CS-150 Worksheet 2
Data Representation

This worksheet is about getting familiar with representation of different number types, including negative numbers, real numbers, and calculations on them. Show your working for all tasks.

☐ Task 2.1 – Convert to Two's Complement binary

i. Convert the following decimal numbers to 8-bit Two's Complement binary:

- 34
  34/2 = 17r**0**
  17/2 = 8r**1**
  8/2 = 4r**0**
  4/2 = 2r**0**
  2/2 = 1r**0**
  1/2 = 0r**1**
  = 100010 = 00100010 in 8-bit Two's Complement binary

- -50
  Calculate 50:
  50/2 = 25r**0**
  25/2 = 12r**1**
  12/2 = 6r**0**
  6/2 = 3r**0**
  3/2 = 1r**1**
  1/2 = 0r**1**
  = 110010 = 00110010
  ∴ -50 = *flip*(00110010)+1 = 11001110 in 8-bit Two's Complement binary

ii. Convert the following numbers from 8-bit Two's Complement binary to decimal:

- 10111011
  First bit tells us it is negative, ∴ do *flip*(10111011)+1 and convert the result.
  flip(10111011) + 1 = 01000101

  0×2^7
  $+1 \times 2^6$
  $+0 \times 2^5$
  $+0 \times 2^4$
  $+0 \times 2^3$
  $+1 \times 2^2$
  $+0 \times 2^1$
  $+1 \times 2^0$
  = 0+64+0+0+0+4+0+1 = 69

  then reintroduce the negative sign:
  10111011 in 8-bit Two's Complement binary = -69 in base 10

- 00100101
  0×2^7
  $+0 \times 2^6$
  $+1 \times 2^5$
  $+0 \times 2^4$
  $+0 \times 2^3$
  $+1 \times 2^2$
  $+0 \times 2^1$
  $+1 \times 2^0$
  = 0+0+32+0+0+4+0+1
  00100101 in 8-bit Two's Complement binary = 37 in base 10

☐ Task 2.2 – Two's Complement binary arithmetic

i. Perform the following additions with 8-bit Two's Complement binary representation:

- 00010101 + 00101110
  00010101
  + 00101110
  1111 (carry in)
  = 01000011

- 10010110 + 00010111
  10010110
  + 00010111
  1 11 (carry in)
  = 10101101

ii. Perform the following subtractions with 8-bit Two's Complement binary representation:

- 00110111 - 00001101
  (I will use A+(-B) method):
  = 00110111 + (*flip*(00001101) + 1)
  = 00110111 + 11110011

  00110111
  + 11110011
  1111 111 (carry in)
  = 100101010
  = 00101010 (due to 8-bit representation)

- 01011010 - 11101111
  (I will use A+(-B) method):
  = 01011010 + (*flip*(11101111) + 1)
  = 01011010 + 00010001

  01011010
  + 00010001
  1 (carry in)
  = 01101011

☐ Task 2.3 – Convert Real Numbers from base x to base y

i. Convert the following from decimal to binary

- 10.125
  Whole part:
  $10/2 = 5r\mathbf{0}$
  $5/2 = 2r\mathbf{1}$
  $2/2 = 1r\mathbf{0}$
  $1/2 = 0r\mathbf{1}$
  $= 1010$
  Fractional part:
  $0.125 \times 2 = \mathbf{0}.25$
  $0.25 \times 2 = \mathbf{0}.5$
  $0.5 \times 2 = \mathbf{1}.0$
  $= 001$
  $\therefore$ 10.125 in base 10 = 1010.001 in base 2

- 223.25
  Whole part:
  $223/2 = 111r\mathbf{1}$
  $111/2 = 55r\mathbf{1}$
  $55/2 = 27r\mathbf{1}$
  $27/2 = 13r\mathbf{1}$
  $13/2 = 6r\mathbf{1}$
  $6/2 = 3r\mathbf{0}$
  $3/2 = 1r\mathbf{1}$
  $1/2 = 0r\mathbf{1}$
  $= 11011111$
  Fractional part:
  $0.25 \times 2 = \mathbf{0}.5$
  $0.5 \times 2 = \mathbf{1}.0$
  $= 01$
  $\therefore$ 223.25 in base 10 = 11011111.01 in base 2

ii. Convert the following real numbers from binary to hexadecimal:

- 10010111100.0111
  note: 16 in base 2 is 10000
  Whole part:
  $10010111100/10000 = 1001011r\mathbf{1100} = r\mathbf{C}$
  $1001011/10000 = 100r\mathbf{1011} = r\mathbf{B}$
  $100/10000 = 0r\mathbf{100} = r\mathbf{4}$
  $= 4BC$
  Fractional part:
  $0.0111 \times 10000 = \mathbf{111}.0 = r\mathbf{7}$
  $= 7$
  $\therefore$ 10010111100.0111 in base 2 = 4BC.7 in base 16

- 1100.0010101
  note: 16 in base 2 is 10000
  Whole part:
  1100/10000 = 0r**1100** = r**C**
  = C
  Fractional part:
  0.0010101 × 10000 = **10**.101 = r**2**
  0.101 × 10000 = **1010**.0 = r**A**
  = 2A
  ∴ 1100.0010101 in base 2 = C.2A in base 16

☐ Task 2.4 – The `sign` × `mantissa` × `base`$^{\texttt{exp}}$ scheme

i. Convert the following decimal real numbers, identifying `sign`, `mantissa`, `base` and `exp`, **your representation should only use a mantissa of 5 digits**, e.g. 3.141592 becomes Sign: +1, Mantissa: 31415, Base: 10, Exponent: -4. **Note:** we drop the "92", and rounding does not occur as we haven't defined as such in this representation scheme.
   - 23.451
     Sign: +1, Mantissa: 23451, Base: 10, Exponent: -3
   - 0.123141
     Sign: +1, Mantissa: 12314, Base: 10, Exponent: -5

ii. Convert each of the following to their real number form, in decimal.
   - Sign: -1, Mantissa: 57231, Base: 10, Exponent: 5
     -5723100000
   - Sign: +1, Mantissa: 13123, Base: 10, Exponent: -7
     0.0013123

☐ Task 2.5 – Scientific Notation

Convert the following decimal real numbers into Scientific Notation, **however this time we can only store 5 significant digits**. For example: 111029 would be 1.1103E5. Note that Scientific Notation **does** define what happens with regards to rounding.

- 5240.82
  5.2408E3 (rounding does not occur)
- 249236.23
  2.4924E5 (rounding occurs)
- 0.0014210
  1.4210E-3 (non-leading 0 is still significant)

☐ Task 2.6 – Keyword Encoding

i. Apply Keyword Encoding to the following nursery rhyme:

> Three blind mice. Three blind mice. See how they run. See how they run. They all ran after the farmer's wife, Who cut off their tails with a carving knife, Did you ever see such a sight in your life, As three blind mice?

Common keywords may vary, but we could choose (for example) the mapping (note the case sensitivity):

Keyword	Encoding
Three	@
blind	#
mice	*
the	+

> @ # *. @ # *. See how +y run. See how +y run. They all ran after + farmer's wife, Who cut off +ir tails with a carving knife, Did you ever see such a sight in your life, As three # *?

We could also choose the mapping (note the case sensitivity):

Keyword	Encoding
hree blind mice	@
the	+

> T@. T@. See how +y run. See how +y run. They all ran after + farmer's wife, Who cut off +ir tails with a carving knife, Did you ever see such a sight in your life, As t@?

ii. Calculate the compression ratio of the new compressed message. Original message length: 220, Encoded length: 181. Ratio: 181/220 = 0.823 (to 3 decimal places)

☐ Task 2.7 – Run-Length Encoding

i. Apply Run-Length encoding to the following:

- AAAAAAAAAAaaaAAAABBCCCCDDDdAAAAaEEEEEE

  *A10aaa*A4BB*C4DDDd*A4a*E6

- 10111011011100000001010111111101000001111001000001

  101110110111 *07 1010 *16 01 *05 *14 001 *05 1, obviously here there is then the issue of how you represent the components of the encoding. The flag character identifies the start of the encoding, and the single character that follows is the one to repeat, but the potential for a multi-digit number of repetitions may be ambiguous. In this example I've used spaces to reduce ambiguity, but consider the follow showing the same information by using square brackets: 101110110111*0[7]1010*1[6]01*0[5]*1[4]001*0[5]1. When counting the message length, don't include the spaces or brackets used to reduce ambiguity (they aren't actually part of the message here).

ii. Calculate the compression ratio of the new compressed messages above.

Original message length: 49, Encoded length: 37, Ratio = 37/49 = 0.755

☐ Challenge Task

Construct a Huffman Tree and encode the following message:

- the cat in the hat sat on the mat

Calculate the compression ratio of the new compressed message above.

☐ Challenge Task

Write a program, in either Java or Python, which takes in a decimal floating point number and converts it to a fixed length `sign` $\times$ `mantissa` $\times$ `base`$^{\texttt{exp}}$ representation. Print out the different components of this representation. i.e:

```
Prompts and inputs:
    Enter floating point decimal number: 3.14159265359
    Enter length of mantissa: 5

Outputs:
    Sign: Positive
    Mantissa: 31415
    Base: 10
    Exponent: -4
```