BOOLEAN ALGEBRA DESIGN PROJECTS

For each problem,

(a) *specify the input and output variables and the two states of each;*
(b) *construct the truth table listing all possibilities;*
(c) *write a Boolean expression;*
(d) *use a Karnaugh map simpliify the expression;*
(e) *draw the minimal circuit fo each output variable.*

1. A **hallway light** is controlled by three switches. If the light is off, clicking any one of the three switches turns it on. If the light is on, clicking any one of the three switches turns it off. A fourth input variable is needed to register the current state of the light (1 = on, 0 = off). The output variable should be 1 when the light is to be turned on and 0 when it is to be turned off.

2. A **car garage** has a front door and one window, each of which has a sensor to detect whether it is open. A third sensor detects whether it is dark outside. A security system for the garage follows this rule: the alarm rings if and only if the alarm switch is turned on and either the front door is not closed or it is dark and the side window is not closed.

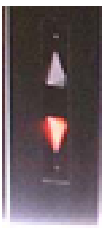
3. A **nuclear power plant** contains three danger sensors, *x*, *y*,and *z* that can shut down the plant. Sensor *z* is in such a sensitive location that it has a backup, *b*. The plant is to be shut down if two of the three sensors *x*, *y*,and *z* indicate danger. However, if sensor *z* is one of only two to register danger, *z*'s backup, *b*, must also register danger for the plant to be shut down.

*Exercises **4-6** deal with a **three-floor elevator**. For each proboem, there are five inputs. Three inputs are from the buttons for the floors (*f *= first floor button,* s *= second floor button,* t *= third floor button). The other two inputs are for the present floor indicator:* 01 *for first floor,* 10 *for second floor, and* 11 *for third floor. (Note that* 00 *is an impossible combination and should be omitted from each table.)*

4. Design the circuit that controls the door opening mechanism.

5. Repeat Exercise **6** on page **33** for the three-floor elevator but now make the output consist of two bits (*m* and *n*), where 00 means stay put, 01 means move *up*, and 10 means move *down*. Draw the simplest circuits for *m* and *n*. Note: You must draw a separate circuit for each of the two output bits.

6. An elevator has triangular lights to indicate whether it is going up or down. There are up and down lights on the outside of the elevator on each floor as well as an up light and a down light inside the elevator. Assume that all the up lights are connected to one circuit, and all the down lights are tied in to a second circuit. Design these two circuits. Include both outputs as part of one truth table but draw separate circuits for the up and down lights.



7. Design a **two-bit binary adder**. For example, 10 + 11 = 101. The truth table should contain four input columns for the bits of the numbers to be added and three output columns, one for each bit of the sum. Draw a separate circuit for each bit of the sum.

8. Encode four-bit data words in the **Hamming Code**. In this system, three parity-check bits are added to each four-bit input word to form a new word of seven bits. The seven bits are numbered from one to seven starting with the leftmost bit. The bit numbers 1, 2, and 4 are parity-check bits. The remaining bits are information bits. Each parity bit checks specific bit positions, and the parity bit is chosen so that the number of 1's in the checked positions is even. The bit positions checked by the parity bits are as follows.

Bit 1 checks bits 1, 3, 5, 7;
Bit 2 checks bits 2, 3, 6, 7;
Bit 4 checks bits 4, 5, 6, 7.

Example Suppose the input word is 1011. Then place these bits in positions 3, 5, 6, and 7 from left to right.

___	___	1	___	0	1	1
1	2	3	4	5	6	7

Parity bit 1 must be 0 because there are 2 1's in bits 3, 5, and 7.
Parity bit 2 must be 1 because there are 3 1's in bits 3, 6, and 7.
Parity bit 4 must be 0 because there are 2 1's in bits 5, 6, and 7.

So the Hamming code for 1011 is this: 0 1 1 0 0 1 1

Design a unit which, given the four bits of a data word, generates the three parity bits of the Hamming Code.

9. Some encoding system for binary transmissions employ a **generator polynomial** to create a coded value to add to a data word. For a 4-bit word, use the following polynomial to generate the fifth bit.

$$p = (w-1)^4 - (x-1)^3 + (y-1)^2 + z + 1$$

w, x, y, and z are the four bits (from left to right) of the data word. p is the value to be appended as the fifth bit. Since p must be 0 or 1, the calculation of the value of the polynomial is done *modulo 2*, which means that any value greater than 2 is replaced by its remainder when divided by 2.

Example Suppose the input word is 0011. Then evaluate the polynomial as follows.

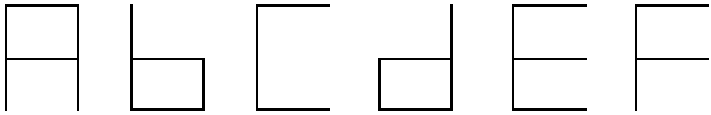
$$\begin{aligned} p &= (0-1)^4 - (0-1)^3 + (1-1)^2 + 1 + 1 \\ &= (-1)^4 - (-1)^3 + (0)^2 + 1 + 1 \\ &= 1 - (-1) + 0 + 1 + 1 \\ &= 1 + 1 + 0 + 1 + 1 \\ &= 4, \text{ which is converted to 0, its remainder when divided by 2.} \end{aligned}$$

So the 5-bit word that is transmitted is 0011**0**.

Similarly, for the input word 0100, the fifth bit is **1**.

Design a unit which, given the four bits of a data word, generates the fifth bit using the generator polynomial listed above.

10. Expand Exercise **7** on page **34** to allow for display of **hexadecimal digits** 0 through 9 and A through F.



Expand the truth table to *16* rows to allow 0000 through 1111 as input. Draw a separate circuit for each of the seven segments forming the display.

11. Design a **binary adder** that adds two three-bit numbers. The six inputs are the bits of the two numbers to be added. The four outputs are the bits of the sum. Draw a separate circuit for each bit of the sum.

Examples 110 + 101 = 1001 111 + 111 = 1110

12. Design a **binary adder** that adds three two-bit numbers. The six inputs are the bits of the three numbers to be added. The four outputs are the bits of the sum. Draw a separate circuit for each bit of the sum.

Examples 10 + 01 + 11 = 0110 11 + 11 + 11 = 1001

13. Design a **binary multiplier** that multiplies two two-bit numbers. The four inputs are the bits of the two numbers to be multiplied. The four outputs are the bits of the product. Draw a separate circuit for each bit of the product.

Examples 10 x 11 = 0110 11 x 11 = 1001

*In a **Gray code**, only a single bit changes when moving from one count to the next consecutive count. Although many Gray codes are possible, the most widely-used one is as follows.*

0	0	0	0	0		**8**	1	1	0	0
1	0	0	0	1		**9**	1	1	0	1
2	0	0	1	1		**10**	1	1	1	1
3	0	0	1	0		**11**	1	1	1	0
4	0	1	1	0		**12**	1	0	1	0
5	0	1	1	1		**13**	1	0	1	1
6	0	1	0	1		**14**	1	0	0	1
7	0	1	0	0		**15**	1	0	0	0

14. Design a 4-bit **binary-to-Gray** converter. That is, given the standard binary representation of a number from 0 through 15, output the Gray code for it using the table above.

Examples Binary 7, 0111, becomes 0100. Binary 14, 1110, converts to 1001.

15. Design a 4-bit **Gray-to-binary** converter. That is, given the Gray code of a number from 0 through 15 using the table above, output the standard binary code for it.

Examples Gray 6, 0101, becomes 0110. Gray 11, 1110, converts to 1011.

3-Input Variable Table/Map

Name ____

Problem ____________________

	y		y'	
x				
x'				
	z'	z		z'

1	1	1				
1	1	0				
1	0	1				
1	0	0				
0	1	1				
0	1	0				
0	0	1				
0	0	0				

	y		y'	
x				
x'				
	z'	z		z'

	y		y'	
x				
x'				
	z'	z		z'

1	1	1				
1	1	0				
1	0	1				
1	0	0				
0	1	1				
0	1	0				
0	0	1				
0	0	0				

	y		y'	
x				
x'				
	z'	z		z'

	y		y'	
x				
x'				
	z'	z		z'

1	1	1				
1	1	0				
1	0	1				
1	0	0				
0	1	1				
0	1	0				
0	0	1				
0	0	0				

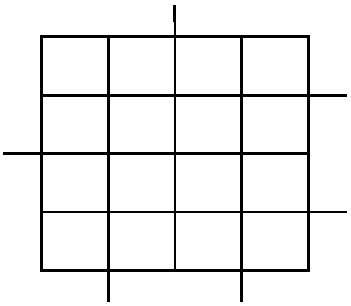
	y		y'	
x				
x'				
	z'	z		z'

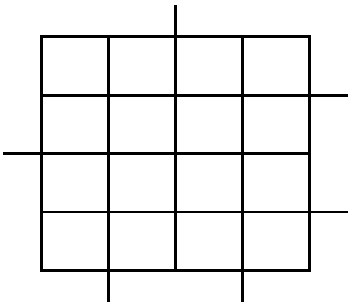
4-Input Variable Table/Map

Name _____

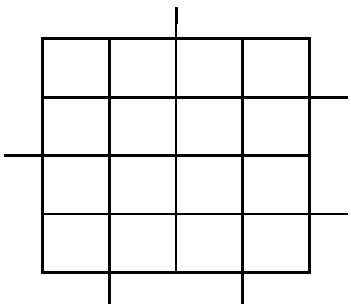
Problem _____

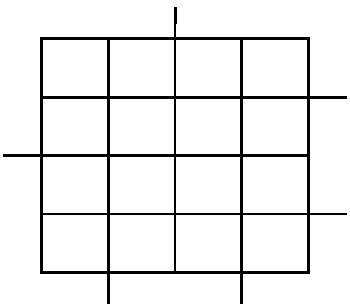
1	1	1	1				
1	1	1	0				
1	1	0	1				
1	1	0	0				
1	0	1	1				
1	0	1	0				
1	0	0	1				
1	0	0	0				
0	1	1	1				
0	1	1	0				
0	1	0	1				
0	1	0	0				
0	0	1	1				
0	0	1	0				
0	0	0	1				
0	0	0	0				





1	1	1	1				
1	1	1	0				
1	1	0	1				
1	1	0	0				
1	0	1	1				
1	0	1	0				
1	0	0	1				
1	0	0	0				
0	1	1	1				
0	1	1	0				
0	1	0	1				
0	1	0	0				
0	0	1	1				
0	0	1	0				
0	0	0	1				
0	0	0	0				





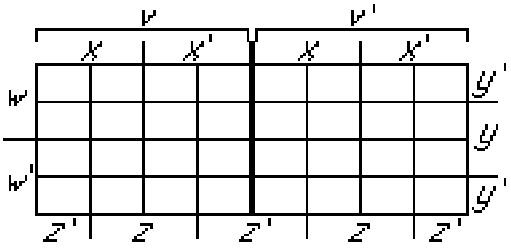
5-Input Variable Table/Map

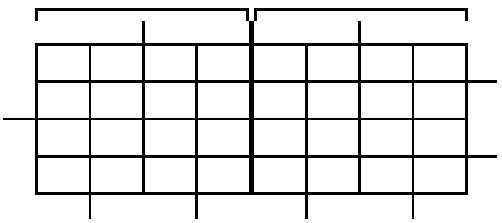
Name _____

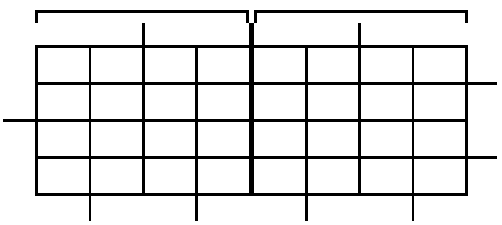
Problem _____

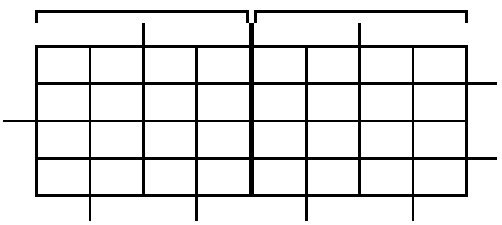
1	1	1	1	1			
1	1	1	1	0			
1	1	1	0	1			
1	1	1	0	0			
1	1	0	1	1			
1	1	0	1	0			
1	1	0	0	1			
1	1	0	0	0			
1	0	1	1	1			
1	0	1	1	0			
1	0	1	0	1			
1	0	1	0	0			
1	0	0	1	1			
1	0	0	1	0			
1	0	0	0	1			
1	0	0	0	0			
0	1	1	1	1			
0	1	1	1	0			
0	1	1	0	1			
0	1	1	0	0			
0	1	0	1	1			
0	1	0	1	0			
0	1	0	0	1			
0	1	0	0	0			
0	0	1	1	1			
0	0	1	1	0			
0	0	1	0	1			
0	0	1	0	0			
0	0	0	1	1			
0	0	0	1	0			
0	0	0	0	1			
0	0	0	0	0			

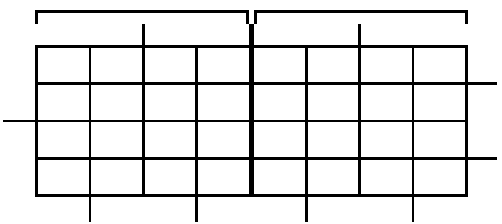
MODEL











6-Input Variable Table/Map

Name _____

Problem _____

1	1	1	1	1	1				
1	1	1	1	1	0				
1	1	1	1	0	1				
1	1	1	1	0	0				
1	1	1	0	1	1				
1	1	1	0	1	0				
1	1	1	0	0	1				
1	1	1	0	0	0				
1	1	0	1	1	1				
1	1	0	1	1	0				
1	1	0	1	0	1				
1	1	0	1	0	0				
1	1	0	0	1	1				
1	1	0	0	1	0				
1	1	0	0	0	1				
1	1	0	0	0	0				
1	0	1	1	1	1				
1	0	1	1	1	0				
1	0	1	1	0	1				
1	0	1	1	0	0				
1	0	1	0	1	1				
1	0	1	0	1	0				
1	0	1	0	0	1				
1	0	1	0	0	0				
1	0	0	1	1	1				
1	0	0	1	1	0				
1	0	0	1	0	1				
1	0	0	1	0	0				
1	0	0	0	1	1				
1	0	0	0	1	0				
1	0	0	0	0	1				
1	0	0	0	0	0				

0	1	1	1	1	1				
0	1	1	1	1	0				
0	1	1	1	0	1				
0	1	1	1	0	0				
0	1	1	0	1	1				
0	1	1	0	1	0				
0	1	1	0	0	1				
0	1	1	0	0	0				
0	1	0	1	1	1				
0	1	0	1	1	0				
0	1	0	1	0	1				
0	1	0	1	0	0				
0	1	0	0	1	1				
0	1	0	0	1	0				
0	1	0	0	0	1				
0	1	0	0	0	0				
0	0	1	1	1	1				
0	0	1	1	1	0				
0	0	1	1	0	1				
0	0	1	1	0	0				
0	0	1	0	1	1				
0	0	1	0	1	0				
0	0	1	0	0	1				
0	0	1	0	0	0				
0	0	0	1	1	1				
0	0	0	1	1	0				
0	0	0	1	0	1				
0	0	0	1	0	0				
0	0	0	0	1	1				
0	0	0	0	1	0				
0	0	0	0	0	1				
0	0	0	0	0	0				

(Karnaugh maps over)

6-Input Variable Table/Map

Name ______________________________________

Problem ___________________

WARNING: Do not cut across the heavier lines when forming groups of 2 or 4.