoP-simplified circuit that takes $ABCD$ as the input value of a roll, and returns 0 for a natural (i.e., roll was 7,11), and 1 any other roll between 2 and 12. Again, you only need to provide the algebraic solution - no need to draw the circuit.

3. (20 pts) Let's now extend the design to addressing the remaining rolls. One thing we need to test for is whether a subsequent roll matches the first roll. Suppose the robot produces two 4-bit inputs, $WXYZ$ and $ABCD$, where $WXYZ$ represents the value of the first roll, and $ABCD$ represents the value of the current roll. We need to build a circuit, **FIRST-REPEAT**, which returns 1 when $WXYZ = ABCD$ (i.e., $W = A, X = B, Y = C, Z = D$), and 0 otherwise.

   (a) (5 pts) One way to test equality is to directly test if $W = A$ and $X = B$ and $Y = C$ and $Z = D$. Let **BitEqual** be a circuit that takes two 1-bit inputs, F and G, and has output H where $H = 1$ if $F = G$ and 0 otherwise. Design a **BitEqual** circuit. Your solution may be written as a circuit using basic gates (e.g., AND, OR, NOT, XOR) or in algebraic form.

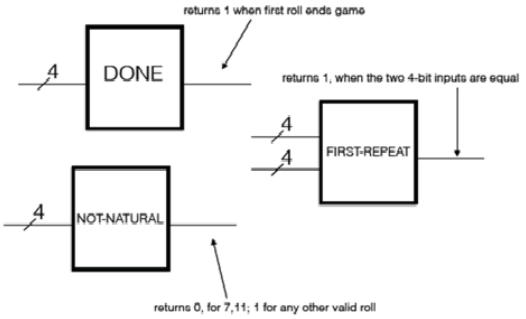
   (b) (5 pts) Using **BitEqual** circuits and additional basic gates (AND, OR, NOT, XOR), draw circuitry that takes $WXYZ$ and $ABCD$ as inputs and returns 1 when they both represent the same unsigned binary value.

Figure for parts c and d:



   (c) (5 pts) Above is an alternative circuit to test equality of $ABCD$ with $WXYZ$. Assume here (for the moment) that $ABCD$ and $WXYZ$ are signed 2's-complement binary values (i.e., ranging between -8 and 7). The "ADD" circuit adds three unsigned 2's-complement 4-bit values together: in this case, it adds $\bar{A}\bar{B}\bar{C}\bar{D}$, the 4-bit constant 0001, and $WXYZ$. The ADD circuit outputs the 4-bit representation of the sum, and these bits are OR'd together, then complemented. Describe in one or two sentences why this circuit only produces 1 when and only when $ABCD$ and $WXYZ$ are identical.

   (d) (5 pts) Suppose that $ABCD$ and $WXYZ$ are not in 2's complement form but are instead **unsigned**. Will the circuit still work correctly? Give a 1 or 2 sentence explanation.



4. (20 pts) Finally, let's build a circuit where the robot inputs the initial roll, the current roll, and a boolean indicator of whether the current roll is the first roll, and outputs whether the game is ended and if so, whether the gambler won.

   Using the **DONE**, **NOT-NATURAL**, **FIRST-REPEAT** circuits built previously and pictured above, possibly along with additional gates (AND,OR,NOT,XOR) and basic circuits (MUX, DECODER, ENABLER), build a circuit with 2 outputs, E and N.

   The circuit takes as input:

   - A boolean F that equals 1 if the current roll is the first roll, and 0 otherwise (if it's a subsequent roll).
   - The 4-bit value $WXYZ$ of the first roll
   - The 4-bit value $ABCD$ of the current roll.

   The output E should equal 1 if the current roll ends the game. The output N equals 1 if, when the game is ended, the gambler wins. If the game is not ended (i.e., $E = 0$, W's output is irrelevant).

   Hint: It may help to think of the behavior of these circuits as the following:

   - For E:
     - When $F = 1$: if **DONE** outputs 1 then $E = 1$, else $E = 0$.
     - When $F = 0$: if **NOT-NATURAL** outputs 0 or **FIRST-REPEAT** outputs 1, then $E = 1$, else $E = 0$.
   - For N:
     - When $F = 1$: if **DONE** = 1 and **NOT-NATURAL** = 0 then $N = 1$, else N can safely be set to 0.
     - When $F = 0$, if **FIRST-REPEAT** = 1 then then $N = 1$. Else N can safely be set to 0.

   If you would prefer to build the circuit that outputs E from the circuit that outputs N, you may do so.

1. (35 pts) Given two unsigned n-bit integers, $A = A_{n-1}A_{n-2}\cdots A_0$ and $B = B_{n-1}B_{n-2}\cdots B_0$, suppose you wish to do the subtraction $A - B$, returning both the n-bit unsigned result $R = R_{n-1}R_{n-2}\cdots R_0$ and an additional bit V indicating overflow.

   (a) (7 pts) For humans, it is easy to see whether $V = 1$, even before doing the subtraction. Give the simplest pseudocode[1] which, given A and B as inputs, computes V. [No boolean algebra or circuits needed.]

   (b) (7 pts) R can be computed by treating A and B as signed 2's-complement values, and performing subtraction (i.e., let $-B = \bar{B} + 1$ then apply the unsigned binary add algorithm to inputs A and $-B$ to generate R), and then re-interpreting R as an unsigned value. R will be correct modulo 2^n, but overflow needs to be determined. For the case where $n = 3$ and $A = 6$, $B = 3$, show that when using the process described above to compute $R = A - B$, neither the traditional unsigned or 2's-complement overflow detectors correctly identify overflow for this unsigned subtraction.

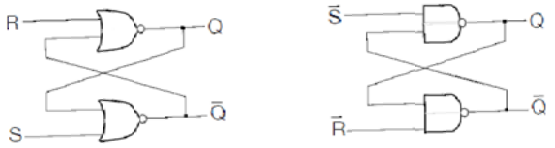
   (c) (7 pts) If $A_{n-1} = B_{n-1}$, then overflow occurs when and only when $R_{n-1} = 1$ (take this as a given). How about when $A_{n-1} \neq B_{n-1}$? Give a boolean expression for overflow (V) when it is given that $A_{n-1} \neq B_{n-1}$, and a brief (1 sentence) explanation that explains why this is the right expression. The boolean expression can utilize the following variables:

   - the input or output bits (i.e., any or all bits from A, B, and R),
   - carry bits from the adder that added A and $-B$, where C_i is the resulting carry of the ith most significant bits.

   (d) (7 pts) Use the observations from part 1c to build, using a 4-to-1 MUX, the circuit that computes V for any possible values of A_{n-1} and B_{n-1}. You may assume that the inputs to the circuit can be any of the following:

   - the input or output bits (i.e., any or all bits from A, B, and R),
   - constants,
   - carry bits from the adder that added A and $-B$,
   - Basic gates (AND, OR, NOT, XOR).

   (e) (7 pts) Redo part 1d, but use a 2-1 MUX instead of a 4-1 MUX (the inputs to the circuit can come from the same place as specified in part 1d).

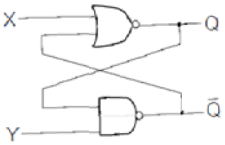


2. (20 pts) Above we have pictured an SR-Latch (on the left) and an $\bar{S}\bar{R}$-Latch (on the right). They can both be set, reset, or made to hold the current value. If it helps, remember these are the characteristic tables for these latches:

SR-latch			
R	S	Q	$\bar{Q}$
0	0	stay same	
0	1	1	0
1	0	0	1
1	1	don't use	

$\bar{S}\bar{R}$-latch			
$\bar{S}$	$\bar{R}$	Q	$\bar{Q}$
0	0	don't use	
0	1	1	0
1	0	0	1
1	1	stay same	

   Below is a circuit that is some weird hybrid of these 2 latches.



   Is this hybrid sequential circuit also a latch? Show what effect different inputs (values assigned to X and Y) have on what this circuit outputs. Don't forget to explain why it is or isn't the case that this circuit is a latch. Show all work!

3. (40 pts) This semester is the first time Professor Rubenstein has ever taught 2 sections of the same class, and he's been trying hard to keep them moving at the same pace. To help him maintain consistency, you are to build a sequential circuit that Professor Rubenstein can use to make sure that neither section falls more than 1 topic behind the other. Each lecture, he teaches some standard set of material and then has the option to either cover or skip one additional topic.

   - The circuit runs on a clock, where he teaches one section's lecture within each clock cycle[2], where the section taught in the current clock cycle differs from the section taught in the previous (and next) clock cycle.
   - At the start of lecture, Professor Rubenstein enters into the circuit a 1-bit input indicating whether he is willing to teach an additional topic near the end of lecture.
   - If he is not willing (input of 0), or the current section is ahead of the other section, the circuit outputs 0, telling him to not teach the additional topic.[3]
   - If he indicates he is willing (input of 1) and the current section is not ahead of the other section, the circuit outputs 1, telling him to teach the additional topic, which he does during the current lecture (current clock cycle).[4]

   The table below is an example with the two sections respectively labeled A and B and shows the number of topics by which A leads. The amount A leads by is shown in an alternate form in the next row as the lead of the current active section (same as the previous row's value when the active section is A, the negation when B). Note that each time the output is 1 when A is being taught, A's lead grows by 1, and each time the output is 1 when B is being taught, A's lead shrinks by 1.

Clock Cycle	0	1	2	3	4	5	6	7	8	9	10
Active Section	A	B	A	B	A	B	A	B	A	B	A
A ahead by	0	0	1	-1	1	-1	0	0	1	0	1
Active Section ahead by	0	0	-1	1	-1	1	0	0	-1	0	-1
Input	0	1	0	1	1	0	1	1	1	0	1
Output	0	1	0	0	1	0	1	1	1	0	0

   (a) (20 pts) Design a state machine that describes the above system (i.e., takes a 1-bit "random" input each clock cycle that indicates whether there is anything Prof. Rubenstein wants to teach, and outputs whether or not he should teach during the current lecture). Each state should include **a short text description that indicates what the state represents**, as well as the **binary labeling** you choose to use.

   HINT: why, in the example above, did we show the lead from two perspectives (one row from just A's perspective, the other from the active section's perspective)? One perspective might be more efficient in terms of the number of states than the other.

   (b) (20 pts) Provide **simplified** algebraic equations when using T flip-flops describing a sequential circuit implementing your state machine. The characteristic table for the T flip-flop is given below, with $T(t)$ being the input to the T flip-flop during clock cycle t.

$T(t)$	$Q(t+1)$
0	$Q(t)$
1	$\overline{Q(t)}$

[1] Giving new meaning to feeling the clock is running slow when sitting in his class
[2] He instead entertains students with great jokes and fascinating factoids during the remaining time.
[3] He still works great jokes and fascinating factoids into his remaining lecture.

1. (20 pts) If you've been following the news, President Trump just fired FBI Director Comey, and Comey found out about it from TV. To prevent this from happening again, a circuit will be built to institute a delay between the time when a decision is made to fire the FBI director and when the announcement is made. The circuit operates in 3 modes:

   - Green mode: the President is happy with the FBI director
   - Yellow mode: the President is has decided to fire the FBI director, but the announcement has not yet been made.
   - Red mode: the President has fired the FBI director (announcement made).

   The modes transit from green to yellow to red and then back to green (once the new hire has been made). The President provides a single input bit each clock cycle. When the input is 0, the circuit stays in its current mode. When the bit is a 1 in the tth clock cycle, then the following happens:

   - If in green mode at time t, the circuit transits to yellow mode for time $t+1$.
   - If in red mode at time t, the circuit transits to green mode for time $t+1$.
   - If in yellow mode at times $t-2, t-1$ and t, the system transits to red mode for time $t+1$. Otherwise, it remains in yellow mode (forcing a delay).

   Or, in other words, when the input is a 1, change immediately from green to yellow, or from red to green, but don't change from yellow to red unless at least three yellows have already occurred in a row.

   The circuit takes in a single input I and has a 2-bit output $C = C_1C_0$ which indicates the color: 00 for green, 01 for yellow, and 10 for red.

   An example:

time	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
I	0	0	1	1	0	0	0	1	0	0	0	1	1	1	1	1	1
C_1C_0	00	00	01	01	01	01	01	10	10	10	10	00	01	01	01	10	00

   (a) (8 pts) Draw a state machine that describes the circuit above. For each state in your state machine, briefly describe (in at most one sentence per state) what the "state" of the system is when the machine is in the given state. Label each state in a way that describes what the state represents intuitively, and assign to each state a unique binary sequence that will be used for the remainder of the problem.

   (b) (4 pts) Suppose at time 0 the light is red, and suppose the input is fixed at 1 for all cycles. Duplicate the table below in your blue book, and indicate what state your state machine is in for each clock cycle.

time	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
I	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
State Binary # (Not the output)																	

   (c) (8 pts) Give (simplified) algebraic equations for the circuit output C_1C_0, and inputs to the flip-flops, assuming a JK flip-flop is used to maintain the highest order bit of the state, and D flip-flops are used to maintain the remaining bits.

Recall the following excitation table for a JK flip-flops whose value at time t is $Q(t)$:

$J(t)$	$K(t)$	$Q(t+1)$
0	0	Q(t)
0	1	0
1	0	1
1	1	$\overline{Q(t)}$

1. (30 pts) You are to build a sequential circuit that implements the strategy of one player (who we call the *strategic* player) over multiple rounds. In each round (which takes a clock cycle), the circuit does three things:

   - It chooses a configuration (R,P,S) that the strategic player will play for that round
   - It reads in a 2-bit input XY that specifies its opponent's configuration (R,P,S) for that round
   - It outputs a 2-bit result R_1R_0 that specifies the strategic player's outcome (W,L,D) against the opponent.

   The digital representations of the configuration and outcome are specified as:

Opponent Configuration (XY)		Outcome (R_1R_0)	
00	Rock	0X	Draw
01	Paper	10	Lose
10	Scissors	11	Win

   (Note for the outcome, the high-order bit R_1 indicates whether someone won the round, and if so, the low order bit then indicates whether the strategic player won).

   The configuration played in the $t+1$st round (clock cycle) by the strategic player depends on what both players played in the tth round, and on the outcome (W,L,D).

   - If the strategic player lost in round t, the configuration in round $t+1$ matches the configuration in round t.
   - If the strategic player wins or draws in round t, the configuration in round $t+1$ is what would have lost against the opponent in round t.

   The following table enumerates the 9 scenarios that the 2 players could have played in the round t, and shows the outcome and what the strategic player will play in round $t+1$:

Strategic player round t configuration:	R	R	R	P	P	P	S	S	S
Opponent round t config:	R	P	S	R	P	S	R	P	S
Strategic player outcome [(W)in, (L)ose or (D)raw]:	D	L	W	W	D	L	L	W	D
Strategic Player's round $t+1$ config:	S	R	P	S	R	P	S	R	P

   (NOTE: The above is not an example of the game being played over a series of rounds, but shows the outcome and next move for all possible configurations of the current round).

   (a) (20 points) Draw a state machine that implements the sequential circuit described above.

   (b) (10 pts) Give simplified expressions for the sequential circuit using using JK flip-flops. For full credit, provide simplified expressions for both output bits and inputs to each of the inputs to the JK flip-flops.

   For your reference, the following table excitation table describes the behavior of a JK flip-flop:

$J(t)$	$K(t)$	$Q(t+1)$
0	0	$Q(t)$
0	1	0
1	0	1
1	1	$\overline{Q(t)}$

   NOTE: Part (a) is 2/3 of the points for this problem, and is a lot less grunt work. Be sure to look at other problems before sinking too much time into part (b).

1. (30 pts) A lighting system has 3 modes: OFF, DIM, and BRIGHT. In each clock cycle, the system is in one of these 3 modes, and takes a 2-bit input $I = I_1 I_0$, to determine its mode in the next clock cycle as follows:

   - If in OFF or BRIGHT mode during clock cycle $t-1$ and input $I(t) = 00$, the system stays in the same mode for clock cycle t.
   - If in OFF mode in clock cycle $t-1$ and input $I(t) = 10$ is received, the system switches to BRIGHT mode for clock cycle t. Similarly, if in BRIGHT mode for clock cycle $t-1$ and input $I(t) = 10$ is received, the system switches to OFF mode for clock cycle t.
   - If in OFF mode in clock cycle $t-1$ and input $I(t) = 11$ is received, the system switches to DIM mode for clock cycle t, and BRIGHT mode for clock cycle $t+1$. Similarly, if in BRIGHT mode in clock cycle $t-1$ and input $I(t) = 11$ is received, the system switches to DIM mode for clock cycle t and OFF mode for clock cycle $t+1$.

   Some observations:

   - the system cannot be in DIM mode for two consecutive clock cycles.
   - The input has no effect for the clock cycle when the system is in DIM mode (the next mode was determined by the input of the previous clock cycle).
   - It is appropriate to assume input 01 never occurs.

   You are to design a sequential circuit using JK flip-flops that generates a 2-bit output $O(t) = O_1(t)O_0(t)$ each clock cycle that specifies the mode of the lighting system for that clock cycle t as follows:

$O_1(t)$	$O_2(t)$	mode
0	0	Off
1	0	Dim
1	1	Bright

   An example follows, assuming the initial state is OFF.

clock cycle(t)	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
$I(t)$	00	00	10	00	10	00	00	11	00	00	11	11	11	00	10
$O(t)$	00	00	11	11	00	00	00	10	11	11	10	00	10	11	00

   Produce algebraic expressions that feed into the JK inputs of the flip-flops you use, as well as algebraic expressions for $O_1(t)$ and $O_0(t)$. For full credit, your solutions need to be simplified (and produce the correct behavior).

   Any partial work (e.g., state machine, excitation table, K-map, etc.) will be considered and can be used for partial credit.

   For your reference, the following table excitation table describes the behavior of a JK flip-flop:

$J(t)$	$K(t)$	$Q(t+1)$
0	0	$Q(t)$
0	1	0
1	0	1
1	1	$\overline{Q(t)}$