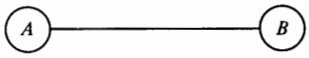
*	I	V	H	D
I				
V				
H				
D				

Granting associativity, explain why $\langle G, * \rangle$ is a group.

E. A Coin Game



Imagine two coins on a table, at positions A and B. In this game there are eight possible moves:

M_1: Flip over the coin at A.
M_2: Flip over the coin at B.
M_3: Flip over both coins.
M_4: Switch the coins.
M_5: Flip coin at A; then switch.
M_6: Flip coin at B; then switch.
M_7: Flip both coins; then switch.
I: Do not change anything.

We may consider an operation on the set $\{I, M_1, \ldots, M_7\}$, which consists of performing any two moves in succession. For example, if we switch coins, then flip over the coin at A, this is the same as first flipping over the coin at B then switching:

$$M_4 * M_1 = M_2 * M_4 = M_6$$

If $G = \{I, M_1, \ldots, M_7\}$ and $*$ is the operation we have just described, write the table of $\langle G, * \rangle$.

*	I	M_1	M_2	M_3	M_4	M_5	M_6	M_7
I								
M_1								
M_2								
M_3								
M_4								
M_5								
M_6								
M_7								

Granting associativity, explain why $\langle G, * \rangle$ is a group. Is it commutative? If not, show why not.

F. Groups in Binary Codes

The most basic way of transmitting information is to code it into strings of 0s and 1s, such as 0010111, 1010011, etc. Such strings are called *binary words*, and the number of 0s and 1s in any binary word is called its *length*. All information may be coded in this fashion.

When information is transmitted, it is sometimes received incorrectly. One of the most important purposes of coding theory is to find ways of *detecting errors*, and *correcting* errors of transmission.

If a word $\mathbf{a} = a_1a_2 \cdots a_n$ is sent, but a word $\mathbf{b} = b_1b_2 \cdots b_n$ is received (where the a_i and the b_j are 0s or 1s), then the *error pattern* is the word $\mathbf{e} = e_1e_2 \cdots e_n$ where

$$e_i = \begin{cases} 0 & \text{if } a_i = b_i \\ 1 & \text{if } a_i \neq b_i \end{cases}$$

With this motivation, we define an operation of *adding* words, as follows: If **a** and **b** are both of length 1, we add them according to the rules

$$0+0=0 \qquad 1+1=0 \qquad 0+1=1 \qquad 1+0=1$$

If **a** and **b** are both of length n, we add them by *adding corresponding digits*. That is (let us introduce commas for convenience),

$$(a_1, a_2, \ldots, a_n) + (b_1, b_2, \ldots, b_n) = (a_1 + b_1, a_2 + b_2, \ldots, a_n + b_n)$$

Thus, the sum of **a** and **b** is the error pattern **e**.

For example,

$$\begin{array}{r} 0010110 \\ +0011010 \\ \hline =0001100 \end{array} \qquad \begin{array}{r} 10100111 \\ +11110111 \\ \hline =01010000 \end{array}$$

The symbol $\mathbb{B}^n$ will designate the set of all the binary words of length n. We will prove that the operation of word addition has the following properties on $\mathbb{B}^n$:

1. It is commutative.
2. It is associative.
3. There is an identity element for word addition.
4. Every word has an inverse under word addition.

First, we verify the commutative law for words of length 1:

$$0 + 1 = 1 = 1 + 0$$

1 Show that $(a_1, a_2, \ldots, a_n) + (b_1, b_2, \ldots, b_n) = (b_1, b_2, \ldots, b_n) + (a_1, a_2, \ldots, a_n)$.

2 To verify the associative law, we first verify it for words of length 1:

$$1 + (1 + 1) = 1 + 0 = 1 = 0 + 1 = (1 + 1) + 1$$

$$1 + (1 + 0) = 1 + 1 = 0 = 0 + 0 = (1 + 1) + 0$$

Check the remaining six cases.

3 Show that $(a_1, \ldots, a_n) + [(b_1, \ldots, b_n) + (c_1, \ldots, c_n)] = [(a_1, \ldots, a_n) + (b_1, \ldots, b_n)] + (c_1, \ldots, c_n)$.

4 The identity element of $\mathbb{B}^n$, that is, the identity element for adding words of length n, is _____.

5 The inverse, with respect to word addition, of any word $(a_1, \ldots, a_n)$ is _____.

6 Show that $a + b = a - b$ [where $a - b = a + (-b)$].

7 If $a + b = c$, show that $a = b + c$.

G. Theory of Coding: Maximum-Likelihood Decoding

We continue the discussion started in Exercise F: Recall that $\mathbb{B}^n$ designates the set of all binary words of length n. By a *code* we mean a subset of $\mathbb{B}^n$. For example, below is a code in $\mathbb{B}^5$. The code, which we shall call C_1, consists of the following binary words of length 5:

00000
00111
01001
01110
10011
10100
11010
11101

Note that there are 32 possible words of length 5, but only eight of them are in the code C_1. These eight words are called *codewords*; the remaining words of $\mathbb{B}^5$ are not codewords. Only codewords are transmitted. If a word is received which is not a codeword, it is clear that there has been an *error of transmission*. In a well-designed code, it is unlikely that an error in transmitting a codeword will produce another codeword (if that were to happen, the error would not be detected). Moreover, in a good code it should be fairly easy to locate errors and correct them. These ideas are made precise in the discussion which follows.

The *weight* of a binary word is the number of 1s in the word: for example, 11011 has weight 4. The *distance* between two binary words is the number of positions in which the words differ. Thus, the distance between 11011 and 01001 is 2 (since these words differ only in their first and fourth positions). The *minimum distance* in a code is the smallest distance among all the distances between pairs of codewords. For the code C_1 above, pairwise comparison of the words shows that the minimum distance is 2. What this means is that *at least two* errors of transmission are needed in order to transform a codeword into another codeword; single errors will change a codeword into a *non*codeword, and the error will therefore be detected. In more desirable codes (for example, the so-called Hamming code), the minimum distance is 3, so any one or two errors are *always* detected, and only three errors in a single word (a very unlikely occurrence) might go undetected.

In practice, a code is constructed as follows: in every codeword, certain positions are *information positions*, and the remaining positions are *redundancy positions*. For instance, in our code C_1, the first three positions of every codeword are the information positions: if you look at the eight codewords (and confine your attention only to the first three digits in each word), you will see that every three-digit sequence of 0s and 1s is there namely,

000, 001, 010, 011, 100, 101, 110, 111

The numbers in the fourth and fifth positions of every codeword satisfy *parity-check equations*.

1 Verify that every codeword $a_1a_2a_3a_4a_5$ in C_1 satisfies the following two parity-check equations: $a_4 = a_1 + a_3$; $a_5 = a_1 + a_2 + a_3$.

2 Let C_2 be the following code in $\mathbb{B}^6$. The first three positions are the information positions, and every codeword $a_1a_2a_3a_4a_5a_6$ satisfies the parity-check equations $a_4 = a_2$, $a_5 = a_1 + a_2$, and $a_6 = a_1 + a_2 + a_3$.

(*a*) List the codewords of C_2.

(*b*) Find the minimum distance of the code C_2.

(*c*) How many errors in any codeword of C_2 are sure to the detected? Explain.

3 Design a code in $\mathbb{B}^4$ where the first two positions are information positions. Give the parity-check equations, list the codewords, and find the minimum distance.

If **a** and **b** are any two words, let $d(\mathbf{a}, \mathbf{b})$ denote the distance between **a** and **b**. To *decode* a received word **x** (which may contain errors of transmission) means to find the codeword closest to **x**, that is, the codeword **a** such that $d(\mathbf{a}, \mathbf{x})$ is a minimum. This is called *maximum-likelihood decoding*.

4 Decode the following words in C_1: 11111, 00101, 11000, 10011, 10001, and 10111.

You may have noticed that the last two words in part 4 had ambiguous decodings: for example, 10111 may be decoded as either 10011 or 00111. This situation is clearly unsatisfactory. We shall see next what conditions will ensure that every word can be decoded into only *one* possible codeword.

In the remaining exercises, let C be a code in $\mathbb{B}^n$, let m denote the minimum distance in C, and let **a** and **b** denote codewords in C.

5 Prove that it is possible to detect up to $m - 1$ errors. (That is, if there are errors of transmission in $m - 1$ or fewer positions of a codeword, it can always be determined that the received word is incorrect.)

6 By the *sphere of radius k* about a codeword **a** we mean the set of all words in $\mathbb{B}^n$ whose distance from **a** is no greater than k. This set is denoted by $S_k(\mathbf{a})$; hence

$$S_k(\mathbf{a}) = \{\mathbf{x} : d(\mathbf{a}, \mathbf{x}) \leq k\}$$

If $t = \frac{1}{2}(m - 1)$, prove that any two spheres of radius t, say $S_t(\mathbf{a})$ and $S_t(\mathbf{b})$, have no elements in common. [HINT: Assume there is a word **x** such that $\mathbf{x} \in S_t(\mathbf{a})$ and $\mathbf{x} \in S_t(\mathbf{b})$. Using the definitions of t and m, show that this is impossible.]

7 Deduce from part 6 that if there are t or fewer errors of transmission in a codeword, the received word will be decoded correctly.

8 Let C_2 be the code described in part 2. (If you have not yet found the minimum distance in C_2, do so now.) Using the results of parts 5 and 7, explain why two errors in any codeword can always be detected, and why one error in any codeword can always be corrected.