1. A Boolean function is described by the following truth table:

a	b	c	z
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	0
1	1	1	0

(a) Write out a Boolean algebra expression for this function.

(b) Simplify the expression as much as possible.
Hint: In the final expression, each of the input variables should appear exactly once.

(c) Draw a realization of this Boolean function via logic gates.
Note: If you get stuck in part (b), you can draw the version from part (a).

(5+5+5)

(a) $z = (a' \wedge b' \wedge c') \vee (a' \wedge b' \wedge c) \vee (a \wedge b' \wedge c')$

(b) $z = (a' \wedge b' \wedge \underbrace{(c' \vee c)}_{=1}) \vee (a \wedge b' \wedge c')$ (distributivity) (complement)

$= (a' \wedge b') \vee (a \wedge b' \wedge c')$ (identity)

$= b' \wedge \underbrace{(a' \vee (a \wedge c'))}$ (distributivity)

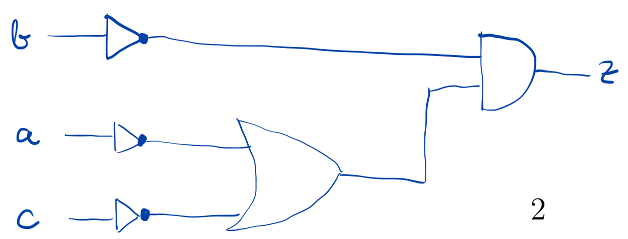
$= \underbrace{(a' \vee a)}_{=1} \wedge (a' \vee c')$ (distributivity) (complement)

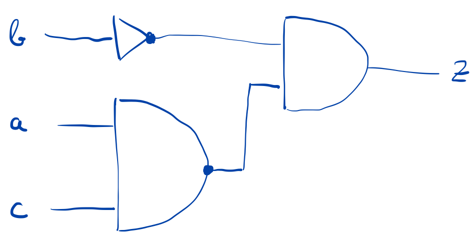
$= (a' \vee c')$ (identity)

$= b' \wedge (a' \vee c')$

Alternatively: $z = b' \wedge (a \wedge c)'$ (de Morgan)

(c)





2. Interpret the 8-bit string 01001001 in each of the following ways:

   (a) Unsigned integer

   (b) Signed two's complement integer

   (c) ASCII

   (d) An 8-bit floating point representation where the leading bit is the sign, the next three bits encode the exponent with a bias of 3, and the last four bits encode the normalized mantissa, with the leading 1 before the binary point implied, but not stored

   (e) Hamming-(8,4)-encoded bit string (correct or detect errors, if necessary)

(5+5+5+5+5)

(a) $2^0 + 2^3 + 2^6 = 1 + 8 + 64 = (73)_{10}$

(b) Leading digit is zero → positive, so it's again $(+73)_{10}$

(c) I (capital letter i, from ASCII-table)

(d) sign is positive,

exponent is $(100)_2 = (4)_{10}$ without bias, so $4-3=1$ with bias

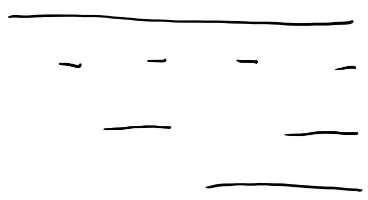
mantissa (significant) is $(\underset{\substack{\uparrow \\ \text{not stored}}}{1}.1001)_2 = 1 + \frac{1}{2} + \frac{1}{16} = 1\frac{9}{16}$

In total: $+1\frac{9}{16} \cdot 2^1 = 2\frac{9}{8} = 3\frac{1}{8} = 3.125$

(e)

0	1	2	3	4	5	6	7
p_0	p_1	p_2	d_0	p_4	d_1	d_2	d_3
0	1	0	0	1	0	0	1

Check:



$p_0 = 1$ ⇒ assume single-bit error

$\left.\begin{array}{l} p_1 = 0 \\ p_2 = 1 \\ p_4 = 0 \end{array}\right\}$ error in bit $(010)_2 = 2$

So p_2 is wrong,

message is 0001.

3. For each of the following cases, is it possible to construct a finite state transducer that reads the infinitely repeating input sequence 01010101 . . . and outputs the stated infinite output sequence? If possible, draw the finite state transducer, otherwise argue why it cannot be done.

   (a) ABABABAB. . .
   (b) AAAAABABABA. . .
   (c) ABBAAABBBBAAAAABBBBBB. . .

(5+5+5)

(a)

$$\begin{tikzcd}
{} \arrow[r] & S \arrow[loop above, "0:A"] \arrow[loop below, "1:B"]
\end{tikzcd}$$

(b)

$$\begin{tikzcd}
{} \arrow[r] & S1 \arrow[r, "0:A"] & S2 \arrow[r, "1:A"] & S3 \arrow[r, "0:A"] & S4 \arrow[r, "1:A"] & S5 \arrow[loop above, "0:A"] \arrow[loop below, "1:B"]
\end{tikzcd}$$

(c) No, as the machine has to keep track of the length of the sequence, which is increasing, so that this is not possible with a finite number of states.

4. Consider what is known as the "cigarette-smoker's problem": Three smokers are sitting around a table. The first has a supply of paper, the second has tobacco, and the third has matches. There is also an agent who has all three, but does not smoke. The agent randomly places two of the three ingredients on the table. The smoker who has the third missing ingredient could then roll a cigarette, light it, and smoke. When done, the smoker~~s~~ signals the agent to continue providing the next set of two items.

   (a) State the four necessary conditions for deadlock.

   (b) Explain why, without further provisions, each of the four necessary conditions for deadlock is present here, then describe how deadlock is possible.

(4+6)

(a) (i) Allocation of resources is exclusive

(ii) A process can hold a resource while waiting for another one

(iii) No preemption (temporary reallocation of resources)

(iv) Circular waiting

(b) (i) In the description of the problem, each resource is allocated to exactly one person.

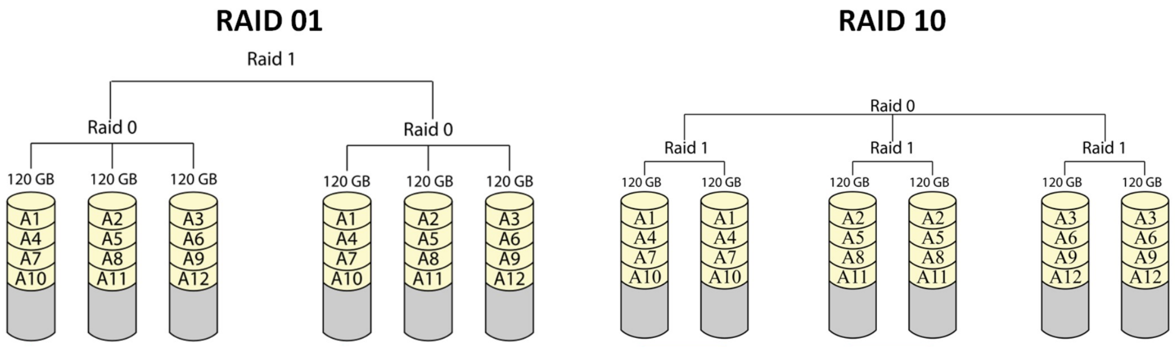
(ii) There is no provision that a smoker cannot grab one of the two resources on the table while another grabs the second, so that nobody can smoke.

(iii) Not included in description of problem

(iv) In the description it says "when done, the smoker signals the agent" which closes the dependency loop.

So deadlock occurs when the situation sketched under (ii) occurs. Then nobody smokes $\to$ nobody signals the agent $\to$ deadlock.

5. A RAID-01 configuration is a RAID-1 ("mirroring") arrangement of several RAID-0 ("striping") disk arrays. A RAID-10 configuration is similar, but in reversed order. Consider the 6-disk case as shown:



(Image from: `https://serverfault.com/a/848689`)

(a) What is the total storage capacity of the RAID-01 or RAID-10 array as shown?

(b) What is the read speed of RAID-01 or RAID-10 compared to the read speed of a single disk?

(c) What is the write speed of RAID-01 or RAID-10 compared to the write speed of a single disk?

(d) Which one is safer? Answer this question assuming, as is often the case in practical implementations, that an entire RAID-0 array is failing as soon a single disk in that array is failing.

(5+5+5+5)

(a) For both RAID-10 and RAID 01: Each disk is mirrored by another one, so capacity is half of the total raw capacity. Here: $C = 3 \cdot 120$ GB $= 360$ GB

(b) Reading can proceed on all disks in parallel, so it can be up to 6 times faster than a single disk. (Both cases)

(c) As each data block goes on two disks, writing is up to 3 times faster than writing a single disk. (Both cases)

(d) If one disk in RAID-01 fails, the entire RAID-0 it belongs to fails, so all redundancy is gone.

If one disk in RAID-10 fails, only one RAID-1 pair lacks redundancy, the other two pairs still provide mirroring.

So RAID-10 is safer than RAID-01 in a typical setup.