Complete the following truth tables:

a b	a + b
0 0	
0 1	
1 0	
1 1	

a b	b + a
0 0	
0 1	
1 0	
1 1	

a b	a * b
0 0	
0 1	
1 0	
1 1	

a b	b * a
0 0	
0 1	
1 0	
1 1	

Complete the following truth tables:

a b c	a + (b * c)
0 0 0	
0 0 1	
0 1 0	
0 1 1	
1 0 0	
1 0 1	
1 1 0	
1 1 1	

a b c	(a + b) * (a + c)
0 0 0	
0 0 1	
0 1 0	
0 1 1	
1 0 0	
1 0 1	
1 1 0	
1 1 1	

Complete the following truth tables:

a b c	a * (b + c)
0 0 0	
0 0 1	
0 1 0	
0 1 1	
1 0 0	
1 0 1	
1 1 0	
1 1 1	

a b c	(a * b) + (a * c)
0 0 0	
0 0 1	
0 1 0	
0 1 1	
1 0 0	
1 0 1	
1 1 0	
1 1 1	

Complete the following truth tables:

a	a + 0
0	
1	

a	a * 1
0	
1	

a	$a + \overline{a}$
0	
1	

a	$a * \overline{a}$
0	
1	

Boolean Sum, Product, and Complement operations:

$a + b * c \quad = \quad a + b * c$

$a * \overline{b} \quad = \quad a * \overline{b}$

Precedence:

Additional Laws

Idempotent Laws

a + a = a   a * a = a

Boundless Laws

a + 1 = a   a * 0 = 0

Absorption Laws

a + (a * b) = a   a * (a + b) = a

Associative Laws

(a + b) + c = a + (b + c)   (a * b) * c = a * (b * c)

Additional Notes

For circuits, we use $\vee$ to represent OR gates
use $\wedge$ to represent AND gates
use ' to represent NOT

We also use these for truth tables that represent circuits:

A	B	$A \vee B$
1	1	1
1	0	1
0	1	1
0	0	0

Karnaugh Maps

	y	y'
x		
x'		

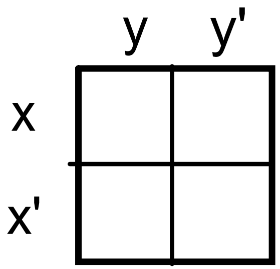
	y	y'
x		
x'		

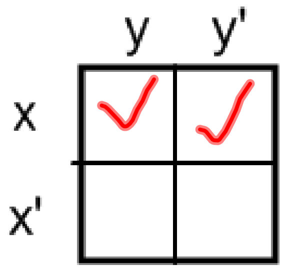
$E_1 = xy + xy'$

	y	y'
x		
x'		

$E_2 = xy + x'y + x'y'$

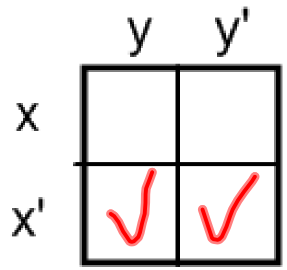
Karnaugh Maps





$E_1 = xy + xy'$

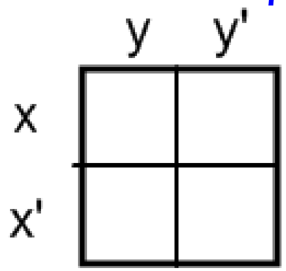
$E_1 = x$



$E_2 = xy + x'y + x'y'$

$E_1 = x' + y$

Adjacent squares isolated squares



$E_3 = xy + x'y'$

minimum disjunctive normal forms

Karnaugh Maps

Three variables x, y, z

	yz	yz'	y'z'	y'z
x				
x'				

Karnaugh Maps

Four variables x, y, z, t

t

	zt	zt'	z't'	z't
xy				
xy'				
x'y'				
x'y				

As one can initially see, there are four logic gates. To start creating an expression, first start with the gates located on the far left of the diagram, or in other words, the first gates that need to be calculated.

(A•B) ... (A•C)

One can then proceed to the next operations, which combine both gates to produce now a single output.

(A•B) + (A•C)

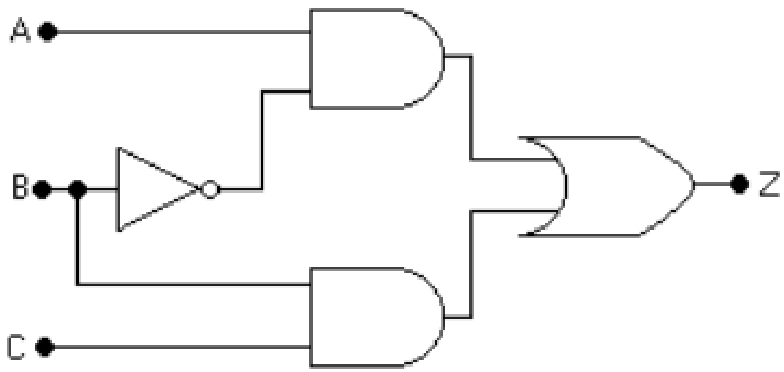
The last gate is an AND gate, which actually gets its input from the same input source as B. After we add that final gate we can conclude the expression with making it equal to the value of F.

(((A•B) + (A•C)) • B) = F

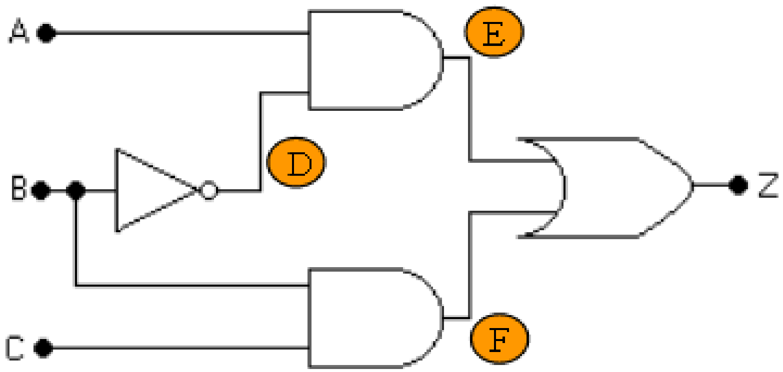
To finish it all off, one can remove some of the non-essential parentheses.

(A•B + A•C) • B = F

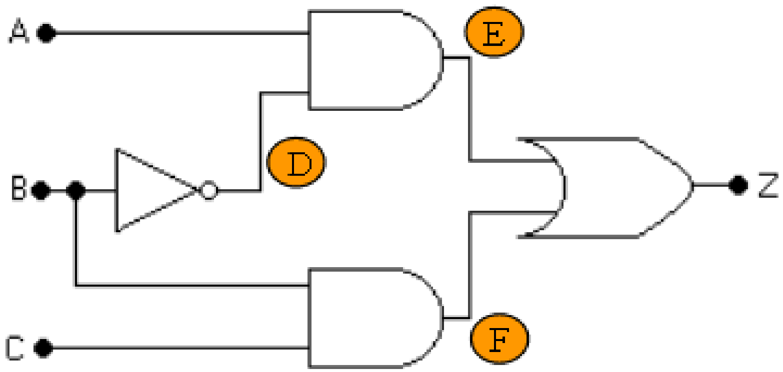
What Boolean expression is this circuit representing?



The first step is to sketch in some interim points on the output of each gate:



The first step is to sketch in some interim points on the output of each gate:



Then we can construct a trace table:

A	B	C	D (not B)	E (A.D)	F (B.C)	Z (E+F)
0	0	0	1	0	0	0
0	0	1	1	0	0	0
0	1	0	0	0	0	0
0	1	1	0	0	1	1
1	0	0	1	1	0	1
1	0	1	1	1	0	1
1	1	0	0	0	0	0
1	1	1	0	0	1	1

From this we can derive an expression for Z (remember that D, E & F are just our temporary labels, they **must not** appear in the final expression!):

$$Z = \overline{A}BC + A\overline{B}\,\overline{C} + A\overline{B}C + ABC$$

Simplify according to taste and we arrive at: $Z = A\overline{B} + BC$ or $Z = $ (A not B) OR (BC) - which amounts to the same thing.

Worked Past Paper Question:

November 2001, Paper 1, Question 14:

An elevator (lift) operates only between floors 2 and 5 of a building. The current floor number of the lift is stored in a three-bit register in binary code. If the value in the register corresponds to one of the floors that the lift serves, a circuit turns the lights in the elevator on, otherwise it turns them off.

(a) Construct the truth table for the operation of the lights (where an output signal of 1 will activate the lights) for all possible contents of the register. *[4 marks]*

(b) State the Boolean expression for the truth table in (a) using only operators from AND,OR and NOT. *[2 marks]*

(c) Simplify the Boolean expression given in (b). Your final answer can use any valid Boolean operators. *[4 marks]*

A bit of advice: **Do** learn to spot $Z = A\overline{B} + \overline{A}B$ as the XOR function. A quick check in the mark scheme revealed that had we left the answer in its "raw" form, a mark would have been deducted!

(c): Simplify by canceling out as many minterms as possible. It will come as no surprise to you that I am going to opt for a Karnaugh Map:

AB \ C	00	01	11	10
0		1		1
1		1		1

C is eliminated, leaving the expression $Z = A\overline{B} + \overline{A}B$. (*Hmm*... This is beginning to look familiar)

Both you and I know that this expression is an Exclusive OR (XOR) function, so to ensure maximum marks, we should write it as $Z = A \oplus B$ or A XOR B.

Et Voila! , Ten marks in the bag.

The Answer:

(a): We can construct the truth table quite easily from the information given in the question. When the code held in the 3 bit register indicates a floor between 2 and 5 inclusive (corresponding to the three inputs A, B & C), the lights will be on (output Z will be true or 1).

floor	A	B	C	Z
0	0	0	0	0
1	0	0	1	0
2	0	1	0	1
3	0	1	1	1
4	1	0	0	1
5	1	0	1	1
6	1	1	0	0
7	1	1	1	0

Examples:

	zt	zt'	z't'	z't
xy				
xy'				
x'y'				
x'y				

E_1

$y'z + xyz' + yz't'$

	zt	zt'	z't'	z't
xy				
xy'				
x'y'				
x'y				

E_2

$y' + xzt'$