Solutions to Assignment 1

Question 1

The answer will depend on your student ID.

(a) As an example, if the student ID is A00123456 the LS digit is 6 and the second LS digit is 5 and the waveform would have a frequency of $1000 + 6 \times 100 = 1600$ Hz and an amplitude of $1 + 5 = 6$ V.

(b) if the waveform is sampled at a rate of 3 kHz, the sample period is $\frac{1}{3000}$ s and time of the n'th sample is $t = \frac{n-1}{3000}$.

(c) we can compute the voltage quantization "bin" by subtracting the minimum voltage (-10 V), dividing by the quantization step size (312.5 mV) and truncating the result to an integer. The expression is thus[1]:

$$i = \lfloor \frac{v + 10}{0.3125} \rfloor$$

(d) The following C program:

```
#include <stdio.h>
#include <math.h>

main(){

   float t ;
   int n ;
   for ( n=0; n<10 ; n++ ) {
      t = n/3000.0 ;
      printf ( "%d %.4f %4.0f\n", n, t, floor(
  (10.0+6.0*sin(6.28*1600.0*t))/0.3125) ) ;
   }
}
```

produces in the following output:

```
0 0.0000   32
1 0.0003   28
2 0.0007   39
3 0.0010   20
4 0.0013   46
5 0.0017   15
6 0.0020   50
7 0.0023   12
8 0.0027   51
9 0.0030   13
```

The calculations can also be done with a spreadsheet. For example, using the formulas

	A	B	C	D
1	0	=A1/3000	=6*SIN(2*PI()*1600*B1)	=INT((C1+10)/0.3125)
2	=A1+1	=A2/3000	=6*SIN(2*PI()*1600*B2)	=INT((C2+10)/0.3125)

the results would look like:

n	t	v	i
0	0.0000	0	32
1	0.0003	-1.25	28
2	0.0007	2.44	39
3	0.0010	-3.53	20
4	0.0013	4.46	46
5	0.0017	-5.20	15
6	0.0020	5.71	50
7	0.0023	-5.97	12
8	0.0027	5.97	51
9	0.0030	-5.71	13

[1] If the quantized value were exactly +10 V the result would be 64 rather than 63 but we could convert this to 63 with no loss of accuracy.

Question 2

The answers will (should) be different for each student. For example,

(a) for the word "cake," the translation to Chinese (Simplified) is: 蛋糕

(b) the UTF-8 encoding is the 6 bytes: `e8 9b 8b e7 b3 95`

(c) Two characters are encoded using 6 bytes.

(d) We can look up each character on-line (for example, http://www.unicode.org/charts/unihan.html) or convert from UTF-8 using the encoding table from the Unicode standard:

The first character's UTF-8 encoding is `e8 9b 8b` so z=1000, y=01 1011, x=00 1011 and the code point in binary is 1000 0110 1100 1011 which is U+86CB.

The second character's UTF-8 encoding is e7 b3 95 so z=0111, y=11 0011, x=01 0101 and the code point in binary is 0111 1100 1101 0101 which is U+7CD5.

Question 3

There are various ways to obtain a solution.

The following program uses the numerical value of each character (all of which are ASCII characters and thus less than 128) as an index into an array. The values in this array keep track of the number of times each character appears in the text. One for-loop then calculates the number of characters and a second loop calculates the individual probabilities and the entropy.

```
#include <ctype.h>
#include <math.h>
#include <stdio.h>

char *s =
   "Consider the text of this question, not"
   "including the Hint below. Find the"
   "number of unique letters and their"
   "frequencies (number of times they"
   "occur).  Consider upper- and lower-case"
   "to be the same and ignore punctuation"
   "and spaces. How often does each letter"
   "occur? Using the relative frequencies of"
   "the letters as their probabilities, what"
   "is the entropy in bits/letter? Show how"
   "you obtained your result." ;

int count [128] ;

main()
{
   int i, nl=0, n=0 ;
   float p, h=0 ;
   while ( *s ) {
      count[tolower(*s)]++ ;
      s++ ;
   }
   for ( i=0 ; i<128 ; i++ )
      if ( islower(i) && count[i] ) {
         nl++ ;
         n += count[i] ;
      }
   printf ("found %d letters, %d unique\n", n, nl ) ;
   for ( i=0 ; i<128 ; i++ ) {
      if ( islower(i) && count[i] ) {
         p = (float)count[i]/n ;
         printf( "%c %3d %5.2f\n",
                 i, count[i], p ) ;
         h += -log(p)/log(2.0) * p ;
      }
   }
   printf ("entropy is %.2f bits/character\n", h ) ;
}
```

The output is as follows:

```
found 329 letters, 23 unique
a  14  0.04
b   8  0.02
c  13  0.04
d  10  0.03
e  49  0.15
f   8  0.02
g   3  0.01
h  17  0.05
i  25  0.08
l  10  0.03
m   4  0.01
n  25  0.08
o  25  0.08
p   6  0.02
q   4  0.01
r  22  0.07
s  21  0.06
t  36  0.11
u  17  0.05
v   1  0.00
w   6  0.02
x   1  0.00
y   4  0.01
entropy is 4.06 bits/character
```

Another method is to reformat the text so there is one lower-case alphanumeric character per line, import the text into a spreadsheet and use the `frequency()` spreadsheet function to compute the frequency of each letter. The `code()` function is used to convert characters into numbers for the `frequency()` function. The result is shown below:

a	97	a	97	14	0.0426	0.1938
a	97	b	98	8	0.0243	0.1304
a	97	c	99	13	0.0395	0.1842
a	97	d	100	10	0.0304	0.1532
a	97	e	101	49	0.1489	0.4092
a	97	f	102	8	0.0243	0.1304
a	97	g	103	3	0.0091	0.0618
a	97	h	104	17	0.0517	0.2209
a	97	i	105	25	0.0760	0.2825
a	97	j	106	0	0.0000	0.0000
a	97	k	107	0	0.0000	0.0000
a	97	l	108	10	0.0304	0.1532
a	97	m	109	4	0.0122	0.0773
a	97	n	110	25	0.0760	0.2825
b	98	o	111	25	0.0760	0.2825
b	98	p	112	6	0.0182	0.1054
b	98	q	113	4	0.0122	0.0773
b	98	r	114	22	0.0669	0.2610
b	98	s	115	21	0.0638	0.2534
b	98	t	116	36	0.1094	0.3493
b	98	u	117	17	0.0517	0.2209
b	98	v	118	1	0.0030	0.0254
c	99	w	119	6	0.0182	0.1054
c	99	x	120	1	0.0030	0.0254
c	99	y	121	4	0.0122	0.0773
c	99	z	122	0	0.0000	0.0000
c	99			329	1	4.0626
c	99					

The equations (not all visible) are:

	A	B	C	D	E	F	G
1	a	=CODE(A1)	a	=CODE(C1)	###	=E1/E$27	=-LOG(F1,2)*F1
2	a	=CODE(A2)	=CHAR(CODE(C1)+1)	=CODE(C2)	###	=E2/E$27	=-LOG(F2,2)*F2
3	a	=CODE(A3)	=CHAR(CODE(C2)+1)	=CODE(C3)	###	=E3/E$27	=-LOG(F3,2)*F3
4	a	=CODE(A4)	=CHAR(CODE(C3)+1)	=CODE(C4)	###	=E4/E$27	=-LOG(F4,2)*F4
5	a	=CODE(A5)	=CHAR(CODE(C4)+1)	=CODE(C5)	###	=E5/E$27	=-LOG(F5,2)*F5
6	a	=CODE(A6)	=CHAR(CODE(C5)+1)	=CODE(C6)	###	=E6/E$27	=-LOG(F6,2)*F6
7	a	=CODE(A7)	=CHAR(CODE(C6)+1)	=CODE(C7)	###	=E7/E$27	=-LOG(F7,2)*F7
8	a	=CODE(A8)	=CHAR(CODE(C7)+1)	=CODE(C8)	###	=E8/E$27	=-LOG(F8,2)*F8
9	a	=CODE(A9)	=CHAR(CODE(C8)+1)	=CODE(C9)	###	=E9/E$27	=-LOG(F9,2)*F9
10	a	=CODE(A10)	=CHAR(CODE(C9)+1)	=CODE(C10)	###	=E10/E$27	0.0000
11	a	=CODE(A11)	=CHAR(CODE(C10)+1)	=CODE(C11)	###	=E11/E$27	0.0000
12	a	=CODE(A12)	=CHAR(CODE(C11)+1)	=CODE(C12)	###	=E12/E$27	=-LOG(F12,2)*F12
13	a	=CODE(A13)	=CHAR(CODE(C12)+1)	=CODE(C13)	###	=E13/E$27	=-LOG(F13,2)*F13
14	a	=CODE(A14)	=CHAR(CODE(C13)+1)	=CODE(C14)	###	=E14/E$27	=-LOG(F14,2)*F14
15	b	=CODE(A15)	=CHAR(CODE(C14)+1)	=CODE(C15)	###	=E15/E$27	=-LOG(F15,2)*F15
16	b	=CODE(A16)	=CHAR(CODE(C15)+1)	=CODE(C16)	###	=E16/E$27	=-LOG(F16,2)*F16
17	b	=CODE(A17)	=CHAR(CODE(C16)+1)	=CODE(C17)	###	=E17/E$27	=-LOG(F17,2)*F17
18	b	=CODE(A18)	=CHAR(CODE(C17)+1)	=CODE(C18)	###	=E18/E$27	=-LOG(F18,2)*F18
19	b	=CODE(A19)	=CHAR(CODE(C18)+1)	=CODE(C19)	###	=E19/E$27	=-LOG(F19,2)*F19
20	b	=CODE(A20)	=CHAR(CODE(C19)+1)	=CODE(C20)	###	=E20/E$27	=-LOG(F20,2)*F20
21	b	=CODE(A21)	=CHAR(CODE(C20)+1)	=CODE(C21)	###	=E21/E$27	=-LOG(F21,2)*F21
22	b	=CODE(A22)	=CHAR(CODE(C21)+1)	=CODE(C22)	###	=E22/E$27	=-LOG(F22,2)*F22
23	c	=CODE(A23)	=CHAR(CODE(C22)+1)	=CODE(C23)	###	=E23/E$27	=-LOG(F23,2)*F23
24	c	=CODE(A24)	=CHAR(CODE(C23)+1)	=CODE(C24)	###	=E24/E$27	=-LOG(F24,2)*F24
25	c	=CODE(A25)	=CHAR(CODE(C24)+1)	=CODE(C25)	###	=E25/E$27	=-LOG(F25,2)*F25
26	c	=CODE(A26)	=CHAR(CODE(C25)+1)	=CODE(C26)	###	=E26/E$27	0.0000
27	c	=CODE(A27)			###	=SUM(F1:F26)	=SUM(G1:G26)

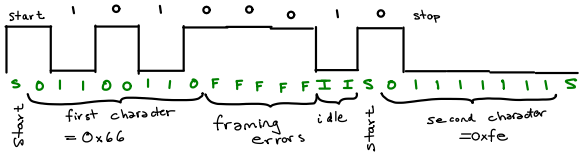
Question 4

(a) "Escaping" will require $2 \times \frac{10^7}{256} = 78125$ additional bytes. This file will be transmitted as $\lceil \frac{10^7+78125}{1023} \rceil = 9852$ frames (the last one will not be a full 1023 bytes). Each frame adds 8 bytes so the total number of bytes to be transmitted is $10^7 + 78125 + 9852 \times 8 = 10156941$ bytes. Since each byte requires 11 bits to transmit and the bit rate is 9600 bps, the time required is $\frac{10156941 \times 11}{9600} \approx$ 11638 seconds (about 3 hours).

(b) The throughput is the number of payload bits ($10^7 \times 8$) divided by the total time (11638 s) or about 6874 bps.

Question 5

The answers will vary, but the letter 'E' (0x45) is 0100 0101 in binary and will result in the the following waveform being transmitted (assuming 8 bits per character). The notes along the bottom of the waveform show how it would be interpreted by the receiver (also assumed to be set to 8 bits per character):



(a) The first character received begins with the second half of the transmitted start bit and ends after 8 bits. The hex value of the received character is 0x66 ('f'). The second character starts with the next low-to-high transition and the value received is 0xef (not a valid ASCII character or UTF-8 encoding by itself).

(b) A framing error will be detected for the first character because a high (0) level will appear where a stop bit is expected.

Question 6

(a) The parametric search feature of a vendor's site is a good way to find suitable ICs. For example, a search on digikey.com finds a number of suitable components:

(i) Silicon Labs CP2104 (the IC in the lab's DMM)
(ii) FTDI FT230X
(iii) Microchip MCP2200
(iv) Exar XR21B1411
(v) Cypress CY7C65213

(b) All of these devices feature configurable USB vendor and product IDs.

Question 7

An open-circuit transmission line acts as a capacitor. To find the required length we need to find the capacitance per meter.

Since $Z_0 = \sqrt{\frac{L}{C}}$, $L = CZ_0^2$.

For air dielectric $\varepsilon_r = 1$ so

$$v = c \times VF = \frac{c}{\sqrt{\varepsilon_r}} = \frac{1}{\sqrt{LC}} = \frac{1}{CZ_0}$$

Solving for C:

$$C = \frac{1}{cZ_0} = \frac{1}{3 \times 10^8 \times 90} \approx 37 \text{ pF/m}$$

To make a 10 pF capacitor we would thus need $\frac{10}{37} = 0.27$ m.

Question 8

At a frequency of 500 MHz, the wavelength is $\lambda = \frac{c}{f} = \frac{3 \times 10^8}{500 \times 10^6} = 0.6$ m. A gain of 16 dBi is a power ratio of $10^{16/10} \approx 40$. The effective aperture can be computed as:

$$A_e = \frac{G\lambda^2}{4\pi} = \frac{40 \times 0.6^2}{4\pi} = 1.14 \text{ m}^2$$