1. Let n be a positive integer represented in decimal notation as $a_{k-1}a_{k-2}\ldots a_1a_0$ where $k \geq 1$ and $a_{k-1} \geq 1$. Suppose we want to know whether n is divisible by 11. There is a simple and nice rule. Let n_1 be the sum of the digits of n in odd position and let n_2 be the sum of the digits in the even position. Then n is divisible by 11 iff $n' = |n_1 - n_2|$ is divisible by 11. As an example if $n = 3311$ then $n' = 0$ and n' is divisible by 11. If $n = 383974139871230$ then $n' = |(0+2+7+9+1+7+3+3) - (3+1+8+3+4+9+8)| = 4$ which is not divisible by 11 and hence n is not divisible by 11. If $n = 909090909090909090909090$ then $n' = 108$. To check whether n' is divisible by 11 we can apply the process again to obtain 9 which is not divisible by 11 and conclude that n is not divisible by 11. We thus have a recursive algorithm where we stop the algorithm in the base case of n being a single digit number, otherwise the algorithm computes n' and recurses on n'. We would like to argue about the correctness and running time of this algorithm.

   (a) Prove that as long as n has at least 2 digits, $n' < n$.

   Solution: We prove the following stronger property. For any n represented in decimal as $a_{k-1}a_{k-2}\ldots a_0$ with $k \geq 2$ and $a_{k-1} \geq 1$ we have $n > a_{k-1} + a_{k-2} + \ldots + a_0$. To see this we note that

$$\begin{aligned} n &= a_{k-1}10^{k-1} + a_{k-2}10^{k-2} + \ldots + a_0 \\ &\geq 10a_{k-1} + a_{k-2} + \ldots + a_0 \quad \text{since } k \geq 2 \text{ and digits are } \geq 0 \\ &> a_{k-1} + a_{k-2} + \ldots + a_0 \quad \text{since } a_{k-1} \geq 1. \end{aligned}$$

   Since n_1 and n_2 only take sums of a subset of the digits which are all non-negative, we have $n_1 \leq a_{k-1} + a_{k-2} + \ldots + a_0$ and $n_2 \leq a_{k-1} + a_{k-2} + \ldots + a_0$. Thus $n > n_1$ and $n > n_2$. Since both n_1 and n_2 are non-negative, $n > n_1 - n_2$ and $n > n_2 - n_1$. Therefore $n > |n_1 - n_2|$. ■

   Rubric: 1 point for a correct proof.

   (b) Prove that n is divisible by 11 iff n' is divisible by 11. You can give a direct proof or a proof by induction.

   Solution: We will use some basic facts about modulo arithmetic. Suppose m is a non-negative integer. Then for integers x, y we have $(x+y) \mod m = (x \mod m + y \mod m) \mod m$. Also, $xy \mod m = (x \mod m)(y \mod m) \mod m$.

   We will first prove that for any non-negative integer $i \geq 0$, $10^i \mod 11 = (-1)^i \mod m$. We can easily prove this by induction but we will use the binomial theorem

$$(x+y)^i = \sum_{j=0}^{i} \binom{i}{j} x^{i-j} y^j$$

   Thus,

$$10^i = (11-1)^i = \sum_{j=0}^{i} \binom{i}{j} 11^{i-j} (-1)^j$$

   In the sum on the right hand side every term is a multiple of 11 except the last term which is $(-1)^i$. Thus, $10^i \mod 11 = (-1)^i$. Using this we can compute $n \mod 11$

as follows.

$$\begin{aligned} n \mod 11 &= \left(a_{k-1}10^{k-1} + a_{k-2}10^{k-2} + \ldots + a_0\right) \mod 11 \\ &= \left(a_{k-1}10^{k-1} \mod 11 + a_{k-2}10^{k-2} \mod 11 + \ldots + a_0 \mod 11\right) \mod 11 \\ &= \left(a_{k-1}(-1)^{k-1}) \mod 11 + a_{k-2}(-1)^{k-2} \mod 11 + \ldots + a_0 \mod 11\right) \mod 11 \\ &= (n_2 - n_1) \mod 11. \end{aligned}$$

Note that $n_2 - n_1$ can be negative. From the above it follows that $n \mod 11 = 0$ if and only if $(n_2 - n_1) \mod 11 = 0$ which implies that n is divisible by 11 if and only if $(n_2 - n_1)$ is divisible by 11. Note that $n_2 - n_1$ being divisible by 11 is equivalent to $|n_1 - n_2|$ being divisible by 11. ■

Rubric: 3 points for a correct proof. If an inductive proof is given, this will use the standard induction rubric (see the end of question 2), scaled.

(c) For any positive integer x with decimal representation $b_{k-1}b_{k-2}\ldots b_1b_0$ with $b_{k-1} \geq 1$, prove that $k = O(\log x)$.

Solution: We have $x = \sum_{i=0}^{k-1} b_i 10^i$ with $b_{k-1} \geq 1$. We can upper and lower bound the sum as follows:

$$10^{k-1} \leq x \leq \sum_{i=0}^{k-1} 9 \cdot 10^i \leq 10^k$$

Hence we obtain $k - 1 \leq \log_{10} x \leq k$. Thus $k = \Theta(\log x)$ and in particular $k \leq 1 + \log_{10} x$. ■

Rubric: 1 point for a correct proof.

(d) Using the preceding part show that the decimal representation of n' has $O(\log k)$ digits where k is the number of bits in the representation for n.

Solution: We have $n = \sum_{i=0}^{k-1} a_i 10^i$. Let $x = a_0 + \ldots + a_{k-1}$. We have $x \leq 9k$, so by the previous part the number of digits in the representation of x is at most $O(\log(9k))$ which is $O(\log k)$. Note that $n' \leq x$ and hence the decimal representation of n' has no more digits than that of x.

By being a bit more careful one can in fact see that the value of n' is at most $9 \cdot \frac{k+1}{2}$. This is because $|n_1 - n_2|$ is maximized when all the even-position digits are 9 and all the odd-position digits are 0. Since there are $\lceil \frac{k}{2} \rceil \leq \frac{k+1}{2}$ even positions, this gives the desired bound on n'. Plugging this into the previous part, we get that the number of digits in n' is at most $\log_{10}(9(k+1)/2) + 1 \leq 2 + \log_{10}(k+1)$. ■

Rubric: 1 point for a correct proof. The first paragraph by itself would be sufficient for this.

(e) Assuming that summing ℓ decimal digits to compute the result of the sum in decimal representation takes $O(\ell)$ time, write a recurrence for the total time for the algorithm to terminate as a function of k, the number of digits in the representation of n. Using the recurrence, argue that the running time of the algorithm is $O(k)$.

Solution: The recursive algorithm is simple. Given $n = a_{k-1}a_{k-2}\ldots a_1a_0$ the base case is when $k = 1$; the algorithm outputs yes if $n = a_0 = 0$ otherwise if $a_0 \geq 1$ the answer is no. If $k > 1$ the algorithm computes the decimal representation of n' in $O(k)$ time and recurses on n'. To compute the decimal representation it computes n_1 and n_2 which can be done in $O(k)$ time and subtraction can also be done in $O(k)$ time. To be concrete we will assume that the time outside of the recursive calls is at most ck for some sufficiently large but fixed constant c. Let h be the number of digits in the representation of n'. We saw in the previous part that $h \leq 2 + \log_{10}(k+1)$. We note that $h < k$ for $k \geq 3$ and thus the number of digits is strictly decreasing. For $k = 2$ one can also see that the number of digits strictly decreases since we will be left with a single digit number. Thus we have a proper recursive algorithm. We can write a recurrence

$$T(k) \leq T((2 + \log_{10}(k+1)) + ck$$

with the base case

$$T(k) \leq c', k \leq 2.$$

We will use the following proposition whose proof we postpone for now.

Lemma 1. *For $k \geq 9$, $(2 + \log_{10}(k+1)) \leq k/2$.*

Using the preceding lemma we can rewrite the recurrence as saying $T(k) \leq T(k/2) + ck$ for $k \geq 9$ and $T(k) \leq T(k-1) + ck$ for $k < 8$ with base case of $T(1) = O(1)$. This is easily solved to yield $T(k) = O(k)$ via various simple recurrence analysis techniques. Although this is fine, there may be some worry that we are replacing one recurrence by another. You will get full credit for doing it this way, but for completeness we will next show how to formally prove the bound.

From the recurrence we know that $T(8) \leq c'$ for some fixed constant c' and we choose c'' such that $c'' = \max\{2c, c'\}$. We will prove that $T(k) \leq c''k$ for all $k \geq 1$ by induction on k.

Base case: For $k < 9$ we have $T(k) \leq c'$ by the definition of c' and since $c' \leq c''$ by choice of c'' and $k \geq 1$ we have $T(k) \leq c''k$ for $1 \leq k < 9$.

Induction Hypothesis: Let k be an arbitrary integer such that $k \geq 9$. Assume that for every $k' < k$, we have that $T(k') \leq c''k'$.

Induction step: From the recursion, we have that $T(k) \leq T(h) + ck$ where $h \leq 2 + \log_{10}(k+1)$. From the lemma, $h \leq k/2 < k$ when $k \geq 9$. From the induction hypothesis $T(h) \leq c''h$ which implies that $T(h) \leq c''k/2$. Putting things together:

$$\begin{aligned} T(k) &\leq T(h) + ck \\ &\leq c''k/2 + ck \\ &\leq c''k/2 + c''k/2 \quad \text{since } c'' \geq 2c \\ &\leq c''k. \end{aligned}$$

This finishes the induction step.

We now proof the lemma. Consider the function $f(x) = k - 2(2 + \log_{10}(x+1))$. It suffices to prove that $f(x) \geq 0$ for all $k \geq 9$. We first note that $f(9) = 9 -$

$2(2+\log_{10} 10) = 9 - 6 > 0$. We prove that $f(x)$ is an increasing function in the domain $[9,\infty)$ by considering the derivative $f'(x)$ which is $1 - \frac{2}{\ln 10} \cdot \frac{1}{x+1}$. It is easy to check that $f'(x) > 0$ for all $x \geq 9$. Thus $f(x) > 0$ for $x \geq 9$ which implies that $k/2 \geq (2 + \log_{10}(k+1))$ for all $k \geq 9$. This finishes the proof of the lemma. ■

Rubric: 4 pts. 2pts for writing a correct recursion. -0.5 to -1pts for being sloppy in using bounds in the recursive formula or base cases. 2pts for justifying solution to recurrence.

(f) **Not to submit for grading:** Write a recurrence for the depth of the recursion. It turns out that the answer is $O(\log^* k)$ (you may want to look up and read about this function).

Solution: To compute the depth of the recursion we note that there is only one recursive call in the algorithm that starts with k digits and create a problem with h digits where $h \leq 2 + \log_{10}(k+1)$.

$$D(k) \leq D((2 + \log_{10}(k+1)) + 1 \quad k \geq 3$$

and for $k = 2, 1$ we have $D(k) \leq k$ since the number of digits decreases by one.

$$D(k) \leq k, k \leq 2$$

■

2. Consider the set of strings $L_1 \subseteq \{\mathtt{0}, \mathtt{1}\}^*$ defined recursively as follows:

- The string ε is in L_1.
- For any string x in L_1, the strings $\mathtt{0}x\mathtt{1}$ and the string $\mathtt{1}x\mathtt{0}$ are also in L_1.
- For any two strings $x, y \in L_1$ with $|x|, |y| \geq 1$, the string $xy \in L_1$.
- The only strings in L_1 are the ones generated by the preceding rules.

Now consider the following language L_2 defined recursively as follows.

- The string ε is in L_2.
- For any string x in L_2 and strings a, b with $|a| = |b| = 2$ where ab is a string with equal number of $\mathtt{0}$'s and $\mathtt{1}$'s, the string axb is in L_2.
- For any two strings $x, y \in L_2$ with $|x|, |y| \geq 1$ the string $xy \in L_2$.
- The only strings in L_2 are the ones generated by the preceding rules.

Let L_{eq} be the set of binary strings that have an equal number of $\mathtt{0}$'s and $\mathtt{1}$'s. Let L'_{eq} is the set of strings in L_{eq} whose length is divisible by 4.

(a) Prove by induction that $L_1 \subseteq L_{eq}$.

Solution: This is equivalent to proving the following claim:

Claim 1. *For every string* $w \in L_1$, $\#(\mathtt{0}, w) = \#(\mathtt{1}, w)$.

We will prove this claim by (strong) induction on the length of w.

Base Case: $|w| = 0$. In this case, we must have that $w = \varepsilon$. Since $\#(\mathtt{0}, \varepsilon) = 0 = \#(\mathtt{1}, \varepsilon)$, the statement holds.

Induction Hypothesis: Let $n \geq 1$. Assume that for every $w' \in L_1$ with $|w'| < n$, we have that $\#(\mathtt{0}, w') = \#(\mathtt{1}, w')$.

Induction Step: Let $w \in L_1$ with $|w| = n$. Since $w \in L_1$ and $|w| \geq 1$, we know that w must satisfy the second or third rule in the definition of L_1. We will show that in either case, $\#(\mathtt{0}, w) = \#(\mathtt{1}, w)$.

- Case 1: $w = \mathtt{0}x\mathtt{1}$ for some string $x \in L_1$. Since $|x| = |w| - 2 < n$, we can apply the inductive hypothesis to say that $\#(\mathtt{0}, x) = \#(\mathtt{1}, x)$. But note that $\#(\mathtt{0}, w) = \#(\mathtt{0}, x) + 1$ and $\#(\mathtt{1}, w) = \#(\mathtt{1}, x) + 1$. Thus, we have that $\#(\mathtt{0}, w) = \#(\mathtt{0}, x) + 1 = \#(\mathtt{1}, x) + 1 = \#(\mathtt{1}, w)$ as desired.
- Case 2: $w = \mathtt{1}x\mathtt{0}$ for some string $x \in L_1$. This is identical to the previous case, but with the roles of $\mathtt{0}$ and $\mathtt{1}$ reversed.
- Case 3: $w = xy$ for two strings $x, y \in L_1$ with $|x|, |y| \geq 1$. Since $|x| = |w| - |y| \leq |w| - 1 < n$ and $|y| = |w| - |x| \leq |w| - 1 < n$, we can apply the inductive hypothesis to say that $\#(\mathtt{0}, x) = \#(\mathtt{1}, x)$ and $\#(\mathtt{0}, y) = \#(\mathtt{1}, y)$. But note that $\#(\mathtt{0}, w) = \#(\mathtt{0}, x) + \#(\mathtt{0}, y)$ and $\#(\mathtt{1}, w) = \#(\mathtt{1}, x) + \#(\mathtt{1}, y)$. Thus, we have that $\#(\mathtt{0}, w) = \#(\mathtt{0}, x) + \#(\mathtt{0}, y) = \#(\mathtt{1}, x) + \#(\mathtt{1}, y) = \#(\mathtt{1}, w)$ as desired.

■

Rubric:

- 4 points: Scaled induction rubric (10 point induction rubric at the end of solutions).

(b) Prove by induction that $L_{eq} \subseteq L_1$.

Solution: This is equivalent to the following claim:

Claim 2. *For every string $w \in \{\mathtt{0}, \mathtt{1}\}^*$, if $\#(\mathtt{0}, w) = \#(\mathtt{1}, w)$ then $w \in L_1$.*

As before, we will prove this by (strong) induction on the length of w.

Base Case: $|w| = 0$. This is only possible for $w = \varepsilon$, and by definition $\varepsilon \in L_1$.

Induction Hypothesis: Let $n \geq 1$. Assume that for every $w' \in \{\mathtt{0}, \mathtt{1}\}^*$ with $|w'| < n$, we have that if $\#(\mathtt{0}, w') = \#(\mathtt{1}, w')$ then $w' \in L_1$.

Induction Step: Let $w \in \{\mathtt{0}, \mathtt{1}\}^*$ with $|w| = n$ with $\#(\mathtt{0}, w) = \#(\mathtt{1}, w)$.[1] We want to show that $w \in L_1$. We can write $w = w_1 \dots w_n$ where each $w_i \in \{\mathtt{0}, \mathtt{1}\}$, and consider four possible cases:

- Case 1: $w_1 = \mathtt{0}$ and $w_n = \mathtt{1}$. In this case, we can write $w = 0x1$, where $x = w_2 \dots w_{n-1}$.[2] Since we have that $|x| = |w| - 2 < n$ and $\#(\mathtt{0}, x) = \#(\mathtt{0}, w) - 1 = \#(\mathtt{1}, w) - 1 = \#(\mathtt{1}, x)$, we can apply the induction hypothesis to say that $x \in L_1$. The definition of L_1 then tells us that $w = 0x1$ must also be in L_1 as desired.
- Case 2: $w_1 = \mathtt{1}$ and $w_n = \mathtt{0}$. This is identical to the previous case, but with the roles of $\mathtt{0}$ and $\mathtt{1}$ reversed.

[1] If $\#(\mathtt{0}, w) \neq \#(\mathtt{1}, w)$, the statement is vacuously true!

[2] Note that x may be ε!

- Case 3: $w_1 = w_n = \mathtt{0}$. Consider the function $f : \{1, \ldots, n\} \to \mathbb{Z}$ defined by $f(i) = \#(\mathtt{0}, w_1 \ldots w_i) - \#(\mathtt{1}, w_1 \ldots w_i)$. We note the following properties of this function:
  - $f(1) = 1$. Since $w_1 = \mathtt{0}$, we have that $\#(\mathtt{0}, w_1) = 1$ and $\#(\mathtt{1}, w_1) = 0$.
  - $f(n-1) = -1$. Since $w_n = 0$, we know that $\#(\mathtt{0}, w_1 \ldots w_{n-1}) = \#(\mathtt{0}, w) - 1$ and $\#(\mathtt{1}, w_1 \ldots w_{n-1}) = \#(\mathtt{1}, w)$. Since $\#(\mathtt{0}, w) = \#(\mathtt{1}, w)$, we have that $\#(\mathtt{0}, w_1 \ldots w_{n-1}) = \#(\mathtt{1}, w_1 \ldots w_{n-1}) - 1$.
  - For any $i \in \{1, \ldots, n-1\}$, we have that $f(i+1) - f(i) \in \{1, -1\}$. By rearranging terms, we can write

$$\begin{aligned} f(i+1) - f(i) &= (\#(\mathtt{0}, w_1 \ldots w_{i+1}) - \#(\mathtt{0}, w_1 \ldots w_i)) \\ &\quad - (\#(\mathtt{1}, w_1 \ldots w_{i+1}) - \#(\mathtt{1}, w_1 \ldots w_i)) \\ &= \#(\mathtt{0}, w_{i+1}) - \#(\mathtt{1}, w_{i+1}) \end{aligned}$$

    If $w_{i+1} = \mathtt{0}$, this will be 1; if instead $w_{i+1} = \mathtt{1}$, this will be -1.

  Putting these three facts together, we have that there must be an $i \in \{2, \ldots, n-2\}$ such that $f(i) = 0$. Given such an i, let $x = w_1 \ldots w_i$ and $y = w_{i+1} \ldots w_n$. Since $f(i) = 0$, we know that $\#(\mathtt{0}, x) = \#(\mathtt{1}, x)$. Additionally, $|x| = i < n$, so we can apply the induction hypothesis to say that $x \in L_1$. Similarly, we know that $\#(\mathtt{0}, y) = \#(\mathtt{0}, w) - \#(\mathtt{0}, x) = \#(\mathtt{1}, w) - \#(\mathtt{1}, x) = \#(\mathtt{1}, y)$ and that $|y| = n - i < n$. Thus, the induction hypothesis also tells us that $y \in L_1$. The definition of L_1 then tells us that $w = xy$ is also in L_1, as desired.
- Case 4: $w_1 = w_n = \mathtt{1}$. This is identical to the previous case, but with the roles of $\mathtt{0}$ and $\mathtt{1}$ reversed.

■

Rubric:

- 4 points: Scaled induction rubric (10 point induction rubric at the end of solutions).

(c) Argue that $L'_{eq} \not\subseteq L_2$ by describing a string that is in L'_{eq} but is not in L_2. You should briefly justify why the string you describe is not in L_2.

Solution: Consider the string $w = \mathtt{00011110}$. This string is indeed in L'_{eq}, as $|w| = 8$ and $\#(\mathtt{0}, w) = 4 = \#(\mathtt{1}, w)$. To see why w is not in L_2, we consider each rule in the definition of L_2 in turn and show that w cannot be constructed by it.

- The first rule only adds ε to L_2, and so cannot add w since $w \neq \varepsilon$.
- If we write $w = axb$ for $|a| = |b| = 2$, we must have that $a = \mathtt{00}$ and $b = \mathtt{10}$. But then $\#(\mathtt{0}, ab) = 3 \neq 1 = \#(\mathtt{1}, ab)$. Since the second rule only adds strings when these two are equal, it cannot apply to w.
- Since $L_2 \subseteq L'_{eq}$ (as would formally be proved in the next part), the only possible way to write $w = xy$ for $x, y \in L_2$ with $|x|, |y| \geq 1$ is if $|x| = |y| = 4$. But this would make $x = \mathtt{0001}$ and $y = \mathtt{1110}$, neither of which is in L'_{eq}, meaning neither is in L_2. Thus, the third rule cannot apply to w.

■

Rubric:

- 1 point for a simple correct string in $L'_{eq} - L_2$
- 1 point for a brief justification for why that string is not in L_2

(d) **Not to submit for grading:** Prove by induction that $L_2 \subseteq L'_{eq}$.

Solution: This is equivalent to the following claim:

Claim 3. *For every string $w \in L_2$, $\#(\mathtt{0}, w) = \#(\mathtt{1}, w)$ and $|w|$ is divisible by 4.*

We will prove the claim by (strong) induction on the length of w.

Base Case: $|w| = 0$. This means $w = \varepsilon$. We note that $\#(\mathtt{0}, \varepsilon) = 0 = \#(\mathtt{1}, \varepsilon)$ and that $|\varepsilon| = 0$, which is divisible by 4.

Induction Hypothesis: Let $n > 0$. Assume that for every $w' \in L_2$ with $|w'| < n$, we have that $\#(\mathtt{0}, w') = \#(\mathtt{1}, w')$ and that $|w'|$ is divisible by 4.

Induction Step: Let $w \in L_2$ with $|w| = n$. Since $w \in L_2$ and $|w| > 0$, we know that w must satisfy the second or the third rules in the definition of L_2. We show that in either case, $\#(\mathtt{0}, w) = \#(\mathtt{1}, w)$ and $|w|$ is divisible by 4.

- Case 1: $w = axb$ for some string $x \in L_2$ and strings a, b such that $|a|, |b| = 2$ and ab has an equal number of $\mathtt{0}$ and $\mathtt{1}$. Since $|x| = |w| - 4 < n$, we can apply the induction hypothesis to say that $\#(\mathtt{0}, x) = \#(\mathtt{1}, x)$ and $|x|$ is divisible by 4. Furthermore noting that $\#(\mathtt{0}, ab) = \#(\mathtt{1}, ab) = 2$, we have that $\#(\mathtt{0}, w) = \#(\mathtt{0}, x) + 2 = \#(\mathtt{1}, x) + 2 = \#(\mathtt{1}, w)$. Additionally, $|w| = |x| + 4$, so since $|x|$ is divisible by 4, so is $|w|$. Thus, we have that w satisfies our desired statement.
- Case 2: $w = xy$ for some strings $x, y \in L_2$ with $|x|, |y| \geq 1$. Since $|x| = |w| - |y| \leq n - 1$, we can apply the induction hypothesis to say that $\#(\mathtt{0}, x) = \#(\mathtt{1}, x)$ and $|x|$ is divisible by 4. Similarly, we can say that $\#(\mathtt{0}, y) = \#(\mathtt{1}, y)$ and $|y|$ is divisible by 4. Thus, we have that $\#(\mathtt{0}, w) = \#(\mathtt{0}, x) + \#(\mathtt{0}, y) = \#(\mathtt{1}, x) + \#(\mathtt{1}, y) = \#(\mathtt{1}, w)$ and $|w| = |x| + |y|$ is divisible by 4, as desired.

■

Let $\#(a, w)$ denote the number of times symbol a appears in string w; for example,

$$\#(\mathtt{0}, \mathtt{101110101101011}) = 5 \quad \text{and} \quad \#(\mathtt{1}, \mathtt{101110101101011}) = 10.$$

You may assume without proof that $\#(a, uv) = \#(a, u) + \#(a, v)$ for any symbol a and any strings u and v, or any other result proved in class, in lab, or in the lecture notes. Otherwise, your proofs must be formal and self-contained.

Rubric (induction): For problems worth 10 points:

+ 1 for explicitly considering an *arbitrary* object
+ 2 for a valid induction hypothesis
+ 2 for explicit exhaustive case analysis
  - No credit here if the case analysis omits an infinite number of objects. (For example: all odd-length palindromes.)
  - -1 if the case analysis omits an finite number of objects. (For example: the empty string.)
  - -1 for making the reader infer the case conditions. Spell them out!

- No penalty if cases overlap (for example: even length at least 2, odd length at least 3, and length at most 5.)

+ 1 for cases that do not invoke the inductive hypothesis (“base cases”)
  - No credit here if one or more “base cases” are missing.
+ 2 for correctly applying the *stated* inductive hypothesis
  - No credit here for applying a *different* inductive hypothesis, even if that different inductive hypothesis would be valid.
+ 2 for other details in cases that invoke the inductive hypothesis (“inductive cases”)
  - No credit here if one or more “inductive cases” are missing.