Problem 1: Maximum XOR of Two Non-Overlapping Subtrees

```
class Solution:
    def maxXOROfTwoNonOverlappingSubtrees(self, n, edges, values):
        from collections import defaultdict
        graph = defaultdict(list)
        for u, v in edges:
            graph[u].append(v)
            graph[v].append(u)

        def dfs(node, parent):
            subtree_sum[node] = values[node]
            for neighbor in graph[node]:
                if neighbor != parent:
                    subtree_sum[node] += dfs(neighbor, node)
            return subtree_sum[node]

        subtree_sum = [0] * n
        dfs(0, -1)

        max_xor = 0
        for i in range(n):
            for j in range(i+1, n):
                max_xor = max(max_xor, subtree_sum[i] ^ subtree_sum[j])
        return max_xor

# Example Usage
n = 6
edges = [[0, 1], [0, 2], [1, 3], [1, 4], [2, 5]]
values = [2, 8, 3, 6, 2, 5]
sol = Solution()
print(sol.maxXOROfTwoNonOverlappingSubtrees(n, edges, values))  # Output:
24
```

Problem 2: Form a Chemical Bond

```
sql
Copy code
SELECT e1.symbol AS metal, e2.symbol AS nonmetal
FROM Elements e1
JOIN Elements e2
ON e1.type = 'Metal' AND e2.type = 'Nonmetal';
```

Problem 3: Minimum Cuts to Divide a Circle

```
python
Copy code
class Solution:
    def minCuts(self, n):
        if n == 1:
            return 0
        return n if n % 2 == 1 else n // 2

# Example Usage
n = 4
sol = Solution()
print(sol.minCuts(n))  # Output: 2
```

Problem 4: Difference Between Ones and Zeros in Row and Column

```
python
Copy code
class Solution:
    def minPenalty(self, customers):
        n = len(customers)
        y_count = customers.count('Y')
        penalty = y_count
        min_penalty = penalty
        best_hour = 0

        for i in range(n):
            if customers[i] == 'Y':
                penalty -= 1
            else:
                penalty += 1

            if penalty < min_penalty:
                min_penalty = penalty
                best_hour = i + 1

        return best_hour

# Example Usage
customers = "YYNY"
sol = Solution()
print(sol.minPenalty(customers))  # Output: 2
```

Problem 5: Minimum Penalty for a Shop

```
python
Copy code
# Reusing the Solution class and method from Problem 4
```

Problem 6: Count Palindromic Subsequences

```
python
Copy code
class Solution:
    def countPalindromicSubsequences(self, s):
        from functools import lru_cache
        MOD = 10**9 + 7

        @lru_cache(None)
        def count_palindromes(i, j, k):
            if k == 0:
                return 1
            if j - i + 1 < k:
                return 0
            if s[i] == s[j]:
                return count_palindromes(i + 1, j - 1, k - 2) % MOD
            return (count_palindromes(i + 1, j, k) + count_palindromes(i, j
- 1, k) - count_palindromes(i + 1, j - 1, k)) % MOD

        return sum(count_palindromes(i, len(s) - 1, 5) for i in
range(len(s) - 4)) % MOD

# Example Usage
```

```
s = "103301"
sol = Solution()
print(sol.countPalindromicSubsequences(s))  # Output: 2
```

Problem 7: Find the Pivot Integer

```
python
Copy code
class Solution:
    def pivotInteger(self, n):
        total_sum = n * (n + 1) // 2
        current_sum = 0
        for x in range(1, n + 1):
            current_sum += x
            if current_sum == total_sum - current_sum + x:
                return x
        return -1

# Example Usage
n = 8
sol = Solution()
print(sol.pivotInteger(n))  # Output: 6
```

Problem 8: Append Characters to String to Make Subsequence

```
python
Copy code
class Solution:
    def appendCharacters(self, s, t):
        i, j = 0, 0
        while i < len(s) and j < len(t):
            if s[i] == t[j]:
                j += 1
            i += 1
        return len(t) - j

# Example Usage
s = "coaching"
t = "coding"
sol = Solution()
print(sol.appendCharacters(s, t))  # Output: 4
```

Problem 9: Remove Nodes From Linked List

```
python
Copy code
class ListNode:
    def __init__(self, val=0, next=None):
        self.val = val
        self.next = next

class Solution:
    def removeNodes(self, head):
        if not head:
            return head
        stack = []
        while head:
            while stack and stack[-1].val < head.val:
                stack.pop()
            stack.append(head)
            head = head.next
        dummy = ListNode()
        current = dummy
        for node in stack:
            current.next = node
            current = current.next
        current.next = None
        return dummy.next

# Example Usage
head = ListNode(5, ListNode(2, ListNode(13, ListNode(3, ListNode(8)))))
sol = Solution()
result = sol.removeNodes(head)
while result:
    print(result.val, end=' ')
    result = result.next  # Output: 13 8
```

Problem 10: Count Subarrays With Median K

```
python
Copy code
class Solution:
    def countSubarraysWithMedianK(self, nums, k):
        count = 0
        prefix_count = {0: 1}
        balance = 0
        k_index = nums.index(k)

        for i, num in enumerate(nums):
            balance += (1 if num > k else -1)
            if i >= k_index:
                count += prefix_count.get(balance, 0) +
prefix_count.get(balance - 1, 0)
            else:
                prefix_count[balance] = prefix_count.get(balance, 0) + 1

        return count

# Example Usage
nums = [3, 2, 1, 4, 5]
k = 4
sol = Solution()
print(sol.countSubarraysWithMedianK(nums, k))  # Output: 3
```