1. (a) Derive a Boolean algebra expression for the following binary truth table:

a	b	c	z
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

(b) Simplify your expression as much as possible.

(c) Draw a circuit diagram representing your (possibly simplified) expression.

(5+5+5)

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

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

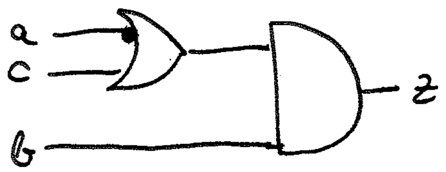
$= a' \wedge b$

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

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

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

(c)



2. Show your work when answering the following questions.

   (a) Convert $(60.625)_{10}$ into hexadecimal.

   (b) Convert $(10100.011)_2$ into decimal.

   (c) Add the 8-bit two's complement integers 0101 0001 and 0101 1101. Do you get overflow?

(5+5+5)

(a) $60 = 3 \cdot 16 + 12 \quad \Rightarrow (60)_{10} = (3C)_{16}$

$0.625 = \frac{5}{8} = \frac{10}{16} \quad \Rightarrow (0.625)_{10} = (0.A)_{16}$

$\Rightarrow (60.625)_{10} = (3C.A)_{16}$

(b) $(10100.011)_2 = 2^4 + 2^2 + 2^{-2} + 2^{-3}$

$= 16 + 4 + 0.25 + 0.125$

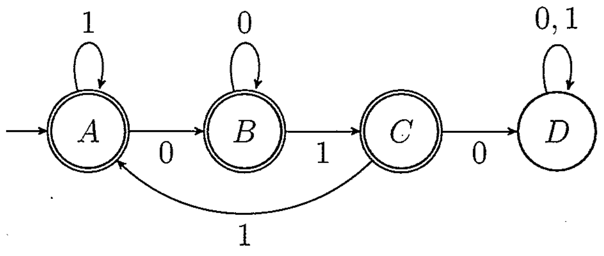
$= (20.375)_{10}$

(c) We can add two's complement integers just like unsigned integers:

$$\begin{array}{r} 01010001 \\ +\ 01011101 \\ \hline 10101110 \end{array}$$

Since both summands are non-negative (0 in leading position) but the result is negative (1 in leading position), overflow has occurred and the results is not a valid two's complement representation of the true sum.

3. Consider the following FSA:



(a) Is the input string 110100 accepted? List the sequence of states that the FSA takes when parsing this string.

(b) What strings does this FSA accept? Answer in words and state a corresponding regular expression.

(5+5)

(a) The sequence of states is

$$\textcircled{A} \xrightarrow{1} \textcircled{A} \xrightarrow{0} \textcircled{B} \xrightarrow{1} \textcircled{C} \xrightarrow{0} \textcircled{D} \xrightarrow{0} \textcircled{D}$$

D is not an accept state, so the string is rejected.

(b) Any string that contains the substring 010 will go to D, where it is trapped and will be rejected

All other strings do not reach D, so are accepted.

A regular expression is actually difficult here (in classical syntax), but this will do:

$$\underbrace{1*}_{\substack{\text{looping on A}\\ \text{any number of times}}} \underbrace{(0+11+)*}_{\text{going round the loop A}\to\text{B}\to\text{C}\to\text{A any number of times}} \underbrace{0*}_{\text{optionally going to B and possibly looping there}} \underbrace{1?}_{\substack{\text{optionally going B}\to\text{C at the end}\\ \text{(if at A, this does not hurt)}}}$$

4. Are the following statements true or false? Give a *brief* explanation for each. (True/False answers without explanation will not earn credit!)

   (a) In an operating system with dynamically managed virtual memory, every page of memory allocated to a process has a page table entry.

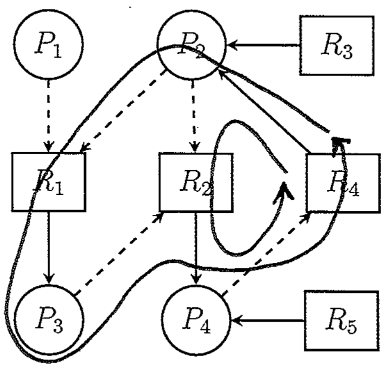
   (b) In an operating system with dynamically managed virtual memory, every page of memory addressable by a process has a page table entry.

   (c) In an operating system with dynamically managed virtual memory, physical memory must be at least as large as the combined size of all pages allocated to processes.

   (d) To avoid deadlock, it is necessary to design a system such that none of the following four conditions may occur: mutual exclusion, circular wait, no preemption, holding while waiting.

   (e) Deadlock implies Starvation.

   (f) Deadlock is possible in the following resource allocation graph. (Solid edges: Resource R_i is held by Process P_j. Dashed edges: Process P_k is waiting for resource R_ℓ.)



True, there is a circular wait pattern (actually two cycles!)

(2+2+2+2+2+2)

(a) True: The process can only address the physical page via the page table, so an entry must be there.

(b) False: It would be prohibitively expensive to create a page table entry for every page that could possibly be addressed.

(c) False: the whole point of memory management is to allow sharing of pages and also to possibly swap out physical pages to a backing store.

(d) False: Each of the conditions is a necessary condition, so it suffices to avoid one to avoid deadlock.

(e) True: Deadlocked processes do not make progress, so they are starved.

5. (a) Explain the good advice that *RAID is not a backup.*
(Mention at least two different aspects in your answer.)
(b) If it's not a backup, what is it good for then?

(5+3)

(a) Possible answers are:

- RAID does not protect against user error / program malfunction / malicious tampering
- RAID does not allow roll-back to a known-good state
- RAID does not protect against desasters (fire / earthquake / accidental destruction of hardware)
- RAID does not protect against random errors introduced during data transmission / processing

(b)
- Availability: A drive failure does not require shutdown of the service provided drives can be hot-swapped. Important since restoring large file systems from backup can take many hours / days.
- Allows creation of large-capacity arrays while maintaining low failure probabilities.

6. (a) The following message is received in Hamming-(8,4) encoding. Decode, correcting a single-bit error or reporting double-bit error as appropriate:

0101 1101

Use the bit-ordering convention employed in class.

(b) Take a correctly encoded Hamming-(7,4) message and flip any two bits. What state is the message in when you then try to decode?

(5+5)

(a)

$$\begin{array}{cccccccc}
0 & 1 & 2 & 3 & 4 & 5 & 6 & 7 \\
p_0 & p_1 & p_2 & d_1 & p_4 & d_2 & d_3 & d_4 \\
0 & 1 & 0 & 1 & 1 & 1 & 0 & 1 \\
\hline
 & \text{—} & & \text{—} & & \text{—} & & \text{—} \\
 & & \text{—} & \text{—} & & & \text{—} & \text{—} \\
 & & & & \text{—} & \text{—} & \text{—} & \text{—}
\end{array}$$

p_0 - check: odd, assume single-bit error

p_1 - check: 0

p_2 - check: 0

p_4 - check: 1

$\Rightarrow$ error at position $(100)_2 = 4$, so p_4 is wrong

$\Rightarrow$ data word is 1101

(b) A Hamming code without overall parity has two basic cases:

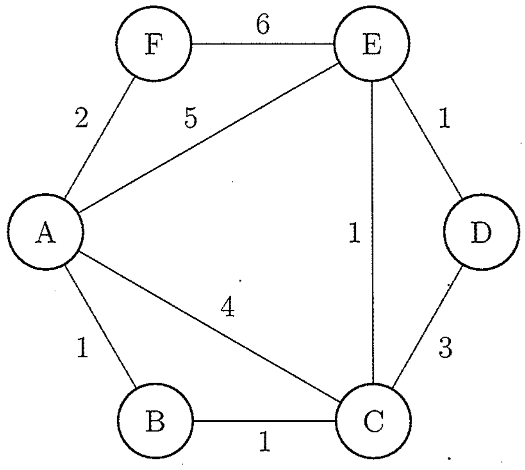
(i) no error detected

(ii) assumed single-bit error

Hamming-(7,4) is a code without overall parity.

If a clean code word has two bits flipped, it will read as a message with a single-bit error. There is no way to flip two bits and have all parity checks come out clean again.

7. Consider the following router network:



Use Dijkstra's algorithm to compute the shortest path from router A to every other router in the network. (10)

Step	M	D(A)	D(B)	D(C)	D(D)	D(E)	D(F)
0	~~A~~,B,C,D,E,F	0	∞	∞	∞	∞	∞
1	B,C,D,E,F	0	1 (B)	4(C)	∞	5(E)	2(F)
2	~~C~~,D,E,F	0	1	2(B)	∞	5	2
3	D,E,~~F~~	0	1	2	5(B)	3(B)	2
4	D,~~E~~	0	1	2	5	3	2
5	D	0	1	2	4(B)	3	2
Next Hop:		A	B	B	B	B	F