1.)
a.
(i) Input: n
(ii) Operation: Addition of two numbers
(iii) No additional information

b.
(i) Input: n, the magnitude of n, i.e., the number of bits in its binary representation
(ii) Operation: Multiplication of two integers
(iii) No additional information

c.
(i) Input: n
(ii) Operation: Comparison of two numbers
(iii) No (for the standard list scanning algorithm)

d.
(i) Input: Either the magnitude of the larger of two input numbers, or the magnitude of the smaller of two input numbers, or the sum of the magnitudes of two input numbers
(ii) Operation: Modulo division
(iii) Yes

e.
(i) Input: n, the magnitude of n, i.e., the number of bits in its binary representation
(ii) Operation: Elimination of a number from the list of remaining candidates to be prime
(iii) No additional information

f.
(i) Input: n
(ii) Operation: Multiplication of two digits
(iii) No additional information

4.)
a. The best-case number is two. The worst-case number is twelve: one more than the number of gloves of one-handedness.

b. There are two possible outcomes: the two missing socks make a pair (the best case) and the two missing socks do not (the worst case). The total number of different outcomes (the ways to choose the missing socks) is 45.

7.)
a. T (2n) T (n) ≈ c M 1 3 (2n) 3 c M 1 3 n 3 = 8, where c M is the time of one multiplication.

b. We can estimate the running time for solving systems of order n on the old computer and that of order N on the new computer as T old (n) ≈ c M 1 3 n 3 and T new (N) ≈ 10−3 c M 1 3 N 3, respectively, where c M is the time of one multiplication on the old computer. Replacing T old (n) and T new (N) by these estimates in the equation T old (n) = T new (N) yields c M 1 3 n 3 ≈ 10−3 c M 1 3 N 3 or N n ≈ 10

9.)

a. n (n + 1) ≈ n^2 has the same order of growth (quadratic) as 2000n^2 to within a constant multiple.
b. 100n^2 (quadratic) has a lower order of growth than 0.01n^3 (cubic).
c. Since changing a logarithm's base can be done by the formula log_a(n) = log_b(n) / log_b(a), all logarithmic functions have the same order of growth to within a constant multiple.
d. log_2(2n) = log_2(n) + log_2(n) and log_2(n^2) = 2log_2(n). Hence log_2(2n) has a higher order of growth than log_2(n^2).
e. 2n - 1 = 1 / 2 * 2^n has the same order of growth as 2^n to within a constant multiple.
f. (n - 1)! has a lower order of growth than n! = (n - 1)! * n

10.) : 64^i=12^i−1= 63^j=0^2j=2^64−1≈1.8×10^19.

2.2

2.)
a. n(n+1)/2 is quadratic. Therefore n(n+1)/2 ∈ O(n^3) is true.
b. n(n+1)/2 is quadratic. Therefore n(n+1)/2 ∈ O(n^2) is true.
c. n(n+1)/2 is quadratic. Therefore n(n+1)/2 ∈ Θ(n^3) is false.
d. n(n+1)/2 is quadratic. Therefore n(n+1)/2 ∈ Ω(n) is true.

3.) a. Informally, (n^2 + 1)^10 ≈ (n^2)^10 = n^20 is in Θ(n^20). Formally, as n approaches infinity, (n^2 + 1)^10 / n^20 approaches 1. Hence, (n^2 + 1)^10 is in Θ(n^20).

b. Informally, √(10n^2 + 7n + 3) ≈ √(10n^2) = √10n is in Θ(n). Formally, as n approaches infinity, √(10n^2 + 7n + 3) / n approaches √10. Hence, √(10n^2 + 7n + 3) is in Θ(n).

c. 2n log(n + 2)^2 + (n + 2)^2 log n^2 simplifies to 2n^2 log(n + 2) + (n + 2)^2 (log n - 1) which is in Θ(n log n) + Θ(n^2 log n) = Θ(n^2 log n).

d. 2^n + 1 + 3^(n - 1) = 2^n + 3^(n/3) is in Θ(2^n) + Θ(3^n) = Θ(3^n).

e. Informally, log_2 n ≈ log_2 n is in Θ(log n). Formally, using inequalities x - 1 < x ≤ x, we obtain an upper bound log_2 n ≤ log_2 n and a lower bound log_2 n > log_2 (n - 1) ≥ log_2 (n - 1) > 1/2 log_2 n (for every n ≥ 4). Hence, log_2 n is in Θ(log_2 n) = Θ(log n).

4.)
B. The limit of log 2 n / n is 0.
The limit of n / log 2 n is 0.
The limit of n log 2 n / n^2 is 0.
The limit of n^2 / n^3 is 0.
The limit of n^3 / (2n) is 3 ln 2.
The limit of n^2 / (2n) is 6 ln 2 / 2, which simplifies to 3 ln 2.
The limit of 2n / n! is 0.

5.)$(n-2)! \in \Theta((n-2)!)$
$5\lg(n+100)^{10} = 50\lg(n+100) \in \Theta(\log n)$
$22^n = (22)^n \in \Theta(4^n)$
$0.001n^4 + 3n^3 + 1 \in \Theta(n^4)$
$\ln 2n \in \Theta(\log 2n)$
$3\sqrt{n} \in \Theta(n^{(1/3)})$
$3n \in \Theta(3n)$

Ordered in increasing order of growth:

$5\lg(n+100)^{10}$
$\ln 2n$
$3\sqrt{n}$
$0.001n^4 + 3n^3 + 1$
$3n$
22^n
$(n-2)!$

6.)
a. As n approaches infinity, the limit of p(n) / n^k equals the limit of a_k n^k + a_{k-1} n^{k-1} + ... + a_0 divided by n^k. This simplifies to a_k as n approaches infinity. Hence, p(n) is in $\Theta(n^k)$.

b. As n approaches infinity, the limit of a_n / a_{n+1} equals:

0 if $a_1 < a_2$, which implies a_n is in $o(a_{n+1})$.
1 if $a_1 = a_2$, which implies a_n is in $\Theta(a_{n+1})$.
∞ if $a_1 > a_2$, which implies a_{n+1} is in $o(a_n)$.

7.)
d. The assertion is false. The following pair of functions can serve as a counterexample:
t(n) = n if n is even, n^2 if n is odd
g(n) = n^2 if n is even, n if n is odd

2.3.

2.)a. The sum of (i^2+1)^2 from i=0 to n-1 equals the sum of (i^4+2i^2+1) from i=0 to n-1, which simplifies to the sum of i^4 plus twice the sum of i^2 plus the sum of 1. This is in Theta(n^5) + Theta(n^3) + Theta(n) = Theta(n^5) (or approximately Theta(n^5)).

b. The sum of log_2(i^2) from i=2 to n-1 equals twice the sum of log_2(i) from i=2 to n-1. This is in 2Theta(n log n) - Theta(log n) = Theta(n log n).

c. The sum of (i+1)^(2i-1) from i=1 to n equals the sum of i^(2i) plus the sum of 2^i. This is in Theta(n^(2n)) + Theta(2^n) = Theta(n^(2n)) (or approximately Theta(n^(2n))).

d. The double sum of (i+j) from i=0 to n-1 and j=0 to i equals the sum of i^2 plus (i-1)i. This is in Theta(n^3) - Theta(n^2) = Theta(n^3).

4.)a. Compute the sum S(n) = n * (n + 1) * (2n + 1) / 6.
b. Multiplication (or, if multiplication and addition are assumed to take the same amount of time, either of the two).
c. C(n) = n / n = 1.
d. C(n) = n is in Theta(n). Since the number of bits b = log2(n) + 1 is approximately log2(n) and hence n is approximately 2^b, C(n) is approximately 2^b is in Theta(2^b).
e. Use the formula S(n) = n * (n + 1) * (2n + 1) / 6 to compute the sum in Theta(1) time (which assumes that the time of arithmetic operations stays constant irrespective of the size of the operations' operands).

2.4.

1.) a.) x(n) = x(n-1) + 5 for n > 1, x(1) = 0.
- x(n) = x(n-1) + 5 = [x(n-2) + 5] + 5 = x(n-2) + 5 * 2 = [x(n-3) + 5] + 5 * 2 = x(n-3) + 5 * 3 = ... = x(n-i) + 5 * i = ... = x(1) + 5 * (n-1) = 5(n-1).
Note: The solution can also be obtained by using the formula for the nth term of the arithmetic progression: x(n) = x(1) + d(n-1) = 0 + 5(n-1) = 5(n-1).

b. x(n) = 3x(n-1) for n > 1, x(1) = 4.
- x(n) = 3x(n-1) = 3[3x(n-2)] = 3^2x(n-2) = 3^2[3x(n-3)] = 3^3x(n-3) = ... = 3^ix(n-i) = ... = 3^(n-1)x(1) = 4 * 3^(n-1).
Note: The solution can also be obtained by using the formula for the nth term of the geometric progression: x(n) = x(1) * q^(n-1) = 4 * 3^(n-1).

c. x(n) = x(n-1) + n for n > 0, x(0) = 0.
- x(n) = x(n-1) + n = [x(n-2) + (n-1)] + n = x(n-2) + (n-1) + n = [x(n-3) + (n-2)] + (n-1) + n = x(n-3) + (n-2) + (n-1) + n = ... = x(n-i) + (n-i+1) + (n-i+2) + ... + n = ... = x(0) + 1 + 2 + ... + n = n(n+1)/2.

d. x(n) = x(n/2) + n for n > 1, x(1) = 1 (solve for n = 2^k).
- x(2^k) = x(2^k-1) + 2^k = [x(2^k-2) + 2^(k-1)] + 2^k = x(2^k-2) + 2^(k-1) + 2^k = [x(2^k-3) + 2^(k-2)] + 2^(k-1) + 2^k = x(2^k-3) + 2^(k-2) + 2^(k-1) + 2^k = ... = x(2^k-i) + 2^(k-i+1) + 2^(k-i+2)

+ ... + 2^k = ... = x(2^k-k) + 2^1 + 2^2 + ... + 2^k = 1 + 2^1 + 2^2 + ... + 2^k = 2^(k+1) - 1 = 2 * 2^k - 1 = 2n - 1.

e. x(n) = x(n/3) + 1 for n > 1, x(1) = 1 (solve for n = 3^k).
- x(3^k) = x(3^k-1) + 1 = [x(3^k-2) + 1] + 1 = x(3^k-2) + 2 = [x(3^k-3) + 1] + 2 = x(3^k-3) + 3 = ... = x(3^k-i) + i = ... = x(3^k-k) + k = x(1) + k = 1 + log_3(n).

3.) Let M(n) be the number of multiplications made by the algorithm. We have the following recurrence relation for it: M(n) = M(n-1) + 2, M(1) = 0. We can solve it by backward substitutions: M(n) = M(n-1) + 2 = [M(n-2) + 2] + 2 = M(n-2) + 2 + 2 = [M(n-3) + 2] + 2 + 2 = M(n-3) + 2 + 2 + 2 = ... = M(n-i) + 2i = ... = M(1) + 2(n-1) = 2(n-1).

b. Here is pseudocode for the non-recursive option:
Algorithm NonrecS(n)
// Computes the sum of the first n cubes non-recursively
// Input: A positive integer n
// Output: The sum of the first n cubes.
S ← 1
for i ← 2 to n do
S ← S + i * i * i
return S
The number of multiplications made by this algorithm will be Σ(i=2 to n) 2 = 2 Σ(i=2 to n) 1 = 2(n-1). This is exactly the same number as in the recursive version, but the non-recursive version doesn't carry the time and space overhead associated with the recursion's stack.

4.) a. Q(n) = Q(n-1) + 2n-1 for n > 1, Q(1) = 1. Computing the first few terms of the sequence yields the following: Q(2) = Q(1) + 22 - 1 = 1 + 22 - 1 = 4; Q(3) = Q(2) + 23 - 1 = 4 + 23 - 1 = 9; Q(4) = Q(3) + 24 - 1 = 9 + 24 - 1 = 16. Thus, it appears that Q(n) = n^2. We'll check this hypothesis by substituting this formula into the recurrence equation and the initial condition. The left-hand side yields Q(n) = n^2. The right-hand side yields Q(n-1) + 2n-1 = (n-1)^2 + 2n - 1 = n^2. The initial condition is verified immediately: Q(1) = 1^2 = 1.

b. M(n) = M(n-1) + 1 for n > 1, M(1) = 0. Solving it by backward substitutions (it's almost identical to the factorial example – see Example 1 in the section) or by applying the formula for the nth term of an arithmetical progression yields M(n) = n - 1.

c. Let C(n) be the number of additions and subtractions made by the algorithm. The recurrence for C(n) is C(n) = C(n-1) + 3 for n > 1, C(1) = 0. Solving it by backward substitutions or by applying the formula for the nth term of an arithmetical progression yields C(n) = 3(n - 1). Note: If we don't include in the count the subtractions needed to decrease n, the recurrence will be C(n) = C(n-1) + 2 for n > 1, C(1) = 0. Its solution is C(n) = 2(n - 1).

8.)
a. The algorithm computes the value of the smallest element in a given array.

b. The recurrence for the number of key comparisons is $C(n) = C(n-1) + 1$ for $n > 1$, $C(1) = 0$. Solving it by backward substitutions yields $C(n) = n - 1$.

2.5.

3.) a. The question is to find the smallest value of n such that F(n) is greater than $2^{31} - 1$. Using the formula $F(n) = 1/\sqrt{5}\ \varphi^n$ rounded to the nearest integer, we get (approximately) the following inequality: $1/\sqrt{5}\varphi^n > 2^{31} - 1$ or $\varphi^n > \sqrt{5}(2^{31} - 1)$. After taking natural logarithms of both hand sides, we obtain $n > \ln(\sqrt{5}(2^{31} - 1)) / \ln\varphi \approx 46.3$. Thus, the answer is $n = 47$.

b. Similarly, we have to find the smallest value of n such that F(n) is greater than $2^{63} - 1$. Thus, $1/\sqrt{5}\varphi^n > 2^{63} - 1$, or $\varphi^n > \sqrt{5}(2^{63} - 1)$ or, after taking natural logarithms of both hand sides, $n > \ln(\sqrt{5}(2^{63} - 1)) / \ln\varphi \approx 92.4$. Thus, the answer is $n = 93$.