1. (Notation for Lists, 3 × 2 = 6 points)

a. $\phi(L) := \forall i \in \{0, ..., |L| - 1\}.L[|L| - 1 - i] = L[i]$.

The list L is the palindrome list of elements, which means a sequence that has the same backward as forward.

For exlanation, $\forall i \in \{0, ..., |L| - 1\}$ indicates all position index of the elements in the list L, and $L[|L| - 1 - i] = L[i]$ is the condition of the list, where the value of first element equals to the value of the last element. It repeats like that, which means the next element's value of the first element equals to the previous element's value of the last element. Therefore, this is a palindrome list.

b. $\psi(L) := |\{|L[(i + 1)\%|L|] - L[i]||0 \leq i < |L|\}| = 1$.

The list L is the collection of all elements where the differences of absolute values between adjacent elements must be consitent and equaled to 1.

For exlanation, $|L[(i + 1)\%|L|] - L[i]|$ means the diffrence of absolute value between two adjacent elements. $0 \leq i < |L|$ means all elements in the list L. The whole cardinality is checked to be equal to 1, which indicates that the difference of value between adjacent elements must be 1.

c. $\chi(L) := \phi(L) \wedge \psi(L)$.

Reasonable answers include statements like "the list is nonempty and homogeneous" or "the list contains a single element repeated one or more times."
Explanation: Because $\phi(L)$ holds, the first and last elements are equal to each other, so they have a difference of zero. Then, by $\psi(L)$ holding, we know that the difference between any adjacent elements in the list is also zero, so every element must be equal to every other element; the list is homogeneous. Finally, $\psi(L)$ also tells us that the list is nonempty.

2. (Notation for Sets and Graphs, 3 × 2 = 6 points)

a. $\phi((V, E)) := \forall e \in E.\exists v \in e.\{u|\{u, v\} \in E\} = e\backslash\{v\}$.

This is a walk in graph. To be clear, the predicate shows a chain of connected dots, which means if we remove one dot in the graph, the remaining dots on the edge are still connected. Also, it indicates that for each vertex in the graph such that the set of neighboring vertices of that chosen vertex must be equal to the edge after removing it.

For explanation, $\forall e \in E.\exists v \in e$ indicates that for each edge e in the graph E, we need to find a vertex v on that edge. $\{u|\{u, v\} \in E\} = e\backslash\{v\}$ means all nodes u in the graph such that an edge u, v in the graph E, where v is the vertex that we found before, and this set of vertices $\{u|\{u, v\} \in E\}$ should be exactly equal to the edge e after we remove the vertex v.

To illustrate and make the explanation be clearer, here is the example. Assume we have a graph with vertices $V = \{A, B, C\}$ and edges $E = \{\{A, B\}, \{B, C\}, \{C, A\}\}$. We check for edge $\{A, B\}$, and see that the set of neighboring vertices of A, which is B, is equal to the edge $\{A, B\}$ after removing vertex A, which is $\{A, B\}\backslash\{A\} = \{B\}$. This holds true for edges $\{B, C\}$ and $\{C, A\}$ if we choose vertex B and C, respectedly. It also shows that if we remove one dot on the graph, the remaining neighboring dots still be connected to each other, forming a chain of connection, or a walk.

b. $\psi((V,E)) := |\{u|\exists v \in V.\exists \omega \in V.\{\{u,v\},\{v,\omega\},\{\omega,u\}\} \subseteq E\}| \leq 2.$

The predicate shows that it is true if there are at most two vertices in the graph that participate in a triangular relationship formed by three connected vertices, which share edges with two other vertices.

For explanation,$\exists v \in V.\exists \omega \in V$ means there exists a vertex v and vertex ω in the set V.
$\{\{u,v\},\{v,\omega\},\{\omega,u\}\}$ is a set of three distinct edges that form a triangle connecting the vertices u, v, and ω. So, the expression inside the set $\{u|...\}$ means that we look for vertices u such that there exists a pair of vertices v and ω in the graph such that the triangle formed by these three vertices is in the set of edges E. Besides, $\{u|...\} < 2$ indicates that the number of vertices u that satisfies the condition must be at most of 2, which also means that the degree of certain vertices must be less than or equal to 2.

c. If a graph satisfies ϕ, it does not necessarily satisfy ψ, but if a graph satisfies ψ, it necessarily satisfy ϕ.

From part a and b, we see that ϕ shows the linear graph, while ψ shows the triangular graph. To form a line, we only need two vertices, with one undirected edge connected that two vertices. However, we need at least three lines, which means three vertices, and three undirected edges to form a triangle. Thus, if a graph satisfies ψ, it necessarily satisfy ϕ, but it does not necessarily to satisfy ϕ to satisfy ψ.

3. (Summations, 3 x 2 = 6 points)

a.

$$\begin{aligned}
\sum_{i=0}^{n-1}(2\sum_{j=i}^{2i-1} j) &= \sum_{i=0}^{n-1}(2\ \frac{i(i-1)}{2}\Big|_i^{2i}) \\
&= \sum_{i=0}^{n-1}(3i^2 - i) \\
&= 3\sum_{i=0}^{n-1} i^2 - \sum_{i=0}^{n-1} i \\
&= 3\ \frac{i(i-1)(2i-1)}{6}\Big|_0^n - \frac{i(i-1)}{2}\Big|_0^n \\
&= \frac{n(n-1)(2n-1)}{2} - \frac{n(n-1)}{2} \\
&= \frac{(n-1)(n(2n-1)-n)}{2} \\
&= \frac{2n(n-1)(n-1)}{2} \\
&= n(n-1)^2.
\end{aligned}$$

b.

$$\sum_{i=1}^{2\sum_{j=0}^{n-1} j} 2^i$$

Let calculate $2\sum_{j=0}^{n-1} j$ first:

$$2\sum_{j=0}^{n-1} j = j(j-1)|_0^n = n^2 - n.$$

So:

$$\begin{aligned}\sum_{i=1}^{2\sum_{j=0}^{n-1} j} 2^i &= \sum_{i=1}^{n^2-n} 2^i \\ &= \frac{2^i}{2-1}\Big|_1^{n^2-n+1} \\ &= 2^{n^2-n+1} - 2.\end{aligned}$$

c.

$$\begin{aligned}\sum_{i=0}^{n-1}\sum_{j=0}^{n-1}\sum_{k=0}^{i-1}(i+j) &= \sum_{i=0}^{n-1}\sum_{j=0}^{n-1}((i+j)\sum_{k=0}^{i-1}1) \\ &= \sum_{i=0}^{n-1}\sum_{j=0}^{n-1}((i+j)\ k|_0^i) \\ &= \sum_{i=0}^{n-1}\sum_{j=0}^{n-1}(i^2+ij) \\ &= \sum_{i=0}^{n-1}(\sum_{j=0}^{n-1} i^2 + i\sum_{j=0}^{n-1} j) \\ &= \sum_{i=0}^{n-1}(i^2\ j|_0^n + i\ \frac{j(j-1)}{2}\Big|_0^n) \\ &= \sum_{i=0}^{n-1}(i^2 n + \frac{in^2}{2} - \frac{in}{2}) \\ &= n\sum_{i=0}^{n-1} i^2 + \frac{n^2-n}{2}\sum_{i=0}^{n-1} i \\ &= n\ \frac{i(i-1)(2i-1)}{6}\Big|_0^n + \frac{n^2-n}{2}\ \frac{i(i-1)}{2}\Big|_0^n \\ &= n\frac{n(n-1)(2n-1)}{6} + \frac{n^2-n}{2}\frac{n(n-1)}{2} \\ &= \frac{2n^4-3n^3+n^2}{6} + \frac{n^4-2n^3+n^2}{4} \\ &= \frac{7n^4-12n^3+5n^2}{12}.\end{aligned}$$

4. (Recurrences, 2 × 3 = 6 points)

a.

$$T(n) = \begin{cases} 10T(n-1) + 10^n & \text{if } n > 0 \\ 0 & \text{otherwise.} \end{cases}$$

Preparing a table of $T(n)$ for various small values of n:

n	$T(n)$
0	0
1	10
2	200
3	3000
4	40000
5	500000
$\vdots$	$\vdots$

suggest the formula

$$T(n) = n10^n$$

To check this guess, we rewrite the recurrence using another variable name:

$$T(x) = \begin{cases} 10T(x-1) + 10^x & \text{if } x > 0 \\ 0 & \text{otherwise.} \end{cases}$$

Taking $n = x$, the guess becomes $T(x) = x10^x$ which we can use to replace the left-hand side of the recurrence:

$$x10^x = \begin{cases} 10T(n-1) + 10^x & \text{if } x > 0 \\ 0 & \text{otherwise.} \end{cases}$$

Taking $n = x-1$, the guess becomes $T(x-1) = (x-1)10^{x-1}$, which we can use to replace the recursive part of the right-hand side of the recurrence:

$$x10^x = \begin{cases} 10(x-1)10^{x-1} + 10^x & \text{if } x > 0 \\ 0 & \text{otherwise.} \end{cases}$$

Simplifying, we get:

$$x10^x = \begin{cases} x10^x - 10^x + 10^x & \text{if } x > 0 \\ 0 & \text{otherwise.} \end{cases}$$

$$x10^x = \begin{cases} x10^x & \text{if } x > 0 \\ 0 & \text{otherwise.} \end{cases}$$

Because the equality holds for any integral input size $x \geq 0$, the guess is a correct closed form of the recurrence over nonnegative integers.

b.

$$T(n) = \begin{cases} T(n-1) + 2T(n-2) & \text{if } n > 1 \\ n+1 & \text{otherwise.} \end{cases}$$

Preparing a table of $T(n)$ for various small values of n:

n	$T(n)$
0	1
1	2
2	4
3	8
4	16
5	32
$\vdots$	$\vdots$

suggest the formula

$$T(n) = 2^n$$

To check this guess, we rewrite the recurrence using another variable name:

$$T(x) = \begin{cases} T(x-1) + 2T(x-2) & \text{if } x > 1 \\ x + 1 & \text{otherwise.} \end{cases}$$

Taking $n = x$, the guess becomes $T(x) = 2^x$ which we can use to replace the left-hand side of the recurrence:

$$2^x = \begin{cases} T(n-1) + 2T(n-2) & \text{if } x > 1 \\ x + 1 & \text{otherwise.} \end{cases}$$

Taking $n = x - 1$, the guess becomes $T(x-1) = 2^{x-1}$, which we can use to replace the recursive part of the right-hand side of the recurrence:

$$2^x = \begin{cases} 2^{x-1} + 2T(n-2) & \text{if } x > 1 \\ x + 1 & \text{otherwise.} \end{cases}$$

Taking $n = x - 2$, the guess becomes $T(x-2) = 2^{x-2}$, which we can use to replace the recursive part of the right-hand side of the recurrence:

$$2^x = \begin{cases} 2^{x-1} + 2 \times 2^{x-2} & \text{if } x > 1 \\ x + 1 & \text{otherwise.} \end{cases}$$

Simplifying, we get:

$$2^x = \begin{cases} 2^{x-1} + 2^{x-1} & \text{if } x > 1 \\ x + 1 & \text{otherwise.} \end{cases}$$

$$2^x = \begin{cases} 2^x & \text{if } x > 1 \\ x + 1 & \text{otherwise.} \end{cases}$$

Because the equality holds for any integral input size $x \geq 0$, the guess is a correct closed form of the recurrence over nonnegative integers.

5. (Asymptotics, 3 × 2 = 6 points)

a. Let $f(n) = 2^n$ and $g(n) = 2^{2n}$
By using L'Hopital's rule:

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{2^n}{2^{2n}} = \lim_{n\to\infty} \frac{1}{2^n} = 0.$$

Thus, by the limit method, the first expression has a lower order of growth than the second expression.

b. Let $f(n) = n^{2.1} + 10^6$ and $g(n) = n^2 \ln n$
By using L'Hopital's rule:

$$\lim_{n\to\infty} \frac{f(n)}{g(n)} = \lim_{n\to\infty} \frac{n^{2.1} + 10^6}{n^2 \ln n} = \lim_{n\to\infty} \frac{2.1n^{1.1}}{n(2\ln n + 1)} = \lim_{n\to\infty} \frac{2.1n^{0.1}}{2\ln n + 1} = \lim_{n\to\infty} \frac{0.21n^{-0.9}}{\frac{2}{n}} = \lim_{n\to\infty} \frac{0.21n^{0.1}}{2} = \infty.$$

Thus, by the limit method, the first expression has a higher order of growth than the second expression.

c. A plot of the two expression from Part b could mislead a programmmer, as with the small input n, it can mislead that the expression $f(n)$ has a lower order of growth than $g(n)$, while with the large input n, $f(n)$ shows it grows faster than $g(n)$. The thing is that it can not explain exactly the behavior of the functions as input n goes to infinitive, which means n is extremely large. Thus, the expression of growth rate of these two functions, which applies with the large input n, cannnot present and explain by plotting them.

By varying the bound on the plot, it can support to explain the misleading of comparing the growth rate between two functions. This means if we vary the bounds of plots by using the larger input n, it can show the function $f(n)$ grow faster than $g(n)$. For example, we use two functions from Part b $f(n) = n^{2.1} + 10^6$ and $g(n) = n^2 \ln n$, if $n = 100$, it shows that $f(n)$ grows slower than $g(n)$, but $n = 1000$, it shows that $f(n)$ grows faster than $g(n)$.

6. (Iterative Algorithm Analysis, 6 points)

- Basic operation: Assigments
- Input size: Assume that each element takes roughly the same amount of memory where and define $n = |A|$ where $|A|$ is the number of elements in the list. We choose Assignments as the basic operation, so we do not include a positive number b in the input size, because b only appears in Comparisons, and thus it does not impact on the input size. So, $n = |A|$.
- Worst-case time: $\Theta(2n^2 + n + 1)$
- Algorithm analysis:

We count the numbers of step of algorithm. We choose Assignments as the basic operation, so we analyze those lines having Assignment operation, first. When the program runs at the line 1, 5, 10, and 11, it takes 1 step. For line 4, and 9, which are the comparation operation, it takes zero step, so this means when we analyze the lines 4-6, it takes 1 step, and for the lines 9-12, it takes 2 steps. For line 15, it takes zero step.

Line 3 is the looping function, so for the lines 3-7, the program takes $\sum_{j=0}^{n-1} 1$ steps. Line 8 is the looping function, so for the lines 8-13, the program takes $\sum_{j=i}^{n-1} 2$ steps. Therefore, lines 3-13 takes $\sum_{j=0}^{n-1} 1 + \sum_{j=i}^{n-1} 2$ steps.

Line 2 is the loop function, so lines 2-14 takes $\sum_{i=0}^{n-1} (\sum_{j=0}^{n-1} 1 + \sum_{j=i}^{n-1} 2$ steps. Thus, lines 1-15 involves $1 + \sum_{i=0}^{n-1} (\sum_{j=0}^{n-1} 1 + \sum_{j=i}^{n-1} 2)$ steps.

Now, we evaluate the summation of $1 + \sum_{i=0}^{n-1} (\sum_{j=0}^{n-1} 1 + \sum_{j=i}^{n-1} 2)$ to count how many steps that the algorithm takes:

$$
\begin{aligned}
1 + \sum_{i=0}^{n-1}(\sum_{j=0}^{n-1} 1 + \sum_{j=i}^{n-1} 2) &= 1 + \sum_{i=0}^{n-1}(j|_0^n + 2\ j|_i^n) \\
&= 1 + \sum_{i=0}^{n-1}(n + 2(n-1)) \\
&= 1 + \sum_{i=0}^{n-1}(3n - 2i) \\
&= 1 + \sum_{i=0}^{n-1} 3n - \sum_{i=0}^{n-1} 2i \\
&= 1 + 3n\ i|_0^n - 2\ \frac{i(i-1)}{2}\Big|_0^n \\
&= 1 + 3n^2 - n(n-1) \\
&= 2n^2 + n + 1.
\end{aligned}
$$

which is matched with the worst-case time $2n^2 + n + 1$.

7. (Recursive Algorithm Analysis and Master Theorem, 6 points)

- Basic operation: Comparisons.
- Input size: We can measure input size by looking for a quantity that decreases with each recursive call and finding c. so we take $n = c$.
- Worst-case time: $\Theta(n^{1.585})$.

We choose Comparisons as the basic operation, so lines 5-7 ans lines 13-15 take 1 step. Lines 12-16 is the loop operation in which index i loops from 0 to n - 1, that means it takes $1 + \sum_{i=0}^{n-1} 1$. steps.

Because this problem involves an asymptotic analysis, we consider counting additions, which are guaranteed to occur at least once per iteration and at least once per function call on large inputs ($c > 0$).

Line 10 is the recursive operation, so it takes $T(n/2)$ steps. Lines 9-11 is the loop operation, which loops only in three index i from 1 to 3, so the base case takes $3T(n/2)$.

At line 8 and 17, the base case takes zero step. So, lines 8-17 takes $3T(n/2) + 1 + \sum_{i=0}^{n-1} 1$ in total steps when $n > 0$ if we use T to designate steps taken as a function of input size. That simplifies to $3T(n/2) + n + 1$ when we close the summation. Otherwise, when $n = 0$, the base case take 1 addition:

$$
T(n) = \begin{cases} 3T(\frac{n}{2}) + n + 1 & \text{if } n > 0 \\ 1 & \text{otherwise.} \end{cases}
$$

Using the Master Theorem to the large-input case for analyzing the algorithm.
From Master Theorem, we see that: $a = 3$, $b = 2$, and $d_{critical} = \log_2 3$

$$
T(n) = \underbrace{3T(\frac{n}{2})}_{\Theta(n^{\log_2 3})} + \Theta(n)
$$

$$= \Theta(n^{\log_2 3}).$$

Thus, we conclude that: $T(n) = \Theta(n^{\log_2 3}) \approx \Theta(n^{1.585})$.