Problem 1. **Measuring information**

originally $\log_2 8$ bits
after information $\log_2 3$ bits
$\rightarrow \log_2 8 - \log_2 3 = \log_2 8/3$

A. Someone picks a name out of a hat known to contain the names of 5 women and 3 men, and tells you a man has been selected. How much information have they given you about the selection?

$\log_2 52$, $\log_2 48$, 0

B. You're given a standard deck of 52 playing cards that you start to turn face up, card by card. So far as you know, they're in completely random order. How many new bits of information do you get when the first card is flipped over? The fifth card? The last card?

C. X is an unknown N-bit binary number (N > 3). You are told that the first three bits of X are 011. How many bits of information about X have you been given?

→ 3

number of different bits

D. X is an unknown 8-bit binary number. You are given another 8-bit binary number, Y, and told that the Hamming distance between X and Y is one. How many bits of information about X have you been given?

originally 8 bits
after information $\log_2 8 = 3$ bits (Y can be 8 choices, each hav 1 bit diff fr X)
$\rightarrow 8 - 3 = 5$ bits.

Problem 2. **Measuring information**

After spending the afternoon in the dentist's chair, Ben Bitdiddle has invented a new language called DDS made up entirely of vowels (the only sounds he could make with someone's hand in his mouth). The DDS alphabet consists of the five letters "A", "E", "I", "O", and "U" which occur in messages with the following probabilities:

Letter	Probability of occurrence
A	p(A) = 0.15
E	p(E) = 0.4
I	p(I) = 0.15
O	p(O) = 0.15
U	p(U) = 0.15

If you are told that the first letter of a message is "A", give an expression for the number of bits of information have you received.

$\log_2 \frac{1}{P(A)}$

Problem 3. **Modular arithmetic and 2's complement representation**

Most computers choose a particular word length (measured in bits) for representing integers and provide hardware that performs various arithmetic operations on word-size operands. The current generation of processors have word lengths of 32 bits; restricting the size of the operands and the result to a single word means that the arithmetic operations are actually performing arithmetic modulo 2^{32}.

Almost all computers use a 2's complement representation for integers since the 2's complement addition operation is the same for both positive and negative numbers. In 2's complement notation, one negates a number by forming the 1's complement (i.e., for each bit, changing a 0 to 1 and vice versa) representation of the number and then adding 1. By convention, we write 2's complement integers with the most-significant bit (MSB) on the left and the least-significant bit (LSB) on the right. Also by convention, if the MSB is 1, the number is negative; otherwise it's non-negative.

A. How many different values can be encoded in a 32-bit word? 2^{32}

B. Please use a 32-bit 2's complement representation to answer the following questions. What are the representations for

zero 00000000000000000000000000000000
the most positive integer that can be represented 01111111111111111111111111111111
the most negative integer that can be represented 10000000000000000000000000000000

What are the decimal values for the most positive and most negative integers?

C. Since writing a string of 32 bits gets tedious, it's often convenient to use hexadecimal notation where a single digit in the range 0-9 or A-F is used to represent groups of 4 bits using the following encoding:

hex	bits	hex	bits	hex	bits	hex	bits
0	0000	4	0100	8	1000	C	1100
1	0001	5	0101	9	1001	D	1101
2	0010	6	0110	A	1010	E	1110
3	0011	7	0111	B	1011	F	1111

$32768 = 2^{15}$
00001000 (HEX)
inverse ↓
FFFF7FFF
+1 ↓
FFFF8000

00000025

Give the 8-digit hexadecimal equivalent of the following decimal and binary numbers: 37_{10}, -32768_{10}, $11011110101011011011111011101111_2$. → DEADBEEF

D. Calculate the following using 6-bit 2's complement arithmetic (which is just a fancy way of saying to do ordinary addition in base 2 keeping only 6 bits of your answer). Show your work using binary (base 2) notation. Remember that subtraction can be performed by negating the second operand and then adding it to the first operand.

010010 ← 18
101110 ← -18

011011
+ 111010
1010101

13 + 10 → 001101 +) 001010 = 010111
15 − 18 → 001111 +) 101110 = 111101
27 − 6
−6 − 15
21 + (−21)
31 + 12

6 ↘ 000110
−6 ↘ 111010

Explain what happened in the last addition and in what sense your answer is "right".

E. At first blush "Complement and add 1" doesn't seem to an obvious way to negate a two's complement number. By manipulating the expression A+(-A)=0, show that "complement and add 1" does produce the correct representation for the negative of a two's complement number. Hint: express 0 as (-1+1) and rearrange terms to get -A on one side and XXX+1 on the other and then think about how the expression XXX is related to A using only logical operations (AND, OR, NOT).

$(-A) = (-A-1) + 1$

↳ $(-1) - A$

↳ 1111111111
−) $a_n \cdots a_1 a_0$
$r_n \cdots r_1 r_0$

if $a_i = 0 \rightarrow r_i = 1$ if $a_i = 1 \rightarrow r_i = 0$