Question 2.

For this question, solve the recurrence for $T(n)$ and express your answer using Theta (Θ) notation. For each of the cases, assume that $T(m) = \Theta(1)$ for an appropriate constant m. Just state the bounds. No explanation is necessary.

1. **(3 points)** $T(n) = T(\sqrt{n}) + 1$
2. **(3 points)** $T(n) = 4T(n^{1/4}) + \log n$
3. **(3 points)** $T(n) = 2T(n/2) + n \log n$

Question 3.

For this problem, your input is a positive integer n given in its binary encoding.

1. **(3 points)** Design the fastest algorithm that you can, to test if the input n is a perfect square, i.e. if there is a positive integer a such that $a^2 = n$.
2. **(5 points)** Generalize your ideas in the algorithm in Part (1) to design a fast algorithm for testing if n is a perfect power, i.e. if there exist positive integers a, b such that $b > 1$ and $a^b = n$.
3. **(1 point)** As a function of n, what is the number of bits in the input for this problem (in big-Oh notation)?
4. **(1 point)** State the upper bound on the running time of your algorithm in Part(2) of this problem in big-Oh notation. No explanation is necessary.

Question 4.

For this question, we will assume that the cost of multiplying a $u \times v$ matrix with a $v \times w$ matrix is $O(uvw)$, where for *cost* we just count the number of arithmetic operations.

Matrix multiplication is associative, so there are various ways of computing a product of n matrices based on how we parenthesize the matrices. For instance, let $n = 4$. Then the product $A_1 \times A_2 \times A_3 \times A_4$ of four rectangular matrices A_1, A_2, A_3, A_4 of compatible dimensions can be computed in many different ways based on the parenthesization, e.g., as $A_1 \times ((A_2 \times A_3) \times A_4)$, $(A_1 \times (A_2 \times A_3)) \times A_4$, $(A_1 \times A_2) \times (A_3 \times A_4)$ and a few more ways not mentioned here. Each such parenthesization potentially incurrs a different cost based on the dimensions of these matrices. As an example, let the dimension of A_1 be 50×20, that of A_2 be 20×1, that of A_3 be 1×10 and that of A_4 be 10×100. Then, the cost of computing the product $A_1 \times A_2 \times A_3 \times A_4$ using the three parenthesizations stated above are as follows.

- For $A_1 \times ((A_2 \times A_3) \times A_4)$, the cost is $20 \cdot 1 \cdot 10 + 20 \cdot 10 \cdot 100 + 50 \cdot 20 \cdot 100 = 12020$
- For $(A_1 \times (A_2 \times A_3)) \times A_4$, the cost is $20 \cdot 1 \cdot 10 + 50 \cdot 20 \cdot 10 + 50 \cdot 10 \cdot 100 = 60200$
- For $(A_1 \times A_2) \times (A_3 \times A_4)$, the cost is $50 \cdot 20 \cdot 1 + 1 \cdot 10 \cdot 100 + 50 \cdot 1 \cdot 100 = 7000$

Suppose that we are given n (possibly rectangular) matrices $A_1, A_2, \ldots, A_n$ with integer entries where A_i is of dimension $m_{i-1} \times m_i$. Our goal is to design an algorithm to compute the matrix product $A_1 \times A_2 \times \cdots \times A_n$ by finding an optimal way of parenthesizing the matrices .

1. **(6 points)** Design the fastest algorithm that you can to determine the optimal way of computing the product $A_1 \times A_2 \times \cdots \times A_n$, given their dimensions.
2. **(3 points)** Prove the correctness of your algorithm.
3. **(1 point)** State the running time of your algorithm in big-Oh (O) notation (no explanation is necessary).

[redacted]

Question 5.

For this problem, our input is a set of n integers and the goal is to decide if the set contains three elements that sum to zero. For instance, the set $\{1, 2, 3, \ldots, 10\}$ does not contain three elements that sum to zero, whereas the set $\{-100, -2, 1, 20, -5, 7, 1000\}$ contains the elements $\{-2, -5, 7\}$ that sum to zero.

1. **(6 points)** Design an algorithm for this problem that takes $O(n^2)$ arithmetic operations over integers (or the fastest algorithm that you can).

2. **(2 points)** Prove the correctness of the algorithm.

3. **(2 points)** State and prove an upper bound on the running time of your algorithm.