Problem 1

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 2

Here is one version of G.

$$\begin{pmatrix}
1&0&0&0&0&0&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&0&0&0&0&0&0&0&0&0&0&0\\
0&1&0&0&0&0&1&1&1&1&1&1&1&1&0&0&0&0&0&0&0&1&1&1&1&1&1&1&0&0&0&0\\
0&0&1&0&0&0&1&1&1&1&0&0&0&0&1&1&1&1&0&0&0&1&1&1&1&0&0&0&1&1&1&0\\
0&0&0&1&0&0&1&1&0&0&1&1&0&0&1&1&0&0&1&1&0&1&1&0&0&1&1&0&1&1&0&1\\
0&0&0&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&0&1&1&0&1&0&1&0&1&1&0&1&1\\
0&0&0&0&0&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1&1
\end{pmatrix}$$

Any 3 rows or combinations would do.

Problem 3

We divide the message into sequences of length 16 bits each. Let these sequences be $S_1 S_2 S_3 \cdots$. Then, we encode $S_1 S_5 S_9 \cdots$ using the RS(255,223) encoder. Likewise we encode $S_2 S_6 S_{10} \cdots$, $S_3 S_7 S_{11} \cdots$, and $S_4 S_8 S_{12} \cdots$ 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.

Problem 4

A. No, for $\mathrm{GF}(p^r)[x]$, $x - \alpha$ is only a factor of $x^n - 1$ if $\alpha^n = 1$. This is always true if $n = p^r - 1$, as assumed in class, but not necessarily for other n.

B. $x^{15} - 1$ has 5 distinct factors. This gives us $2^5 = 32$ possible products of the factors. Ignoring the two trivial generators, 1 and $x^{15} - 1$, this gives us 30 distinct codes. (We also accepted the answer 32).

C. First note that in $GF(2)$, 1 and -1 are the same. We have $(x^7 - 1) = (x+1)(x^6 + x^5 + x^4 + x^3 + x^2 + x^1 + x)$, so $g = (x^6 + x^5 + x^4 + x^3 + x^2 + x^1 + x)$, and $k = 1$. This gives two possible messages, 0 and 1, and two codewords 1111111 and 0000000. This code just repeats be bits n times and I guess it is usefull if you want redundancy for for a single bit.

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

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

E. To help distinguish between the polynomials of $\mathrm{GF}(2^2)$ and the polynomials used by the code, I'll use y for the first. We have $\alpha = y = \mathbf{2}, \alpha^2 = y + 1 = \mathbf{3}, \alpha^3 = 1 = \mathbf{1}$, where the boldface numbers are the names we give the elements of $\mathrm{GF}(2^2)$. $g = (x-\alpha)(x-\alpha^2) = (x+\alpha)(x+\alpha^2) = x^2 + (\alpha + \alpha^2) + \alpha^3 = x^2 + x + 1$. In general for a message m, $mx^2 mod(x^2 + x + 1)$ is going to give $mx + m$, so the whole message will be mmm. Therefore the valid codewords are 000000, 010101, 101010 and 111111.