Tim Rumrich
Übung 10
Dienstag 10Uhr

10.1

a)
kleinste positive zahlen:
0 00000001 00000000000000000000000
=> $1.00000000000000000000000 * 2^{-126}$

0 00000001 00000000000000000000001
=> $1.00000000000000000000001 * 2^{-126}$

→ differenz = $0.00000000000000000000001 * 2^{-126}$

b)
die kleinste darstellbare positive zahl ist 0.
die kleinste darstellbare positive zahl größer als 0 ist dargestellt durch:
0 00000000 00000000000000000000001
das steht für $0.00000000000000000000001 * 2^{-126} = 2^{-149}$

c)

10.2

a)
61, 0A, 62, 0A
=
a
b

b)
61,0D,0A,62,0D,0A
wndows und unix systeme verwenden unterschiedliche systeme für den carriage return.
Bei windows wir vor das 0A noch ein 0D geschrieben vgl.: /n bei unix */r /*n bei windows

c)
Smiley = 1F642
unendlich zeichen = 221E
note = 1D15E
x = 0078
@ = 0040
Ö= 00D6

d)
1F642 = 11110000100000001001111110011001100000010
1D15E = 11110000100000001001110110000101100111110
22IE = 11100000100000101000100010011110
0078 = 01111000
0040 = 01000000
00D6 = 1100001110010110

10.3

00100010	01001001	01101110	01100110	01101111	01110010
0022	0049	006E	0066	006F	0072
„	I	n	f	o	r
01101101	01100001	01110100	01101001	01101011	00100000
006D	0061	0074	0069	006B	0020
m	a	t	i	k	
01101001	01101110	00100000	01100111	01100101	01110011
0069	006E	0020	0067	0065	0073
i	n		g	e	s
01100101	01101100	01101100	01110011	01100011	01101000
0065	006C	006C	0073	0063	0068
e	l	l	s	c	h
01100001	01100110	01110100	01101100	01101001	01100011
0061	0066	0074	006C	0069	0063
a	f	t	l	i	c
01101000	01100101	01110010	00100000	01010110	01100101
0068	0065	0072	0020	0056	0065
h	e	r		V	e
01110010	01100001	01101110	01110100	01110111	01101111
0072	0061	006E	0074	0077	006F
r	a	n	t	w	o
01110010	01110100	01110101	01101110	01100111	00100010
0072	0074	0075	006E	0067	0022
r	t	u	n	g	„

10.4

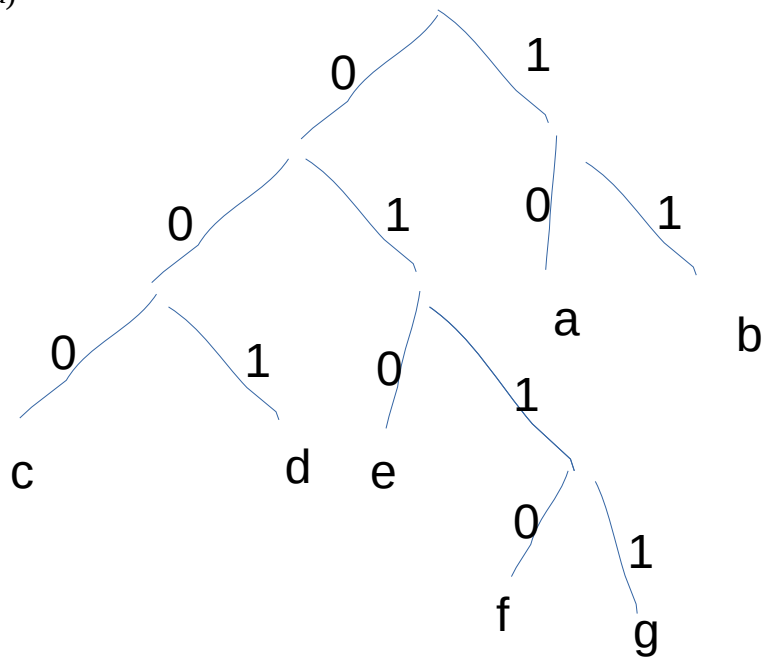
a)
44.1kHz = 44100*1/s
16Bit pro Tasten = 2Byte pro Tasten => 2Byte*44100/s = 88200Byte/s
1h = 3600s => 88200Byte*3600/s = 317520000Byte/Stunde = ~318MB/Stunde

b)
??????????

c)
??????????

10.5

a)



b)
Auf den ersten Blick ja, eventuell könnten geschickte Permutationen noch eine Kollision hervorrufen, aber es ist jetzt 21:50 und ich hab keine Lust mir das komplett durchzurechnen.

c)
001 0110 11 11 010 010 10 000 001 0111 11 0110
d f b b e e a c d g b f

d)
Formel aus Folie:
bei |A|=m braucht man für c:A → {0,1}^n brauchen minimal n=log2(m) Zeichen

<u>was man damit machen soll weiß ich nicht!</u>

Aber wenn man den Baum umbaut sodass alle worte gleich lang sind hat jedes wort die länge 3Bit. Es werden 3 mal f bzw g verwendet, welche ursprünglich 4 Bit gebraucht haben, man spart also zuerst 3Bit. Allerdings werden noch 4 mal a bzw b verwendet, welche ursprünglich 2 Bit hatten, wodurch man wieder 4 Bit verliert. Insgesamt wäre Blockcode also 1 Bit länger als die vorgeschlagene Codierung.

=> Lösung: 12*3Bit= 36Bit