1. Convert the Temperature

```
python
Copy code
def convert_temperature(celsius):
    kelvin = celsius + 273.15
    fahrenheit = celsius * 1.80 + 32.00
    return [round(kelvin, 5), round(fahrenheit, 5)]

# Example usage:
print(convert_temperature(36.50))  # Output: [309.65000, 97.70000]
print(convert_temperature(122.11))  # Output: [395.26000, 251.79800]
```

2. Number of Subarrays With LCM Equal to K

```
python
Copy code
from math import gcd

def lcm(a, b):
    return a * b // gcd(a, b)

def subarrays_with_lcm(nums, k):
    count = 0
    for i in range(len(nums)):
        current_lcm = 1
        for j in range(i, len(nums)):
            current_lcm = lcm(current_lcm, nums[j])
            if current_lcm == k:
                count += 1
            elif current_lcm > k:
                break
    return count

# Example usage:
print(subarrays_with_lcm([3, 6, 2, 7, 1], 6))  # Output: 4
print(subarrays_with_lcm([3], 2))  # Output: 0
```

3. Minimum Number of Operations to Sort a Binary Tree by Level

```
python
Copy code
from collections import deque

def min_operations_to_sort_tree_by_level(root):
    def bfs(root):
        queue = deque([(root, 0)])
        levels = {}
        while queue:
            node, level = queue.popleft()
            if level not in levels:
                levels[level] = []
            levels[level].append(node.val)
            if node.left:
                queue.append((node.left, level + 1))
            if node.right:
                queue.append((node.right, level + 1))
        return levels
```

```
    levels = bfs(root)
    operations = 0
    for level in levels.values():
        sorted_level = sorted(level)
        for i in range(len(level)):
            if level[i] != sorted_level[i]:
                operations += 1
    return operations

# Define the tree nodes and their connections based on the example, then
call the function
# Example usage:
# root = TreeNode(...)
# print(min_operations_to_sort_tree_by_level(root))
```

4. Maximum Number of Non-overlapping Palindrome Substrings

```
python
Copy code
def is_palindrome(s):
    return s == s[::-1]

def max_non_overlapping_palindromes(s, k):
    n = len(s)
    dp = [0] * (n + 1)
    for i in range(n - k + 1):
        if is_palindrome(s[i:i + k]):
            for j in range(i + k, n + 1):
                dp[j] = max(dp[j], dp[i] + 1)
    return max(dp)

# Example usage:
print(max_non_overlapping_palindromes("abaccdbbd", 3))  # Output: 2
print(max_non_overlapping_palindromes("adbcda", 2))  # Output: 0
```

5. Minimum Cost to Buy Apples

```
python
Copy code
import heapq

def min_cost_to_buy_apples(n, roads, appleCost, k):
    graph = [[] for _ in range(n)]
    for a, b, cost in roads:
        graph[a - 1].append((b - 1, cost))
        graph[b - 1].append((a - 1, cost))

    def dijkstra(start):
        pq = [(0, start)]
        distances = [float('inf')] * n
        distances[start] = 0
        while pq:
            dist, node = heapq.heappop(pq)
            if dist > distances[node]:
                continue
            for neighbor, cost in graph[node]:
                new_dist = dist + cost
                if new_dist < distances[neighbor]:
                    distances[neighbor] = new_dist
```

```
                    heapq.heappush(pq, (new_dist, neighbor))
        return distances

    min_costs = [float('inf')] * n
    for i in range(n):
        distances = dijkstra(i)
        min_costs[i] = min(appleCost[j] + distances[j] * (1 if i == j else
1 + k) for j in range(n))

    return min_costs

# Example usage:
print(min_cost_to_buy_apples(4, [[1, 2, 4], [2, 3, 2], [2, 4, 5], [3, 4,
1], [1, 3, 4]], [56, 42, 102, 301], 2))  # Output: [54, 42, 48, 51]
print(min_cost_to_buy_apples(3, [[1, 2, 5], [2, 3, 1], [3, 1, 2]], [2, 3,
1], 3))  # Output: [2, 3, 1]
```

6. Customers With Strictly Increasing Purchases (SQL)

```
sql
Copy code
SELECT customer_id
FROM (
    SELECT customer_id,
           order_date,
           SUM(price) OVER (PARTITION BY customer_id ORDER BY order_date
ROWS BETWEEN UNBOUNDED PRECEDING AND CURRENT ROW) AS cumulative_price,
           LAG(cumulative_price) OVER (PARTITION BY customer_id ORDER BY
order_date) AS previous_price
    FROM Orders
) sub
GROUP BY customer_id
HAVING SUM(CASE WHEN cumulative_price <= previous_price THEN 1 ELSE 0 END)
= 0;
```

7. Number of Unequal Triplets in Array

```
python
Copy code
def unequal_triplets(nums):
    count = 0
    n = len(nums)
    for i in range(n):
        for j in range(i + 1, n):
            for k in range(j + 1, n):
                if nums[i] != nums[j] and nums[i] != nums[k] and nums[j] !=
nums[k]:
                    count += 1
    return count

# Example usage:
print(unequal_triplets([4, 4, 2, 4, 3]))  # Output: 3
print(unequal_triplets([1, 1, 1, 1, 1]))  # Output: 0
```

8. Closest Nodes Queries in a Binary Search Tree

```
python
Copy code
def closest_nodes(root, queries):
```

```
    def in_order_traversal(node):
        if not node:
            return []
        return in_order_traversal(node.left) + [node.val] +
in_order_traversal(node.right)

    sorted_values = in_order_traversal(root)
    answer = []
    for query in queries:
        min_val, max_val = -1, -1
        for val in sorted_values:
            if val <= query:
                min_val = val
            if val >= query:
                max_val = val
                break
        answer.append([min_val, max_val])
    return answer

# Define the tree nodes and their connections based on the example, then
call the function
# Example usage:
# root = TreeNode(...)
# print(closest_nodes(root, [2, 5, 16]))
```

9. Minimum Fuel Cost to Report to the Capital

```
python
Copy code
def min_fuel_cost(roads, seats):
    from collections import defaultdict, deque

    graph = defaultdict(list)
    for u, v in roads:
        graph[u].append(v)
        graph[v].append(u)

    def dfs(node, parent):
        people = 1
        for neighbor in graph[node]:
            if neighbor != parent:
                people += dfs(neighbor, node)
        if node != 0:
            fuel[0] += (people + seats - 1) // seats
        return people

    fuel = [0]
    dfs(0, -1)
    return fuel[0]

# Example usage:
print(min_fuel_cost([[0, 1], [0, 2], [0, 3]], 5))  # Output: 3
print(min_fuel_cost([[3, 1], [3, 2], [1, 0], [0, 4], [0, 5], [4, 6]], 2))
# Output: 7
```

10. Number of Beautiful Partitions

```
python
Copy code
```

```
def count_beautiful_partitions(s, k, minLength):
    primes = {'2', '3', '5', '7'}
    n = len(s)

    def is_prime_digit(c):
        return c in primes

    def is_non_prime_digit(c):
        return c not in primes

    def dfs(start, k):
        if k == 0:
            return 1 if start == n else 0
        if start >= n:
            return 0
        count = 0
        for end in range(start + minLength, n + 1):
            if is_prime_digit(s[start]) and is_non_prime_digit(s[end - 1]):
                count += dfs(end, k - 1)
        return count

    return dfs(0, k)

# Example usage:
print(count_beautiful_partitions("23542185131", 3,
```