Problem 1

The distance between any two code words is at least 2. This is because, no two code words can have a distance of 1. For contradiction assume that the codewords $\{u_i\}$ and $\{v_i\}$ have distance 1. Then $u_i = v_i$ for all i except some i_0. But $\sum_i u_i = 0 = \sum_i v_i$, which implies that $u_{i_0} = v_{i_0}$, leading to a contradiction.

This implies that the code can detect one error but cannot correct any errors.

In order to compute the rate, we first count the number of valid codewords. This can be done by recursion. Let $n(a, b)$ be the number of sequences over $\{-2, \cdots, 2\}^a$ that sum to b. Then, $n(a, b) = n(a-1, b-2) + n(a-1, b-1) + n(a-1, b) + n(a-1, b+1) + n(a-1, b+2)$. $n(0, i) = 1$ for all $i \in \{-2, \cdots, 2\}$. Solving this recurrence using a smal program gives $n(5, 0) = 381$.

The total number of sequences are 5^5. Thus the rate of encoding is $\frac{\log 381}{\log 5^5} = 0.739$.

Problem 2

A. The check bit is given by the equation $2(1) + 3(3) + 4(2) + 7(8) + 9(9) + 10(x) = 0 \pmod{11}$, or $2 + 10x = 0 \pmod{11}$. We get $x = 2$.

B. Let the original codeword be $u_1 \cdots u_{10}$, and the transposed numerals be at positions j and k respectively, $j \neq k$. That is, $u'_i = u_i$ for all $i \neq j, k$, $u'_j = u_k$, and $u'_k = u_j$. By definition we have, $\sum_i iu_i = 0 \pmod{11}$.

Assume that the transposition is not detected. Then, $\sum_i iu'_i = 0 \pmod{11}$. Thus, $\sum_i iu_i = \sum_i iu'_i \pmod{11}$, or, $ju_j + ku_k = ju_k + ku_j \pmod{11}$. This implies, $(j-k)u_j = (j-k)u_k$ mod 11. But, $j - k \neq 0 \pmod{11}$. Thus, we must have $u_j = u_k \pmod{11}$, or, $u_j = u_k$ because both the numbers are less than 11. Thus the two numbers are the same and no transposition took place!

C. The missing bit is given by the equation $2(1) + 3(3) + 4(2) + 5(8) + 6(x) + 7(7) + 8(9) + 9(6) + 10(10) = 0 \pmod{11}$, or, $4 + 6x = 0 \pmod{11}$. We get $x = 3$.

Problem 3

Code C_1 is generated by the polynomial $g_1(x)$. Thus it is also generated by any polynomial that is a multiple of $g_1(x)$. In order to generate $C_1 \cup C_2$, we need to use a polynomial that is a multiple of both g_1 and g_2. The smallest such code is given by the least common multiple of g_1 and g_2.

Likewise, $C_1 \cap C_2$ is generated by the greatest common divisor of g_1 and g_2.

Problem 4

We divide the message into sequences of length 16 bits each. Let these sequences be $S_1S_2S_3\cdots$. Then, we encode $S_1S_5S_9\cdots$ using the RS(255,223) encoder. Likewise we encode $S_2S_6S_{10}\cdots$, $S_3S_7S_{11}\cdots$, and $S_4S_8S_{12}\cdots$ using the RS(255,223) encoder. Clearly the rate is preserved. Furthermore, note that, any consecutive bit errors containing up to 64 bits are divide over the four sequences, such that each sequence contains at most 16 of them consecutively. Thus this code can correct up to 64 consecutive bit errors.