Maximum score: 230 points.
Instructions: For this test, you work in teams to solve a multi-part, proof-oriented question. Problems that use the words "compute," "list," or "draw" require only an answer; no explanation or proof is needed. **Unless otherwise stated, all other questions require explanation or proof.** The problems are ordered by content, *not difficulty*. The difficulties of the problems are generally indicated by the point values assigned to them; it is to your advantage to attempt problems throughout the test. In your solution for a given problem, you may cite the statements of earlier problems (but not later ones) without additional justification, even if you haven't solved them. Footnotes are not necessary to understand the contents of the round. **Do not discuss the contents of this Power Round until November 20.**

1 Error Correcting Codes: A Tale of Bob and Alice (26 pts)

Consider a situation much too common in our modern world: a remote math contest participant, Alice, wants to send her answers to Bob, a staff member of the math contest. There is a "channel" between them, which we can think of as something that Alice can send data across as a bunch of characters. Bob can then receive these characters. The set of possible characters has a name.

Definition 1.1. An **alphabet** is some set Σ of characters. Common alphabets include

- $\Sigma_2 = \{0, 1\}$
- $\Sigma_{10} = \{0, 1, 2, \ldots, 9\}$
- $\Sigma_{\alpha\beta} = \{\texttt{A}, \texttt{B}, \texttt{C}, \ldots, \texttt{Z}\}$

Before she can answer any questions, Alice has to send her name. Beforehand, the pair agree to use the alphabet $\Sigma_{\alpha\beta}$. Alice types in the **codeword**

A L I C E

and sends it through the channel.
Now, Bob is receiving Alice's answer. At the other end he sees:

A L _ _ E

Oh no! Our channel has "erased" the 3rd and 4th characters into underscores (_'s are not in our alphabet). This is a fact of life, as real-world channels often are not reliable enough to send every character perfectly. Let's say that Alice and Bob determine the following empirically: on any given transmission, we are guaranteed that at most two characters will be erased (but we don't know beforehand which ones). Luckily, Alice has a great idea that she has shared with Bob beforehand.

Scheme 1.1. Just like how at a loud party, you have to repeat your message for people to get it, Alice will duplicate her message 3 times, sending:

A L I C E A L I C E A L I C E

Now, Bob may receive

A L I _ E A L _ C E A L I C E

Then Bob does the following:

1. Bob receives a message of length n and divides n by 3 to get the length of the original word, which we denote $\ell = n/3$. In our example, this means he knows the message has length 5.
2. Bob places a marker at every ℓth letter. In our example he places markers in front of all 3 A's.
   (a) He compares all 3 of the letters at this position. He picks a letter that is not an underscore (all the non-underscore letters are necessarily the same).
   (b) He adds this letter to our draft message.
   (c) He advances to the next position and repeat this ℓ times.
3. At the end, our "draft message" is the message we recover.

If we run our scheme on the example:

1. We see 3 A's, so our first letter is A
2. We see 3 L's, so our second letter is L
3. We see 2 I's and 1 _, so our third letter is I
4. We see 2 C's and 1 _, so our fourth letter is C
5. We see 3 E's, so our fifth letter is E

This means Bob recovers A L I C E !
The scheme works only because at most 2 characters can be erased, so if you get to a position where 2 of the regions have an underscore, the third region will still have the letter.

Question 1.1 (3 pts)**.** Suppose now, instead of the channel "erasing" 2 characters, 2 characters are "corrupted" into random characters that are in the alphabet. In the example above, Bob may instead receive:

A L I X E A L Y C E A L I C E

Modify Scheme 1.1 such that step 2 (a) reads:

- He compares all 3 of the letters at this position, picking the one that is the "majority." For example, if the three letters are A, A, C, then Bob would pick A.

Does the scheme still work? If it does, give a proof. If not, give a counter-example.

Question 1.2 (4 pts)**.** Suppose that we had s characters erased and Alice's message has length n. What is the minimum amount of times t that Alice needs to repeat her message to allow Scheme 1.1 to work (replacing 3 in the scheme with t)?

Let's define what has happened formally.

Definition 1.2. A **set** is a collection of elements with no duplicates. The following examples use the sets from Definition 1.1.

- To denote the size of any set, we use $|\cdot|$. For example, $\left|\Sigma_{\alpha\beta}\right| = 26$.
- To denote an element being a member of a set, we use $\in$. To denote an element not being a member of a set, we use $\notin$. For example, $3 \in \Sigma_{10}$ but $3 \notin \Sigma_2$.
- To denote subsets (i.e. every element of one set is also an element of the other), we use $\subseteq$. For example, $\Sigma_2 \subseteq \Sigma_{10}$.
- To denote sets of ordered tuples or sequences, we use $\times$. For example,

$$\Sigma_{10} \times \Sigma_2 = \{(0,0),(0,1),(1,0),(1,1),\ldots,(9,0),(9,1)\}$$

- To denote using the $\times$ on a set with itself, we use exponents. For example,

$$\Sigma_2^3 = \Sigma_2 \times \Sigma_2 \times \Sigma_2 = \{(0,0,0),(0,0,1),(0,1,0),(0,1,1),\ldots,(1,1,0),(1,1,1)\}$$

Definition 1.3. A **code** C **of length** n is some subset of Σ^n. Each element of C is called a **codeword**. The elements of Σ^n are just called **words**.

Using the sets in Definition 1.1,

Question 1.3 (3 pts)**.** Compute $\left|\Sigma_2^m\right|$ in terms of m.

Question 1.4 (4 pts)**.** Compute the number of codes over alphabet Σ_{10} with codeword length n.

Question 1.5 (5 pts)**.** Compute the number of codes over alphabet $\Sigma_{\alpha\beta}$ with the following properties:

1. The length of each codeword is 10.
2. Each codeword has the substring 'SUSBUS' (where the letters are contiguous, in one block).

You may leave your answer in terms of exponents and binomial coefficients.

Definition 1.4. Consider an encoding setup with alphabet Σ and code $C \subseteq \Sigma^n$. The **message** m is the meaningful information we want to send (like Alice's name). From the time we have a message to the time the person at the other end of the channel receives it, we undergo three different stages.

- **Encoding** - Encoding function E takes a message m and returns its corresponding codeword $E(m)$.
- **Transmission** - The message passes through a transmission channel T, which takes a codeword c in C and returns a string c' in Σ^n.
- **Decoding** - Decoding function D takes an element w of Σ^n (not necessarily a valid codeword) and returns a corresponding message $D(w)$.

This means for Scheme 1.1:

- $C =$ the set of all words that are a message repeated three times.

- So 'ALICEALICEALICE' $\in C$ but 'ALICEALICEBOBBY' $\notin C$.

- $E(m) = (m, m, m)$ (our "codeword" is our original message repeated 3 times)
- T takes c and returns c with 2 random characters erased
- $D(w) =$ the result of running Scheme 1.1 on w

Question 1.6 (2 pts)**.** Give an expression for the final message received given an initial message m which passes through the encoding, transmission, and decoding stages in Definition 1.4. For this question, let $T(c)$ denote the output of the transmission channel on some codeword c.

Question 1.7 (5 pts)**.** Suppose that T does not change the codeword at all (it takes codeword c and returns c) and we choose E and D such that our final message always matches our initial message m. Prove that if $x \neq y$, then $E(x) \neq E(y)$.

2 The Hamming Bound and Hamming Codes (58 pts)

2.1 Distances and the Hamming Bound

How can we compare the efficacy of schemes? Well, one answer is to look at a code's **distance**, which tells us how much wiggle room there is between codewords.

Definition 2.1. The **Hamming distance** between two words $x \in \Sigma^n$ and $y \in \Sigma^n$ is the number of places at which they differ. We will denote this by $\delta(x, y)$.
For a code C, its **distance** $d(C)$ is the minimum Hamming distance between any two distinct codewords. Stated mathematically,

$$d(C) = \min_{\substack{c,c' \in C \\ c \neq c'}} \delta(c, c')$$

In addition, the **Hamming Ball of radius** r **centered at** c, denoted $B_c(r)$, is the set of all words that are Hamming distance r or less from c (including c itself).

For example, if

$$x = \texttt{A L I C E} \qquad y = \texttt{A L I X Y} \qquad z = \texttt{A D I Y X}$$

then $\delta(x, y) = 2$ and $\delta(y, z) = 3$.

Question 2.1 (5 pts)**.**

Show that Hamming distance satisfies the triangle inequality: for any three words x, y, and z,

$$\delta(x, y) + \delta(y, z) \geq \delta(x, z).$$

Question 2.2 (3 pts)**.**

The **0-1 distance** between two words is:

$$\delta_{0-1}(x, y) = \begin{cases} 0 & x = y \\ 1 & x \neq y \end{cases}$$

For any two messages x, y, which of the following is always true? Explain your reasoning.

(a) $\delta_{0-1}(x, y) \leq \delta(x, y)$
(b) $\delta_{0-1}(x, y) = \delta(x, y)$
(c) $\delta_{0-1}(x, y) \geq \delta(x, y)$
(d) None of (a), (b), or (c)

Now for Alice and Bob's code, what is $d(C)$? Well, since the message is repeated three times, you have to change all the letters at the same position for the message to be part of C. This means any two distinct codewords in C must differ by at least 3 characters. This tells us $\delta(c, c') \geq 3$ for all $c, c' \in C$ such that $c \neq c'$, so $d(C) \geq 3$ as well. On the other hand,

A L I C E A L I C E A L I C E and A L I C T A L I C T A L I C T

are code-words that differ in exactly three places, so the minimum distance over all code-words $d(C)$ must be at most 3. Thus, $d(C) = 3$.

Definition 2.2. An $[n, k]$ code is a code of length n, message length k. If we know the distance d, then we can also call it a $[n, k, d]$ code.

So the repetition code (Scheme 1.1) we have studied is a $[15, 5]$ or $[15, 5, 3]$ code. Note that with this distance, the code was able to fix 2 characters erased. This is no accident. In fact, we can show the following:

Consider a code C with odd minimum distance $d(C)$.

Question 2.3 (6 pts)**.** Show that we can always correct erasures of up to $d(C) - 1$ characters. (In other words, suppose we have a partially erased word $\tilde{c}$ that is the result of erasing up to $d(C) - 1$ characters from codeword c. Show that the only codeword such that erasing at most $d(C) - 1$ characters gives $\tilde{c}$ is c.) Explain why $d(C)$-character erasures cannot always be corrected.

Question 2.4 (7 pts)**.** Show that we can always correct corruptions of up to $\frac{d(C)-1}{2}$ characters. (*Hint*: Suppose a corrupted word $\tilde{c}$ is distance less than $\frac{d(C)-1}{2}$ from two distinct codewords c and c'. What would this mean?)

The question remains: if all that matters is the distance, what is the best code for the job? One answer may be the code with the most codewords, because then we can represent the most messages.

Consider a code C with length n and $d(C) = 3$ over the alphabet $\Sigma_2 = \{0, 1\}$.

Question 2.5 (2 pts)**.** How many possible words are there of length n over the alphabet Σ_2?

Question 2.6 (2 pts)**.** Consider some $c \in C$. What is $|B_c(1)|$? (Here, $B_c(1)$ denotes the Hamming ball of radius 1 centered at c; see Definition 2.1.)

Question 2.7 (6 pts)**.** Question 2.4 shows that we can fix a 1-character corruption, i.e. the balls of radius 1 about every codeword in C will not overlap (the one corresponding to the codeword we are decoding to). Use this to show:

$$|C| \leq \frac{2^n}{n+1}.$$

The bound we have derived is called the **Hamming bound**.

2.2 Hamming Codes

2.2.1 Motivating Hamming Codes

Due to the nice result from above, **for the rest of this section, we will only consider binary codes.**

Definition 2.3. A **binary code** is a code over the alphabet $\Sigma_2 = \{0, 1\}$. Each 0 or 1 character is called a **bit**.

We now have an idea of how efficient binary codes can be based on the Hamming bound: we know that $|C| \leq \frac{2^n}{n+1}$. The bigger the size of the code, the more messages we can send. The obvious question to ask here is:

$$\text{Are there any codes which achieve the Hamming bound; i.e, codes with } |C| = \frac{2^n}{n+1}?$$

The answer turns out to be **yes**: a family of codes known as **Hamming codes** achieve the Hamming bound of peak efficiency.

As done previously, let's fix distance $d = 3$ for simplicity. Suppose we want to send a 4-bit message (i.e. a length 4 binary message). The idea is that we need to introduce extra information in the form of "parity check" bits that watch over 3 out of 4 message bits each. We see that to account for the data from each of the message bits, we require 3 such parity bits[1]. A suitable operation to "account" for data, both computationally and intuitively, turns out to be the **exclusive or** of two bits, denoted **XOR**.

Definition 2.4. For any two bits x and y, their **XOR**, denoted $x \oplus y$, is simply $x + y \bmod 2$. For a series of bits $x_1 \dots x_n$, $y_1 \dots y_n$, we xor their bits individually to get $(x_1 \oplus y_1) \dots (x_n \oplus y_n)$.

For instance, $1101 \oplus 1011 = 0110$, and $10011 \oplus 11001 = 01010$.

Immediately one may see why computationally **XOR** makes sense: it is one of the simplest operations to implement on a computer. Furthermore, take a look at the table of values for **XOR**:

a	b	$a \oplus b$
0	0	0
1	0	1
0	1	1
1	1	0

[1] There are 3 "unknown" bits, so we need 3 parity "equations" to get a unique solution to the unknowns.

A result of 1 in the **XOR** captures the fact that there is exactly one error between a and b. This property generalizes to **XOR**s of multiple bits too: the value of **XOR** tells us something about errors. We will soon make this more precise.

Question 2.8 (1 pt each)**.** Compute the following:

a) $110011 \oplus 101101$

b) $10111011 \oplus 10111100$.

Question 2.9 (2 pts)**.** Compute a such that $a \oplus (1110111) = (1110110)$.

2.2.2 $[7, 4]$ Hamming Code

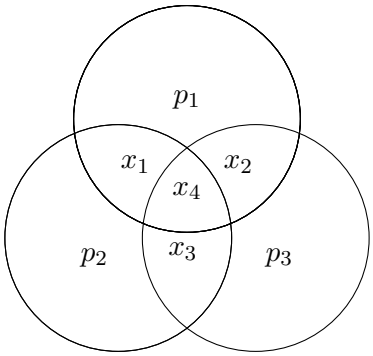
With this inspiration at hand, we can define the $[7, 4]$ Hamming Code.

Definition 2.5. Suppose you have a message $(x_1, x_2, x_3, x_4) \in \Sigma_2^4$. The encoding into [7, 4] Hamming codeword is:

$$E(x_1, x_2, x_3, x_4) = (x_1, x_2, x_3, x_4, p_1, p_2, p_3)$$

where $p_1 = x_2 \oplus x_3 \oplus x_4$, $p_2 = x_1 \oplus x_3 \oplus x_4$, $p_3 = x_1 \oplus x_2 \oplus x_4$.

The first four bits here are the message bits, and the next three are parity bits, where each parity bit encapsulates information about three message bits using **XOR**.



.

Figure 1: Venn Diagram for $[7, 4]$ Hamming Code. Each p_i circle has three message bits within it that it will **XOR** together; intersections show messages bits that are common between the parity bits. For example, p_1 and p_3 share x_2 and x_4 and all of the parity bits contain x_4.

The goal is to now show that this code has distance 3.

Question 2.10 (3 pts)**.** Show that if $\mathbf{x} = (x_1, \ldots x_7)$ and $\mathbf{y} = (y_1, \ldots y_7)$ are two codewords, then $\mathbf{x} + \mathbf{y} \bmod 2$ is also a codeword under the Hamming code. This property is called "additivity."

Question 2.11 (6 pts)**.** Using additivity, prove that for the $[7, 4]$ Hamming code, the minimum distance between any two codewords is the same as the minimum distance from any one nonzero codeword to $\mathbf{0} = (0, 0, \ldots, 0)$. That is, show

$$d(C) = \min_{c \in C, c \neq \mathbf{0}} \delta(c, \mathbf{0}).$$

Hint: $\mathbf{x} - \mathbf{y} = \mathbf{x} + \mathbf{y}$ when adding $\bmod 2$.

The above question reduces our job to showing that the minimum distance of any nonzero codeword from $\mathbf{0}$ is 3.
Let us set up a thought "experiment". Consider the encoded message as seven different lights. A light is on if its corresponding codeword bit is one, and is off otherwise. The lights all being off is the configuration corresponding to the zero codeword. We can only "flip" switches corresponding to the message bits, since the parity bits are derived from them.

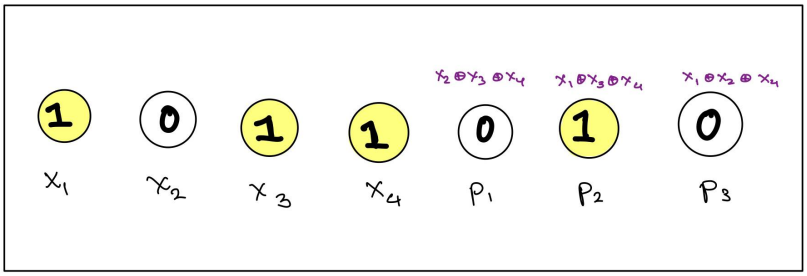


Figure 2: Bulb representation of the message 1011 with Hamming code 1011010

Question 2.12 (6 pts)**.** Show that at least three codeword lights turn on anytime one or two message bits are flipped from the codeword $\mathbf{0}$ in the experiment.

Question 2.13 (4 pts)**.** Prove that $d(C) = 3$.
Hint: First to show that $d(C) \geq 3$, separate into cases of one bit flip, two flips, and three or more flips from the codeword $\mathbf{0}$. Then to establish $d(C) = 3$, find a codeword with distance exactly 3 from $\mathbf{0}$.

Question 2.14 (4 pts)**.** Prove that the $[7, 4]$ Hamming code achieves the Hamming bound.

3 The Singleton Bound and Reed-Solomon Codes (78 pts)

3.1 The Singleton Bound

We now turn back to codes over general alphabets. Now that we've introduced a few codes, a natural question to ask is if the codes are "optimal", or if a "better" code can be found. One way to measure optimality is by the trade-off between distance and *dimension*.

Definition 3.1. Consider a code C with alphabet Σ. Then, the **dimension**[a] of the code is $\log_{|\Sigma|}(|C|)$.

[a]It turns out that for a $[n, k, d]$ code, this is the same as the message length k from before (though you need not understand why for the purposes of the round).

Intuitively, this is the amount of information that the code can send with a single codeword. Our goal is to maximize k while fixing n and d. In general, higher values of d will correspond to lower values of k, as requiring codewords to be a larger distance from one another decreases the number of codewords we can have. The Singleton Bound formalizes this intuition.

Consider a $[n, k, d]$ code C over alphabet Σ.

Question 3.1 (5 pts)**.** Let $c_1, c_2 \in C$ be two distinct codewords. Prove that the $(n - d + 1)$-prefixes of c_1 and c_2 are unique; that is, c_1 and c_2 do not share the same first $n - d + 1$ characters.

Question 3.2 (5 pts)**.** Prove the Singleton Bound[a], which states that $k \leq n - d + 1$.

[a]The Singleton Bound is not related to singleton sets; rather, it is named after Richard Singleton, a mathematician who proved this bound in 1964.

3.2 Reed-Solomon Codes

Reed-Solomon Codes are a class of codes widely used in communication that achieve the Singleton Bound, that is, we have $k = n - d + 1$. The main idea of these codes is to treat our message as numbers modulo some prime p. We encode a message as values of a polynomial with integer coefficients, also taken modulo p.

Definition 3.2. A **polynomial with integer coefficients** $P(x)$ of **degree** D for a variable x is an expression written in the form $P(x) = \sum_{i=0}^{D} a_i x^i$, where $a_0, \ldots, a_D$ are integers and $a_D \neq 0$. We often denote $P(x)$ by P. We define the zero polynomial $P(x) = 0$ to have degree -1.

Definition 3.3. We have two natural operations on polynomials: addition and multiplication. Given polynomials $P(x) = \sum_{i=0}^{D_1} a_i x^i$, and $Q(x) = \sum_{i=0}^{D_2} a_i x^i$, we have $(P + Q)(x) = \sum_{i=0}^{D_1} a_i x^i + \sum_{i=0}^{D_2} a_i x^i$ and $(PQ)(x) = \left(\sum_{i=0}^{D_1} a_i x^i\right) \cdot \left(\sum_{i=0}^{D_2} a_i x^i\right)$

Definition 3.4. Polynomials A and B with integer coefficients are **equivalent modulo** p if $A - B = p \cdot C$, where C is another polynomial with integer coefficients.

Question 3.3 (4 pts)**.** Show that two polynomials with integer coefficients are equivalent modulo p if and only if for any i, the coefficients of x^i for each of the two polynomials are equivalent modulo p.

Definition 3.5. Using the previous problem, we define $P(x)$ as a **polynomial of degree D modulo p** as $P(x) = \sum_{i=0}^{D} a_i x^i$, where $a_0, \ldots, a_D \in \{0, 1, \ldots, p-1\}$ and $a_D \neq 0$. Again, we often denote $P(x)$ by P. We can evaluate P at values $t \in \{0, 1, \ldots, p-1\}$, where the value of $P(t)$ is taken modulo p (i.e. in the range $\{0, 1, \ldots, p-1\}$). When polynomials modulo p are added or multiplied together, we add or multiply them as usual and then take each of the coefficients modulo p.

Definition 3.6. An integer $t \in \{0, 1, \ldots, p-1\}$ is a **root** of $P(t)$ modulo p if $P(t) = 0$.

Definition 3.7. A polynomial P modulo p **divides** a polynomial Q modulo p if there exists a polynomial D modulo p such that $Q = P \cdot D$.

We will state the following theorem, which we will not prove, but you can use in the round:

Theorem 3.1. For any two nonzero polynomials $A(x)$ and $B(x)$ modulo p, there exist polynomials $Q(x)$ and $R(x)$ modulo p such that $A(x) = B(x)Q(x) + R(x)$ and $\deg R < \deg B$.

For the rest of this section, "polynomial" will refer to a polynomial with integer coefficients taken modulo p, and "roots" of polynomial refer to integers in the set $\{0, \ldots, p-1\}$.

First, let's prove some key properties of polynomials. In the next 4 problems, no facts about polynomials may be used other than the definitions and theorem stated above in Section 3.2 and any previous problems.

Consider an integer $D > 0$.

Question 3.4 (3 pts)**.** Find a polynomial $P(x)$ of degree D with distinct roots $x_1, \ldots, x_D$.

Question 3.5 (3 pts)**.** Show that for any two nonzero polynomials $P(x)$ and $Q(x)$, $\deg PQ = \deg P + \deg Q$.

Question 3.6 (5 pts)**.** Show that if t is a root of polynomial $P(x)$, then $x - t$ divides $P(x)$.

Question 3.7 (5 pts)**.** Prove that a degree $D \geq 0$ polynomial cannot have more than D distinct roots.

Question 3.8 (6 pts)**.** Prove that if we have distinct integers $x_1, \ldots, x_D$, and (not necessarily distinct) integers $y_1, \ldots, y_D$, there is at most 1 unique degree $D-1$ polynomial $P(x)$ modulo a prime p such that $P(x_i) = y_i$ for all $1 \leq i \leq D$.

Question 3.9 (8 pts)**.** Reed-Solomon codes require working modulo a prime p to work, because the property in the previous problem only holds for prime numbers. Show that Question 3.8 is false for **any** composite p.

We showed above that there is at most 1 unique degree $D-1$ polynomial modulo p passing through D points. In fact, such a polynomial always exists and can be easily found given the points using a computer[2]. For the rest of the round, you can use without proof that given D points, you can construct such a polynomial (just say "use a computer").
Using these results, we have a new scheme. We choose a prime number p and make a code over the alphabet $\Sigma_p = \{0, 1, 2, \ldots, p-1\}$ as follows.

[2]We will not prove this in this Power Round, but it can be done using a technique known as Lagrange interpolation.

Scheme 3.1. Suppose Alice has a length k message $m_1m_2\cdots m_k$ that she wants to send, where $m_1, \ldots, m_k$ are elements of Σ_p.

- She finds a polynomial P taken modulo p such that $P(i) = m_i$ for all $1 \leq i \leq k$. Note that degree of such a polynomial is $k-1$.
- Alice chooses a number $a \geq k$. Alice sends the a values $P(1)$, $P(2)$, $\ldots$, $P(a)$ along the channel to Bob.
- Bob receives a values, which we denote as $r_1, \cdots, r_a$, where r_i is the result of transmitting $P(i)$ over the channel. He can determine the polynomial $P(x)$ uniquely from these values using a computer.
- Bob plugs in $P(1), P(2), \ldots, P(k)$ to recover the message.

However, suppose we have a noisy channel, so s characters are erased. In other words, out of the $P(1), \ldots, P(a)$ that Alice sends, Bob only receives $b = a - s$ values he can read. How can we ensure that Bob can actually determine the polynomial? Well, since we need k points to uniquely determine a degree $n-1$ polynomial, we must have $b \geq k$ points received, or $a \geq k + s$ points sent.

Question 3.10 (3 pts)**.** Suppose Alice wishes to send the message $(1, 10, 3)$ to Bob such that he will be able to correct up to $s = 2$ erasures. Alice and Bob work modulo $p = 11$. Find the encoded message that Alice sends. Assume that Alice sends the minimum number of values needed for Bob to always be able to decode her message.

Now suppose that instead of characters being erased, we instead have corruptions, where some characters are replaced with other characters. Compared to erasures, Alice will need to send more bits to represent the message; if up to s bits are corrupted, sending $k + s$ values of polynomial $P(x)$ is insufficient to guarantee that we can correctly identify the original polynomial.

Question 3.11 (3 pts)**.** Find an example demonstrating the previous sentence for $k = 2$ and $s = 1$.

However, even if Alice sends more characters to Bob to represent her message, how can Bob decode it? Interpolation will not work, as he will be interpolating a polynomial through corrupted points, which do not lie on the actual polynomial we want to send. To do this, we first define the following:

Definition 3.8. If Alice sends a message m and Bob receives it with characters at positions $e_1, e_2, \ldots, e_s$ corrupted, then the **error-locating polynomial** is defined as $E(x) = (x - e_1)(x - e_2)\cdots(x - e_s)$. For example, if Alice sends the message $(19, 13, 20)$ to Bob, and Bob receives the message $(2, 13, 20)$, then $E(x) = x - 1$.

Bob does not know the error-locating polynomial, as otherwise, he knows the corrupted bits and can simply interpolate the polynomial through the remaining points. However, the error-locating polynomial has a useful property that allows us to decode a corrupted message:

Question 3.12 (5 pts)**.** Recall that r_i is the result of sending $P(i)$ over the channel, which Bob receives. Prove that $P(i)E(i) = r_iE(i)$ for all $1 \leq i \leq a$.

Let $Q(x) = P(x)E(x)$. If we have $a = k + 2s$, the previous problem shows that $Q(i) = r_iE(i)$ for all $1 \leq i \leq k + 2s$.

Question 3.13 (9 pts)**.** Show that we can determine the polynomials Q and E from the $k+2s$ points received. *Hint*: Consider the degrees of the polynomials defined above.

Question 3.14 (2 pts)**.** Given Q and E, how can we recover P?

Question 3.15 (4 pts)**.** Suppose Alice sends a length-3 message (modulo 29) to Bob, but knowing that 1 corruption occurs, she sends an additional 2 characters. Bob receives the encoded message $(2, 6, 12, 17, 1)$. What is the message that Alice sent?

The process described above is known as the Berlekamp-Welch algorithm, named after Elwyn Berlekamp and Lloyd Welch, who published this method in 1983. Finally, we prove that Reed-Solomon codes are optimal with regards to the tradeoff between codeword length and distance.

Question 3.16 (8 pts)**.** Prove that Reed-Solomon codes which correct for s corruptions achieve the Singleton Bound.
Hint: Prove that its minimum distance is $2s + 1$.

4 More Bounds on ECCs (68 pts)

The Singleton Bound proved in the previous section shows that k cannot be too large for a given d. We can ask the opposite question: again given a fixed d, can we construct a code with a given k? The Gilbert-Varshamov Bound is a lower bound on k such that a code of dimension k always exists.

Question 4.1 (4 pts)**.** Find the number of binary codewords of length n with a Hamming distance of less than or equal to r from the length-n **0** codeword, $(0, \ldots, 0)$. You can write your answer as a summation.

Question 4.2 (1 pts)**.** Let A be the answer to the previous question. Find the number of length n codewords with a Hamming distance of less than or equal to r from some given length n codeword x. Express your answer in terms of A.

Question 4.3 (7 pts)**.** Describe a method of finding a binary code C with length n and distance d such that $|C| \geq \frac{2^n}{A}$, where A is the answer to Question 4.1 (letting $r = d$). This bound on the size of the code is known as the Gilbert-Vashamov Bound.

The Gilbert-Varshamov Bound is currently the strongest known result for binary codes (in fact, for any $q-$ary code with $q < 49$[3]) and it is currently an open question as to whether it is tight or if codes with higher rate can be constructed. It can be shown, however, that the GV Bound is only meaningful when $\frac{d}{n} \leq \frac{1}{2}$, and it tells us nothing when $\frac{d}{n} > \frac{1}{2}$. Intuitively, for this case, we can see that if a code C has high distance, it will contain very few codewords. Thus, a more natural question to ask here is whether we can find an upper bound, ideally much tighter than the Singleton Bound, on the number of codewords such a code can contain. The Plotkin Bound below is such a bound.

[3]For $q \geq 49$, a type of code known as Algebraic Geometric Codes exceeds the GV Bound.

Question 4.4 (16 pts)**.** Prove the Plotkin Bound, which states that for a binary code C of length n and distance d such that $d > \frac{n}{2}$ we have $|C| \leq \frac{2d}{2d-n}$.
Hint: Let r_i denote the number of codewords in C with a 1 in position i. Bound the quantity $\sum_{x,y \in C, x \neq y} \delta(x, y)$.

Up to this point, we have dealt with codes where we can receive a corrupted message and correctly decode it. However, this is often very difficult, so one way to simplify this problem is by weakening our constraints to allow us to output more than one decoded message (a "list of messages"), and consider ourselves successful as long as one of the outputted messages is the correct message. This technique is called **list decoding**. Clearly, we need to limit the number of outputted messages, as we can technically list decode any codeword by simply outputting the entire code C. The Johnson Bound tackles this problem by upper bounding the number of codewords in a given radius.

Definition 4.1. The **Johnson radius** of a binary code C of length n and distance $d \leq \frac{n}{2}$ is $J(n, d) = \frac{n - \sqrt{n(n-2d)}}{2}$.

Question 4.5 (25 pts)**.** (Johnson Bound) Show that for a binary code C of length n and distance $d \leq \frac{n}{2}$, there are at most $2n$ codewords in any Hamming ball of radius $J(n, d) - 1$.
Hint: Let r_i denote the number of codewords in the Hamming ball with a 1 in position i. Bound an appropriate quantity and use Cauchy-Schwarz.

Using the Johnson bound, we can prove the Elias-Bassalygo Bound, which is a stronger upper bound on the size of codes than the Hamming Bound.

Question 4.6 (6 pts)**.** (Lemma) Given a binary code C of length n, distance $d \leq \frac{n}{2}$, and some arbitrary $0 \leq r \leq n$, show that there exists a Hamming ball of radius r with at least $\frac{|C| \cdot A}{2^n}$ codewords, where A is the answer to Question 4.2.

Question 4.7 (9 pts)**.** (Elias-Bassalygo Bound) Show that a code of length n and distance $d \leq \frac{n}{2}$ satisfies $|C| \leq \frac{n2^{n+1}}{\binom{n+1}{J(n,d)-1}}$.