**Problems to be turned in**:

1. Convert the following binary numbers to decimal ***(6 points)***:
   1. 111011
      * 32+16+8+2+1 = 27 + 32 = 59
   2. 111110
      * 32+16+8+4+2+0 = 62
   3. 11011
2. Convert the following decimal numbers to binary ***(6 points)***:
   1. 67
      * 64(1)+32(0)+16(0)+8(0)+4(0)+2(1)+1(1) = 1000011
   2. 132
      * 128(1)+64(0)+32(0)+16(0)+8(0)+4(1)+2(0)+1(0) = 10000100
   3. 511
      * 256(1)+128(1)+64(1)+32(1)+16(1)+8(1)+4(1)+2(1)+1(1) = 111111111
3. Convert the following hex numbers to decimal ***(4 points)***:
   1. FAD
      * 15\*16\*16 + 10\*16 + 13 = 4013
   2. 489
4. Convert the following decimal numbers to hex ***(4 points)***:
   1. 51,966
      * 51966 = CB1C
   2. 16,383
      * 16383 = 3FFF
5. If b is a binary number, what number is represented by *b*0? That is, what number do you get when you place a 0 at the end of a binary number? What happens if you place two 0's at the end? Generalize your answer: what happens if you place *n* 0's at the end? ***(3 points)***
   1. Every time you put a 0 in front of a binary number, b or otherwise, the decimal value is multiplied by two.
   2. If you put two zeros, it is multiplied by four.
   3. If you put n 0’s at the end of a binary number b – b is multiplied by 2n.
6. Determine the binary representation of the following integers using 8 bits ***(6 points)***:
   1. -15
      * -15 = 1111 0001
   2. -64
      * -64 = 1100 0000
   3. -127
      * -127 = 1000 0001
7. ***(15 points)*** Suppose we decide to invent a code based on hand signals. Each signal will be transmitted by holding up exactly 2 fingers on the left hand. See example below. (You may assume the person doing the transmitting has 5 fingers on their left hand.) How many different signals are possible?

|  |  |
| --- | --- |
| **Example:** | |
|  | **Translation:** \* - - \* - |

In the example shown, the person doing the transmitting is holding up the "pinkie" and "forefinger" while concealing the thumb, index, and ring fingers. We can translate this easily to a code of "\*"s and "-"s by using a "\*" to indicate an extended finger and a "-" to indicate a concealed finger. The code corresponding to the example is "\*--\*-".

* 1. How many different signals are possible - that is, how many distinct arrangements of two "\*"s and three "-"s are possible? ***(2 points)***
     + 10 combinations
  2. Show each of the arrangements.***(10 points)***
     + : \*\*--- : \*-\*-- : \*--\*- : \*---\* : -\*\*-- :

: -\*-\*- : -\*--\* : --\*\*- : --\*-\* : ---\*\* :

* 1. Can you think of a logical order for the signals? That is, suppose we want one to stand for the letter "A", another to stand for "B", and so on for as many letters as possible. Try to arrange your signals in some logical fashion, and make up a chart depicting your "code."***(3 points)***

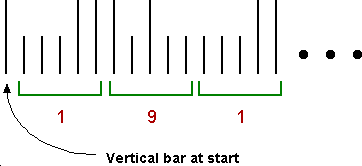
|  |  |  |  |
| --- | --- | --- | --- |
| A | \*\*--- | F | -\*-\*- |
| B | \*-\*-- | G | -\*--\* |
| C | \*--\*- | H | --\*\*- |
| D | \*---\* | I | --\*-\* |
| E | -\*\*-- | J | ---\*\* |

1. ***(12 points)*** In order to make its mail routing more efficient, the US Postal Service (USPS) prints a *bar code* on each envelope it receives for delivery. The bar code is easily read by the many machines the letter encounters between its source and destination. Postal bar codes have gone through several revisions over the years. For this exercise, we shall discuss the original POSTNET code. The bar code is printed by a machine that
   1. uses an optical scanner to read the address of the intended recipient
   2. tries to figure out the recipient's zip code
   3. computes a "check code" (explained below)
   4. prints a bar code on the envelope corresponding to the recipient's Zip Code with the "check code" appended to the end.

For example, examine the following bar code, which was printed on an envelope sent to the given zip code:

|  |  |
| --- | --- |
|  | 19104-2345 |

The bar code is composed of vertical lines, some long and some short. The bar code begins with a long vertical line and ends with a long vertical line. Between these two vertical lines, each digit of the zip code is represented by some arrangement of five vertical lines, 2 long and 3 short. Each digit has its own encoding - for instance, the sequence of three short lines followed by two long lines corresponds to the digit '1', and the sequence "long/short/long/short/short" corresponds to the digit '9', as shown in the figure below:



The purpose of the *check code* is to help machines detect (and possibly correct) mistakes they may make when reading the bar code. The check code is a digit that can be added to all the other digits so that their sum is a multiple of 10. In the example above, the check code is a single digit,'1'.

The USPS has introduced several upgrades to this coding system over the years in order to improve efficiency. For instance, you may find two extra digits between the zip code and the check code. If so, they should be the last two digits of the street address (it may be something else if the address is a "multi-unit" address such as an apartment or suite number). You may also find extra check digits at the end.

* 1. Using the example, determine the sequences of bars that correspond to the digits 0, 2, 3, 4, and 5.
     + 0 = long/long/short/short/short = ||…
     + 2 = short/short/long/short/long = ..|.|
     + 3 = short/long/short/short/long = .|..|
     + 4 = long/short/short/short/long = |…|
     + 5 = short/short/long/long/short = ..||.
  2. Decode the following POSTNET bar codes (generated by [Free Barcode Generator - Barcoding Inc.](http://www.barcoding.com/upc/) ):

https://www.cs.drexel.edu/~introcs/Fa13/assignments/HW4/postal/WhiteHouse_POSTNET.gif

* + - 20500

Redmond

|.|.. |..|. ||… .|.|. ..|.| .||..

* + - 98052-6

***Note***: You are supposed to notice a great similarity between POSTNET bar codes and the "hand signal code" you developed in the previous exercise.

1. ***(10 points)*** The ocarina is a wind instrument with a mouthpiece and finger holes for producing various musical tones (see [Clayzeness Ocarinas](http://www.clayz.com/index.html)(http://www.clayz.com/index.html) or [Wikipedia](http://en.wikipedia.org/wiki/Ocarina) (http://en.wikipedia.org/wiki/Ocarina) for more information). Musical notation for the four-hole ocarina is fairly concise: see [Primer Lesson and Scales](http://www.clayz.com/primer.html) (http://www.clayz.com/primer.html) for a description. Since there are four holes, each of which can be *covered* or *open*, each possible combination can be expressed with four bits, using **1** to signify *covered* and **0** to signify *open*. (We will ignore the "half-open" combinations in the Chromatic scale for this exercise.) Using the numbering system for the holes shown on the [Basic Scale page](http://www.paintandplayocarina.com/ocarina_song1.html)(http://www.paintandplayocarina.com/ocarina\_song1.html) , it becomes a simple matter to assign a four bit binary number (and therefore one hexadecimal digit) to signify each fingering position. For instance, see the chart below for one possible combination:

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Hole 1** | **Hole 2** | **Hole 3** | **Hole 4** | **Diagram** | **Binary Form** | **Hexadecimal Form** | **Decimal Form** |
| *open* | *open* | *covered* | *covered* | |  |  | | --- | --- | | **3** | **1** | | **4** | **2** | | **1100** | **C** | **12** |

1. It is worth noting that some of the fingering positions do not correspond to any note.
   1. Determine the binary, decimal and hexadecimal equivalents for each note in the Diatonic scale
      * do = 0000 = 0 = 0
      * re = 1000 = 8 = 8
      * me = 0100 = 4 = 4
      * fa = 1100 = C = 12
      * sol = 1010 = A = 10
      * la = 1110 = D = 13
      * ti = 0111 = 5 = 5
      * do = 1111 = F = F
   2. Translate the songs [*If I Only Had a Brain*](http://www.clayz.com/abrain.html) (http://www.clayz.com/abrain.html) and [*Zelda's Lullaby*](http://www.clayz.com/zsthemes.html) (http://www.clayz.com/zsthemes.html) into hexadecimal, using this system.
      * 1110 1011 1111 1110 1100 1010 1110 1100 1000 0100 1100 1000 0000 1000 0000 1000 0000 0000 0000 1000 1011 1110 1010 1100 0100 1000 0000 1111 1011 1110 1010 1100 0100 1100 1100 1100 1100 1100
        1. D B F D C A D C 8 | 4 C 8 0 8 0 8 0 0 | 0 8 B D A C 4 8 0 | F B E A C 4 C C C | C C
      * 0100 1010 1000 0000 1000 0100 1010 1000 0100 1010 1111 1111 1010 1100 0100 1000
        1. 4 C 8 0 8 4 C 8 4 A F F A C 4 8
2. **RTTTL *(10 points)***The [Ring Tone Text Transfer Language (RTTTL)](http://en.wikipedia.org/wiki/RTTTL) (http://en.wikipedia.org/wiki/RTTTL) was developed by Nokia to allow an easy scheme for representation of ringtones to be transferred to its cell phones. Free RTTTL players abound on the web; see [FRD Online Tools](http://www.free-ringtone-download.org/online-tools.html)(http://www.free-ringtone-download.org/online-tools.html) , for instance. A ringtone consists of a *name*, *settings* and *notes*. See [RTTTL Technical Specification](http://en.wikipedia.org/wiki/RTTTL#Technical_specification) (http://en.wikipedia.org/wiki/RTTTL#Technical\_specification) for a detailed explanation and an example.
3. ZeldasLullaby: d=2,o=5,b=120: 2e5,2f5,2d5,2c5,2d5,2e5,2f5,2d5,2e5,2g5 2c6,2c6,2g5,2f5,2e5,2d5

Translate *Zelda's Lullaby* from the previous question into RTTTL. Because the ocarina music did not contain note durations, you are permitted some freedom here. You may use the same duration for each note if you wish. You may also use any reasonable values in the *settings* portion. Finally, since RTTTL uses musical notes but the ocarina specification uses "do-re-mi-..." notation, you need to translate between these two notations. You may use the notes "C-D-E-F-G-A-B-C" for "do-re-me-fa-so-la-ti-do", respectively.

1. *Braille* is a code that represents writing with raised dots, allowing visually impaired persons to read by touch. The [National Braille Press' website](http://www.nbp.org/)(http://www.nbp.org/) contains a [visual interpretation of the Braille alphabet](http://www.nbp.org/ic/nbp/braille/alphacard.html) (http://www.nbp.org/ic/nbp/braille/alphacard.html) . Using this system, how many bits are needed to represent any Braille character? Do you need more bits to represent numbers than letters? Design a binary representation for each of the characters in the alphabet. ***(10 points)***
   1. You only need 6 bits to represent any Braille character, with the exception of numbers. In order to display a number, you need to put another 6 bits in front of it to declare a number.

00000 = A 00001 = B 00010 = C 00011 = D 00100 = E 00101 = F 00110 = G 00111 = H 01000 = I 01001 = J 01010 = K 01011 = L 01100 = M 01101 = N 01110 = O 01111 = P 10000 = Q 10001 = R 10010 = S 10011 = T 10100 = U 10101 = V 10110 = W 10111 = X 11000 = Y 11001 = Z

|  |
| --- |
| Click [here](https://www.cs.drexel.edu/~introcs/Fa13/assignments/HW4/index.html#Contents) to return to table of Contents. |