Mock Exam - Solutions

1. (a) $(0 \wedge 1)' \vee (1 \wedge 1)' = 0' \vee 1' = 1 \vee 0 = 1$

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

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

(for all a, b, c in the algebra)

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

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

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

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

2. (a) $(23)_{10} = (010111)_2$

$(28)_{10} = (011100)_2$ → Two's complement:

$$\begin{array}{r} 100011 \\ + \quad\quad 1 \\ \hline 100100 \end{array}$$

Thus, the computation reads

$$\begin{array}{r} 010111 \\ + 100100 \\ \hline 111011 \end{array} \quad (*)$$

Check (not required): Take two's complement of $(*)$:

$$\begin{array}{r} 000100 \\ + \quad\quad 1 \\ \hline 000101 \end{array}$$

But $(000101)_2 = (5)_{10}$, so $(*)$ represents $(-5)_{10}$ as a two's complement

(b) $(1011.101)_2 = 2^3 + 2^1 + 2^0 + 2^{-1} + 2^{-3}$

$= 8 + 2 + 1 + \frac{1}{2} + \frac{1}{8} \quad = 11.625$

3 (a) There is no problem when $x \approx 1$ as all terms are positive, so no loss of significant digits.

(b) When $x \approx \frac{1}{\sqrt{3}}$, $\sqrt{3}x \approx 1$ and $2x \approx \sqrt{x^2+1}$, so that there is loss of significant digits between these terms. Rewrite as follows:

$$\frac{2x - \sqrt{x^2+1}}{\sqrt{3}x - 1} = \frac{2x - \sqrt{x^2+1}}{\sqrt{3}x - 1} \quad \frac{2x + \sqrt{x^2+1}}{2x + \sqrt{x^2+1}}$$

$$= \frac{4x^2 - x^2 - 1}{(\sqrt{3}x-1)(2x+\sqrt{x^2+1})} = \frac{(\sqrt{3}x-1)(\sqrt{3}x+1)}{(\sqrt{3}x-1)(2x+\sqrt{x^2+1})}$$

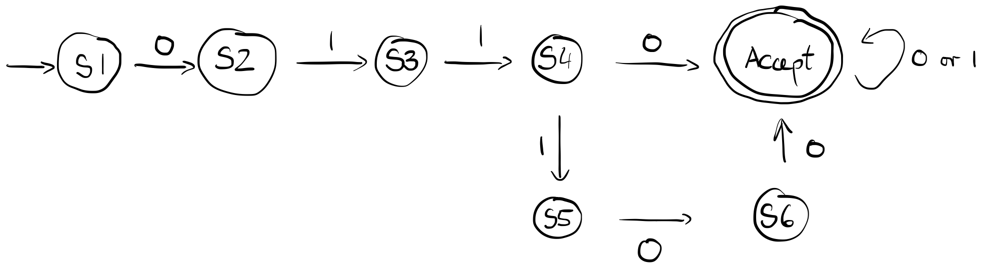
$$= \frac{3\sqrt{x}+1}{2x+\sqrt{x^2+1}}$$

When $x>0$, all terms are positive, so no major loss of significant digits!

4. (a) `.*aa.*` or `[A-Za-z]*aa[A-Za-z]*` (assuming that string contains only latin characters)

(b) The FSA will accept all binary strings that contain an odd number of 1s.

(c)



Note: Digits 0,…,7 are binary 0110xxx go directly from S4 → Accept

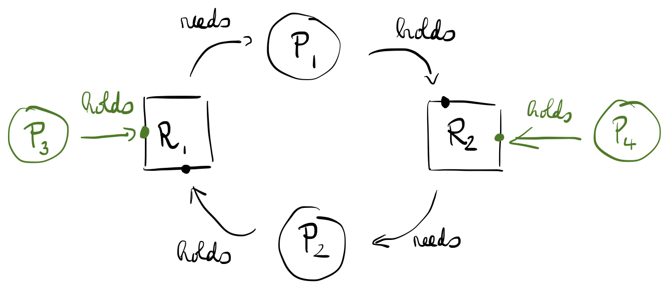
Digits 8,9 are binary 011000x go via S5, S6.

5. while (Z):
   decr (Z)   } set Z to zero

while (X):
   incr (Z)   } now Z is X, X is zero

while (Y)
   incr (Z)   } now Z is X+Y, Y is zero.

6. (a)



cyclic dependency graph → deadlock!

(b) Now P_3 and P_4 can make progress until they release the second copy of R_1 or R_2. Either will break the cycle involving P_1 and P_2.

→ no deadlock.

7. (a)

d_1 d_2 d_3 d_4

1 0 0 0 0 1 1

$p_1 = 0$

$p_2 = 0$

$p_4 = 0$

⇒ First codeword is O.K.

d_5 d_6 d_7 d_8

0 1 0 1 1 0 0

$p_1 = 1$

$p_2 = 1$

$p_4 = 0$

⇒ Bit error at position $(011)_2 = 3$

Here the error is in d_5, so data byte is 00111100

(b) All parities must be even, so codeword is 0000000.

(c) Suppose only one data bits is 1, the other zero.

Since each data bit is checked by at least two parity bits, at least three bits are 1 altogether.

Now suppose two data bits are 1.

Since there is a parity bit that checks one, but not the other, at least one parity bit is 1, so at least 3 bits altogether are 1.

Finally, suppose three data bits are 1.

Then the claim already holds looking at the data bits alone.