For all questions below, show your work. Do not just write down numbers. Do not show every single addition, but show enough work to convince the reader you know what you are doing.

**Problem 1**

Let $C_1$ and $C_2$ be two codes for the same source with codeword lengths $N_1$ and $N_2$, respectively. We define the product code $C_1 C_2$ as follows. This code has codewords of length $N_1 \cdot N_2$. Given a string of that length, arrange it in a matrix with $N_2$ rows and $N_1$ columns, filled from left to right starting with the top row (that is, the $(N_1 + 1)$-st bit in the string is in row 2 column 1 of the matrix). Now, this string is a codeword in the $C_1 C_2$ code if and only if every row is a codeword of $C_1$ and every column is a codeword of $C_2$.

(a) Show that if $C_1$ and $C_2$ are arbitrary codes, their product could be empty.

(b) Show that if $C_1$ and $C_2$ are linear codes, their product is a linear code as well.

Let $C_1$ and $C_2$ be the codes with the following generator matrices:

$$G_1 = \begin{pmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{pmatrix}, G_2 = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}$$

So, $C_1$ is a $[4, 2]$ code and $C_2$ is a $[3, 2]$ code. Think of each codeword in $C_1 C_2$ as a $3 \times 4$ matrix. One way to view encoding is to arrange a block of $2 \times 2$ input bits in the top left corner of a $3 \times 4$ matrix, then complete the rows to be codewords in $C_1$, and finally complete the columns to be codewords in $C_2$.

Let $u$ and $v$ be two column vectors. Their outer product is $u \cdot v^T$, which is a matrix with $|u|$ rows and $|v|$ columns.

(c) Explain how to perform encoding in $C_1 C_2$ in terms of outer products of basis vectors of the original codes. You may assume that the source $K_1 \times K_2 = 2 \times 2$ bits are arranged in a $2 \times 2$ block of the $3 \times 4$ codeword matrix, left to right, starting with the top row.

Now take every codeword $3 \times 4$ matrix and view it as a 12 bit string, by listing the top row, left to right, then the second row and finally the third row.

(d) Give a generator matrix for $C_1 C_2$ seen as a linear $[12, 4]$ code in the manner described above.

It can be shown that if the minimum distances of $C_1$ and $C_2$ are $d_1$ and $d_2$, respectively, then the minimum distance of $C_1 C_2$ is $d_1 \cdot d_2$. Thus, for the specific codes above, the minimum distance of $C_1 C_2$ is $2 \cdot 2 = 4$.

(e) Imagine we use the product code across the BSC. Give an example of a transmission that has distance 2 from 2 distinct codewords. That is, assuming the channel introduced 2 independent errors, there are 2 equally possible ways to decode the transmission.

(f) Imagine we use the product code across a special type of BSC in which errors come non-independently, in pairs of consecutive bits. Show that if the channel modifies only 2 consecutive bits (remember, we transmit the top row, then the middle row, then the bottom row, each left to right) we can always recover the correct codeword.

(g) Is (f) above still true if we replace the row code $C_1$ by $C_1'$ that has the following generator matrix?

$$G_1' = \begin{pmatrix} 1 & 0 & 0 & 1 \\ 0 & 1 & 1 & 0 \end{pmatrix}$$

**Problem 2**

Let $C_3$ be the identity $[N, N]$ linear code which has as generator matrix the identity $N \times N$ matrix. Let $C_4$ be the $[7, 4]$ Hamming code we saw in class. Consider the $C_3 C_4$ product code across the BSC ($C_3$ is used for the rows).
(a) Assume we know the channel flipped $t$ independent bits in a codeword. For what value of $t$ can we be sure to recover the correct codeword?

(b) Assume we know the channel flipped $N$ consecutive bits in a codeword. Show that we can recover the correct codeword.

(c) Using an algorithm similar to part (b) above, what is the maximum $R$ so that we can always recover $N + R$ consecutive bit flips? For this question, do not lose yourself in case analysis just to add 1 or 2 to $R$. If you have an idea of how to deal with $R$ greater than 0, explain that idea, and say for how large an $R$ it works.

Problem 3

In this problem we provide a justification for the choice of random low-density parity check matrices over general parity check matrices when it comes to using the message-passing algorithm we talked about in class. Even though this question is not about numerical computations, it might help to keep in mind a typical setting of parameters.

Suppose we want to communicate over the BEC with erasure probability $f$ (think of $f$ being small, like .1). We know this channel has capacity $C = 1 - f$. We want to construct a linear code that has rate $R$, with $R$ very close to but less than $C$ (for instance, $R = C - \delta$ for some very small $\delta$). Knowing our target rate $R$, we choose $K$ and $N$ so that $K/N = R$. The specific choice of $K$ and $N$ depends on $R = 1 - f - \delta$, thus on $f$ and $\delta$. For the purposes of this question, $f$ is fixed, and we study what happens as $\delta$ approaches 0. You may assume without proof that as $\delta$ goes to 0, $N$ goes to infinity.

In scenario 1, we construct our $[N, K]$ code $C_1$ as follows. Pick a random parity check matrix $H_1$ by setting each bit to 1 independently at random with probability $\alpha$, where $\alpha$ is a constant in $(0, 1)$ (for instance, if the bits are picked uniformly at random, $\alpha = 1/2$).

In scenario 2, we construct our $[N, K]$ code $C_2$ as follows. Pick a random parity check matrix $H_2$ by setting each bit to 1 independently at random with probability $\beta/N$, where $\beta$ is a constant like 6.

Note. Scenario 2 doesn’t always generate a low-density parity check matrix. To see this, note that there is a very small chance that, when flipping all $N$ bits corresponding to one row, we will get all 1s. The probability of this bad event happening is exactly $(\beta/N)^N$, which is a very small number when $N$ is large. However, the expected number of 1s per each row in $H_2$ is $(\beta/N)\cdot N = \beta$, which is a constant, like in a LDPC matrix. That’s why we consider scenario 2 to be very relevant to a generating a true LDPC matrix.

Suppose we transmit a codeword $w$ of length $N$ across our BEC, and we receive $r$.

(a) What is the expected number of ‘?’-s in $r$?

From now on, assume that the number of ‘?’-s in $r$ is exactly equal to the expected number you found in (a). If you do not know how to solve (a), and only then, use $(.3) \cdot N$ as the number of ‘?’-s.

To recover the ‘?’-s, we will use the message passing algorithm we talked about in class. Note that, in the graph associated with the parity check matrix, in order to derive the true value of the transmitted-bit node $i$, we must have a check-bit node $j$ such that $j$ is a neighbour of $i$ and we know the values of all other neighbours of $j$. We call $j$ a useful check-bit node, as we can use it to derive a new transmitted bit.

The questions below are about the very beginning of the message passing algorithm:

(b) Using code $C_1$, what is the probability that the check-bit node $j$ is useful?

(c) Using code $C_1$, give an upper bound for the probability that there exists a useful check-bit node.

(d) Using code $C_2$, what is the probability that the check-bit node $j$ is useful?

(e) Using code $C_2$, what is the probability that the check-bit node $j$ is not useful?

(f) Using code $C_2$, give an upper bound for the probability that none of the check-bit nodes are useful.

(g) Discuss what (c) and (f) tell you about decoding the codes $C_1$ and $C_2$ as $N$ grows.