Problem 1. For each of the following codes indicate the code rate, the number of bit errors that can be detected when using the code, and the number of bit errors that can be corrected:

a. (15,11,3) code
b. (10,1,10) code
c. (154,132,7) code

Problem 2. Suppose management has decided to use 20-bit data blocks in the company’s new (n,20,3) error correcting code. What’s the minimum value of “n” that will permit the code to be used for single bit error correction, i.e. that will achieve a minimum Hamming distance of 3 between codewords?

Problem 3. A set of five 4-bit data values has been encoded using the (8,4,3) code shown in lecture and then transmitted over a noisy channel. For each of the received codewords below make the appropriate corrections to remove any errors, or indicate that an uncorrectable error has been detected.

a. 0 0 1    b. 1 0 1   c. 1 1 0   d. 0 0 1   e. 0 0 1
   1 0 1     0 0 0       1 0 1   1 1 1   1 1 1
   0 0         0 0        0 1        1 1        0 0

Problem 4. Ben Bitdiddle woke up in the middle of the night with the following great idea: in order to implement double-bit error correction he would use the (8,4,3) code described in lecture – which can correct single-bit errors – to encode a message twice. In other words, after the message was encoded for the first time with the (8,4,3) code, the resulting bit stream would be re-encoded with the same code a second time.

a. If the original message had 80 bits, how many bits will be in the doubly-encoded message?

b. Will Ben’s scheme work, i.e., will he be able to correct double-bit errors? Briefly describe why or why not.

Problem 5. As part of its efforts to automate mail delivery, the US Post Office often prints a bar code on each piece of mail as a way of encoding the destination zip code. The POSTNet code is based on a 2-of-5 code: there are five binary digits, exactly two of
which are “1”. To make it easy to read the code using an optical scanner, vertical lines of two different heights are used to represent 0 (short line) and 1 (tall line). Here’s the code:

<table>
<thead>
<tr>
<th>Zip code Digit</th>
<th>Encoding</th>
<th>Printed</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>00011</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>00101</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>00110</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>01001</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>01010</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>01100</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>10001</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>10010</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>10100</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>11000</td>
<td></td>
</tr>
</tbody>
</table>

a. When printing a POSTNet bar code, two tall vertical lines are added, one at either end. What digits does the following bar code depict (it’s not a zip code):

```
[ ] [ ] [ ] [ ] [ ] [ ]
```

b. If we say that each zip code digit is equivalent to 4 bits of information (i.e., k = 4), what’s the code rate (also called the information rate) of the POSTNet code?

c. Describe the error detection and correction capabilities of the POSTNet code assuming that we want to safely both detect and correct errors. What’s the number of errors per symbol that can be detected? Corrected?

d. When adding a POSTNet bar code to a piece of mail, the Post Office encodes a 5-digit zip code using 6 digits, where the sixth digit is chosen so that the sum of all six digits is 0 modulo 10. Briefly describe what purpose is served by adding this extra digit.

e. Ben Bitdiddle, having just finished 6.02, has taken a VI-A internship at the Post Office. After studying the POSTNet code for a while Ben writes an urgent memo to his supervisor suggesting the adoption of “binary zip codes” where zip code digits are constrained to be either “0” or “1”. Ben acknowledges that the original 5-digit zip codes would have be 17-digit binary zip codes, but he argues that using only the “0” and “1” encodings of the POSTNet code would enable much better error detection and correction. Briefly explain Ben’s thinking and answer the questions of part (C) for his proposed code.
**Problem 6.** The Hamming single-error-correcting code requires approximately $\log_2(N)$ parity bits to correct single-bit errors. Here’s how it works: start by renumbering the data bits with indices that aren't powers of two:

Indices for 16 data bits = 3, 5, 6, 7, 9, 10, 11, 12, 13, 14, 15, 17, 18, 19, 20, 21

The idea is to compute the check bits choosing subsets of the data in such a way that a single-bit error will produce a set of parity errors that uniquely indicate the index of the faulty bit:

- $p_0 = \text{even parity for data bits } 3, 5, 7, 9, 11, 13, 15, 17, 19, 21$
- $p_1 = \text{even parity for data bits } 3, 6, 7, 10, 11, 14, 15, 18, 19$
- $p_2 = \text{even parity for data bits } 5, 6, 7, 12, 13, 14, 15, 20, 21$
- $p_3 = \text{even parity for data bits } 9, 10, 11, 12, 13, 14, 15$
- $p_4 = \text{even parity for data bits } 17, 18, 19, 20, 21$

Note that each data bit appears in at least two of the parity calculations, so a single-bit error in a data bit will produce at least two parity errors. When checking a protected data field, if the number of parity errors is zero or one, the data bits are okay (exactly one parity error indicates that one of the parity bits was corrupted). If two or more parity errors are detected then the errors identify exactly which bit was corrupted.

a. What is the relationship between the index of a particular data bit and the check subsets in which it appears? Hint: consider the binary representation of the index.

b. If the parity calculations involving $p_0$, $p_2$ and $p_3$ fail, assuming a single-bit error what is the index of the faulty data bit?

c. The Hamming SECC doesn't detect all double-bit errors. Characterize the types of double-bit errors that will not be detected. Suggest a simple addition to the Hamming SECC that allows detection of all double-bit errors.