Problem Set 1.

1.

A. (5 points) [- Will accept different values for Face Card (definition based) -]

- Uniquely encode all 52 cards: $\log_2(52) \approx 5.7$ bits

- Told a card's suit ($M = 13$): $\log_2(52/13) = 2$ bits of information

- Same card is a face card. Note, will accept 3 or 4 (if include Ace):
  $\log_2(13/3) \approx 2.115$ bits $\quad \log_2(13/4) \approx 1.7$ bits

B. (5 points) [- Will accept different values as well -]

- Total Bits of information in a single deck: $= \frac{5.7 \text{ bits}}{\text{card}} \times 52$ cards
  $= 296.4$ bits

- Remove cards 2-6 inclusive (all suits) $= 5 \times 4 = 20$ cards $\quad 52 - 20 = 32$
  Total Bits of information in smaller deck: $\log_2(32) = 5 \times 32$
  $= 160.$ bits

- Encode the card's suit (4 suits): $\log_2(4) = 2 \times 32$ cards $= 64$ bits

- Encode the card's face value (7-A = 8 cards): $\log_2(8) = 3 \times 32$ cards
  $= 96$ bits

C. (2 pts.)

0100111110010 where $A = 0$, $B = 10$, $C = 110$, $D = 111$

$$\underset{A}{0}\;\underset{B}{\underbrace{10}}\;\underset{A}{0}\;\underset{D}{\underbrace{111}}\;\underset{C}{\underbrace{110}}\;\underset{A}{0}\;\underset{B}{\underbrace{10}}$$

A B A D C A B

D. (5 pts.) [Other methods may be acceptable]

Simple Methods:

- Expected Length:

1000 choices

- $0.5 = A$ = 500 choices × 1 bit = 500 bits
- $0.3 = B$ = 300 choices × 2 bits = 600 bits
- $0.1 = C$ = 100 choices × 3 bits = 300 bits
- $0.1 = D$ = 100 choices × 3 bits =+ 300 bits

1700 bits

- Worst Case Length:

Each choice would result in worst length possible, 3 (C or D)

1000 choices × 3 bits = 3000 bits.

How do they compare with $1000 * \log_2(4/1) = 2000$?

Expected length results in less bits, while worst case results in 50% more bits.

E. (3 pts.)

Crooked Coin: $p_{tail} = 0.345$ / $p_{head} = 0.625$

- Using entropy function given, average bits of information =
$-(0.375\log_2(0.375) + 0.625\log_2(0.625)) \approx .954$ bits.

- Using fair coin ($p_{tail}, p_{heads} = 0.5$), average bits of information =
$-(0.5\log_2(0.5)\cancel{)} + 0.5\log_2(0.5)) = 1$ bit.
- Receive less information from a crooked coin than a fair coin.

F. (20 pts. total)

(2 pts.)

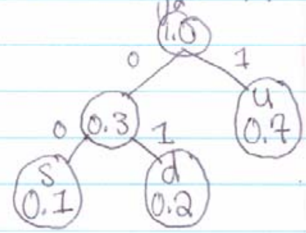
- Probabilities to maximize entropy: u, d, s all 1/3 each.
- Probabilities to minimize entropy: u = 1; d, s = 0.

(3 pts.)

$p(u) = 0.7$, $p(d) = 0.2$, $p(s) = 0.1$

Entropy $= -(0.7\log_2(0.7) + 0.2\log_2(0.2) + 0.1\log_2(0.1))$

$= 1.1568$ bits



(5 pts.)

Close to achieving bits in practice:

s = 00

d = 01

u = 1

Encoding Efficiency: $0.1(2) + 0.2(2) + 0.7(1) = 1.3$ bits

(2 pts.)

Unique code for 2-character combinations:

$p(uu) = 0.49$	$p(du) = 0.14$	$p(su) = 0.07$
$p(ud) = 0.14$	$p(dd) = 0.04$	$p(sd) = 0.02$
$p(us) = 0.07$	$p(ds) = 0.02$	$p(ss) = 0.01$

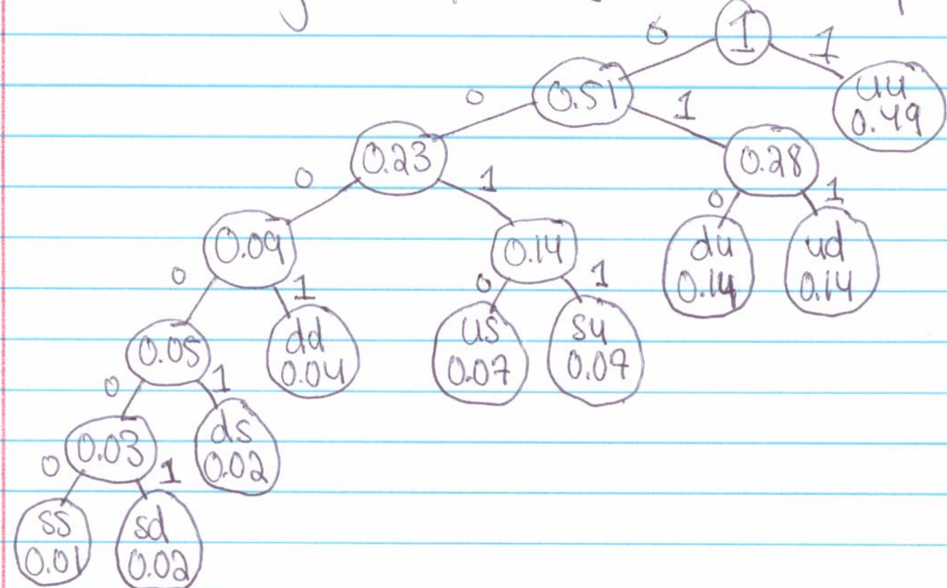
(2 pts.)

Entropy $= -(0.49\log_2(0.49) + 2\cdot 0.14\log_2(0.14) + 2\cdot 0.07\log_2(0.07) +$
$0.04\log_2(0.04) + 2\cdot 0.02\log_2(0.02) + 0.01\log_2(0.01))$

$= 2.3136$

(6 pts).

Close to achieving bits in practise: [Note - answers may vary]



Encoding Efficiency:

Combination	bits	total
uu	1	0.49 × 1
ud	011	0.14 × 3
us	0010	0.07 × 4
du	010	0.14 × 3
dd	0001	0.04 × 4
ds	00001	0.02 × 5
su	0011	0.07 × 4
sd	000001	0.02 × 6
ss	000000	+ 0.01 × 6
		2.33

2. (35 pts. total)

A. (2 pts.)

2^{32}

B. (8 pts total)

(3 pts.)

- 0 = 32 0's
- Most positive integer: 0 (MSB), followed by 31 1's.
- Most negative integer: 1 (MSB), followed by 31 0's.

(2 pts.)

- Decimal: Most Positive = 2147483647
- Decimal: Most Negative = -2144483648

(3 pts.)

- Negate the largest negative integer: Perform Two's Complement

$$\underset{31}{1}\ \underbrace{0}_{30-0} \Rightarrow \underset{31}{0}\ \underbrace{1}_{30-0}$$

$$\begin{array}{r} 011\ldots11 \\ +\qquad\quad 1 \\ \hline \underset{31}{1}\underbrace{0\ldots00}_{30-0} \end{array}$$

Get same value.

However, in decimal, a negated negative integer results in a positive integer (in our case 2144483648)

Solution: the result cannot be represented in 32-bit 2's complement.

C. (10 pts total, 2 pts each)

1. $411_{10} = 0000019B_{16}$
2. $-131072 = FFFE0000_{16}$
3. $00011000100001011010001111010011_2 = 1885A3D3_{16}$
4. $11111111111111111111111110001000_2 = FFFFFF88_{16}$
5. $-1_{10} = FFFFFFFF_{16}$

D. (10 pts total, 2 pts each, 3 bonus)

1) 42+36 =
$$\begin{array}{r} 00101010 \\ +00100100 \\ \hline 01001110 \end{array} = 78$$

2. 96-69 =
$$\begin{array}{r} 01100000 \\ +10111011 \\ \hline 00011011 \end{array} = 27$$

3. 69-96 =
$$\begin{array}{r} 01000101 \\ +10100000 \\ \hline 11100101 \end{array} = -27$$

4. 120-60 =
$$\begin{array}{r} 01111000 \\ +11000100 \\ \hline 00111100 \end{array} = 60$$

5. -60+120 =
$$\begin{array}{r} 11000100 \\ +01111000 \\ \hline 00111100 \end{array} = 60$$

6. $120 + (-120) =$

$$\begin{array}{r} 01111000 \\ +\ 10001000 \\ \hline 00000000 \end{array} = 0$$

7. $120 + 120 =$

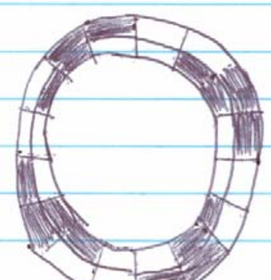
$$\begin{array}{r} 01111000 \\ +\ 01111000 \\ \hline 11110000 \end{array} = -16\text{, wanted } 240$$

240 can't be represented in 8-digit signed binary (-128 – $+127$)
Result: overflow error.

E. (5 pts.)
When people typically perform arithmetic in decimal, $A + (A) = 0$. However, in signed binary, $-A$ is represented by the MSB being 1 (followed by all 0's), for a value of -2^{n-1}. A is represented with its MSB being 0, (followed by all 1's), for a value of $2^{n-1} - 1$. The sum of $(A + {}^{-}A)$ in binary does not result in 0, but -1. Therefore, the "add 1" is necessary to complete the negation of a number in binary.

3. (25 pts. total)

A) (5 pts.)



Outer Ring = 2^0 digit
Inner Ring = 2^1 digit
= not a slot
= slot

Pulse Streams:

...00, 01, 11, 10... clockwise
...00, 10, 11, 01 counterclockwise

Note: It is acceptable for A to use a normal binary counting scheme than a Gray code.

B. (10 pts.)

Use a 3-bit Gray code: 000, 001, 011, 010, 110, 111, 101, 100
- The above sequence avoids glitches by changing 1 bit at a time.

C. [Note, answers may vary] (10 pts.)

To create an N-bit Gray code; take an (N-1) bit Gray Code and do the following:

- Fill out first 2^{N-1} rows with (N-1) bit code, and put 0's in (MSB).
- Fill out lower 2^{N-1} rows with (N-1) bit code, putting 1's in MSB.

For 3-bit Gray code: $\Longrightarrow$

0	00	← first 2^{N-1} rows with 2-bit Gray Code
0	01	
0	11	
0	10	

0	00	
0	01	
0	11	
0	10	
1	10	← fill table with 3-bit Gray code from the bottom upwards.
1	11	
1	01	
1	10	