Problem 1. (15 points)

A. \[ \sum_{i=0}^{n} (2i + 1) = (n + 1)^2 \text{ for } n \geq 0 \]

**Basis (n=0):**

\[ \sum_{i=0}^{0} (2i + 1) = (0 + 1)^2 = 1^2 = 1 \]

**Induction (Assuming formula is true for arbitrary } n \geq 1, \text{ prove it is true for } n+1):**

\[ \sum_{i=0}^{n+1} (2i + 1) = \sum_{i=0}^{n} (2i + 1) + (n+1) = (n+2)^2 \]

\[ = (n^2 + 2n + 1) + (2n + 3) = n^2 + 4n + 4 \]

\[ = \sum_{i=0}^{n} (2i + 1) = (n + 1)^2 \text{ for } n \geq 0 \quad \text{Q.E.D.} \]

B. Every non-negative integer can be expressed in the form \(2a + 3b\) for some integers \(a\) and \(b\).

**Basis (n = 0):** \(0 = 2(0) + 3(0) \Rightarrow a = b = 0 \quad \checkmark\)

**Basis (n = 1):** \(1 = 2(2) + 3(-1) \Rightarrow a = 2, b = -1 \quad \checkmark\)

Any integer \(n \geq 2\) can be expressed as the sum of two integers \(j\) and \(k\), \(n = j + k\), where both \(j\) and \(k\) are in the range 1, 2, ..., \(n-1\). Since \(j\) and \(k\) are smaller than \(n\), the desired property is assumed to be true for \(j\) and \(k\) by the inductive hypothesis. Thus, for the value \(n\):

\[ n = 2(a_n) + 3(b_n) \]

\[ j + k = 2(a_n) + 3(b_n) \]

\[ (2(a_j) + 3(b_j)) + (2(a_k) + 3(b_k)) = 2(a_n) + 3(b_n) \]

\[ 2(a_j + a_k) + 3(b_j + b_k) = 2(a_n) + 3(b_n) \]

This relationship is true if \(a_n = (a_j + a_k)\) and \(b_n = (b_j + b_k)\). Since \(a_j, a_k, b_j,\) and \(b_k\) all are integers, \(a_n\) and \(b_n\) also are integers and the property is proved.

**NOTE:** Other proofs also are possible.
Problem 2. (15 points)

A. \[ \begin{array}{cccccccccccc}
    1 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 \\
    1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 1 & 1 \\
\end{array} \]

  \[ \begin{array}{cccccccc}
    d_7 & \ldots & d_0 & c_4 & \ldots & c_0 \\
    1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 0 \\
    1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
  \end{array} \]

B. 13-bit value Re-computed Explanation

\begin{array}{cccccccc}
    d_7 & \ldots & d_0 & c_4 & \ldots & c_0 & c_4 & \ldots & c_0 \\
    1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 0 & 0 \\
    1 & 1 & 1 & 0 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\
    1 & 1 & 1 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 1 & 1 \\
\end{array}

- check bits \( c_3, c_2, \) and \( c_0 \) don’t match; matches \( d_1 \) row, so \( d_1 \) must be in error. Correct data byte: 1 0 1 1 1 0 0 0
- only check bit \( c_2 \) doesn’t match; so it must be the single bit error; data byte is correct.
- check bits \( c_4, c_3, c_1, \) and \( c_0 \) don’t match; no corresponding matrix row, so this is a detectable, but not correctable, error.

C. Minimum Hamming distance between valid code words = 4

\begin{array}{cccccccc}
    1 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 & 1 & 1 \ \\
    0 & 1 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 1 \\
    0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 1 & 0 & 1 \\
\end{array}

Many examples are possible. Changing data bit \( d_7 \) from 1 to 0 changes 3 check bits, for a distance of four between the two valid code words.

Problem 3. (15 points)

A. \( O(n \log n) \) \( T(n) \rightarrow T(n-1) \) implies approximately \( n \) iterations and \( b \log n + c \) work is done on each iteration. The highest order term in the product is, thus, \( n \log n \).

B. Repeated calculation: (See next page of solution for the repeated substitution approach.)

\[
\begin{align*}
T(0) &= a \\
T(1) &= T(0) + b \log(1) + c = a + b \log(1) + c \\
T(2) &= T(1) + b \log(2) + c = (a + b \log(1) + c) + b \log(2) + c = a + b (\log(1) + \log(2)) + 2 c \\
T(3) &= T(2) + b \log(3) + c = (a + b (\log(1) + \log(2)) + 2 c) + b \log(3) + c \\
&= a + b (\log(1) + \log(2) + \log(3)) + 3 c \\
\vdots \\
T(n) &= a + b (\log(1) + \log(2) + \ldots + \log(n-1) + \log(n)) + n c \\
\log(j / k) &= \log(j) - \log(k), \text{so the term in parentheses can be rewritten as follows:} \\
&= \log(n/n) + \log(n/(n/2)) + \ldots + \log(n/(n/(n-1))) + \log(n/1) \\
&= \log(n) - \log(n) + \log(n) - \log(n/2) + \ldots + \log(n) - \log(n/(n-1)) + \log(n) - \log(1) \\
&= n \log(n) - \text{sum of smaller terms} \\
\therefore \ T(n) &= a + b (n \log(n) - \text{sum of smaller terms}) + c n
\end{align*}
\]
Repeated substitution:

\[ T(n) = T(n-1) + b \log(n) + c \]

\[ = \{T(n-2) + b \log(n-1) + c\} + b \log(n) + c = T(n-2) + b (\log(n-1) + \log(n)) + 2c \]

\[ \ldots \]

\[ T(n) = \{T(n-n) + b \log(n-(n-1)) + c\} + b (\log(2) + \log(3) + \ldots + \log(n-1) + \log(n)) + (n-1) c \]

\[ \therefore \quad \text{As with repeated calculation, } T(n) = a + b (n \log(n) - \text{sum of smaller terms}) + c n \]

C. “Guess” the following solution to \( T(n) \), which can be solved exactly by the inductive proof method:

\[ T(n) = d n \log n + e n + f \quad (d, e, f \text{ are the unknown constants for the exact solution}) \]

**Problem 4. (15 points)**

A. By choosing \((m-1)\) of the \((m + n - 2)\) total moves as horizontal, the remaining moves are vertical by default and the sequence defines a distinct path from A to B. The number of distinct paths thus equals the number of ways to choose (unordered selection) the horizontal moves:

\[
\binom{m + n - 2}{m - 1} = \frac{(m + n - 2)!}{(m - 1)! ((m + n - 2) - (m - 1))!} = \frac{(m + n - 2)!}{(m - 1)! (n - 1)!}
\]

B. Imagine a bag containing \((m-1)\) chips labeled “horizontal” and \((n-1)\) labeled “vertical.” Each sequence of chips drawn defines a path from A to B (e.g., V H V …). Since you cannot distinguish which “vertical” chip is drawn first, second, etc., the number of distinct paths (sequence of chips) can be calculated as an ordering with identical items:

\[
\frac{\text{Total possible sequences}}{\text{indist (horiz) (indist vert)}} = \frac{((m - 1) + (n - 1))!}{(m - 1)! (n - 1)!}
\]

C. Paths from cell A to cell B that pass through cell C can be divided into two segments, as shown to the right. Since every path from A to C can be combined with every path from C to B:

\[ \#\text{Paths}_{ACB} = \#\text{Paths}_{AC} \cdot \#\text{Paths}_{CB} \]

ALL paths from A to B must pass through exactly one of the marked cells (C, D, C’, D’) in the figure to the right. Since the layout is symmetric:

\[ \#\text{Paths}_{AB} = 2 (\#\text{P}_{AC} \cdot \#\text{P}_{CB} + \#\text{P}_{AD} \cdot \#\text{P}_{DB}) \]

\[ \#\text{P}_{AC} = (5+3-2)! / (4! 2!) = 15 \quad \#\text{P}_{CB} = (2+6-2)! / (1! 5!) = 6 \]

\[ \#\text{P}_{AD} = (6+2-2)! / (5! 1!) = 6 \quad \#\text{P}_{DB} = (1+7-2)! / (1! 6!) = 1 \]

\[ \#\text{Paths}_{AB} = 2 ( (15) (6) + (6) (1) ) = 2 (96) = 192 \]

**Problem 5. (15 points)**

Answer depends on your code. In most cases \( \text{prime}(n) \) was \( O(n) \), since it contained a loop for \( i=2 \) to \( n-1 \) that was potentially shortened if an exact divisor of \( n \) was found; i.e., if \( n \) was not prime.