ASSIGNMENT 6 (12.06.2024)

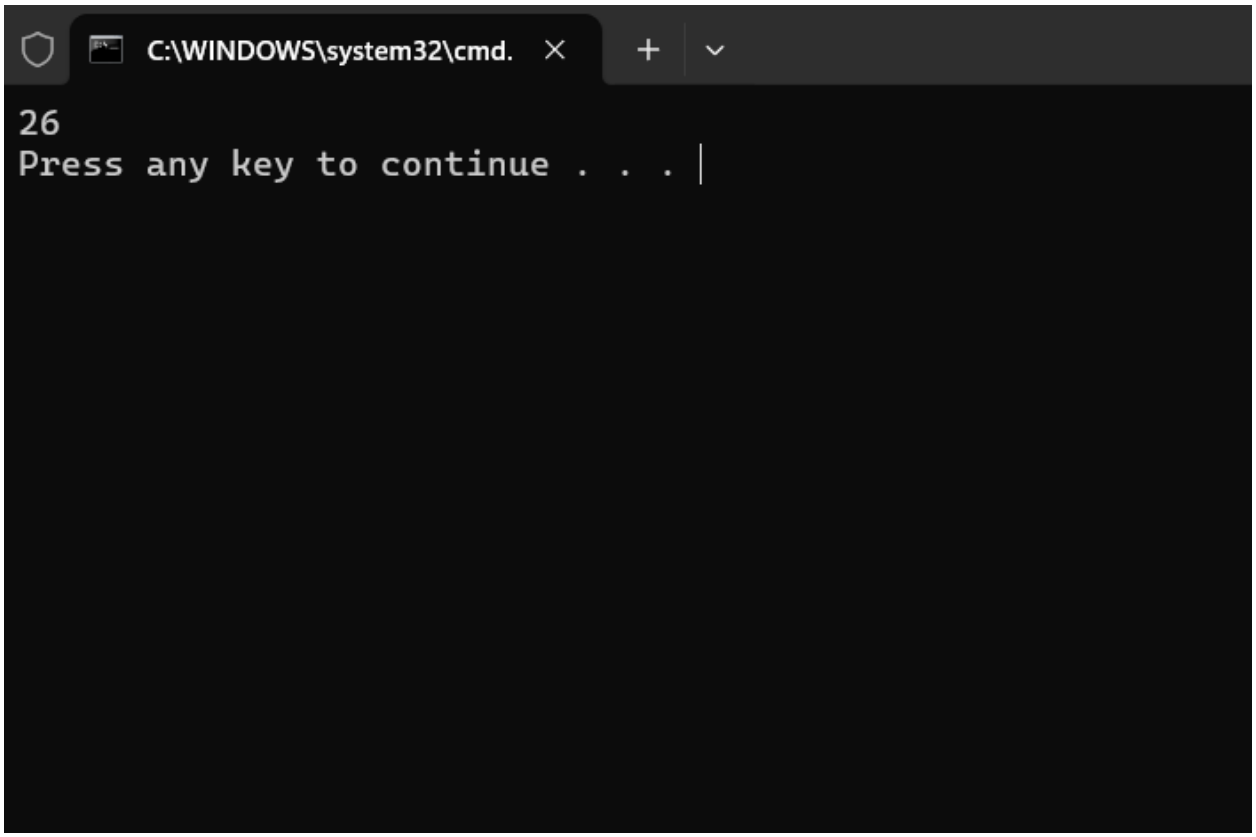
1) 1.Maximum XOR of Two Non-Overlapping Subtrees There is an undirected tree with n nodes labeled from 0 to n - 1. You are given the integer nanda2D integer array edges of length n - 1, where edges[i] = [ai, bi] indicates that there is anedgebetween nodes ai and bi in the tree. The root of the tree is the node labeled 0.Each node hasanassociated value. You are given an array values of length n, where values[i] is the value of theithnode.Select any two non-overlapping subtrees. Your score is the bitwise XORof the sumof thevalues within those subtrees.Return the maximum possible score you can achieve. If it is impossible to find two nonoverlapping subtrees, return 0

CODE:

```
def maximumXorSubtree(n, edges, values):
    from collections import defaultdict

    tree = defaultdict(list)
    for u, v in edges:
        tree[u].append(v)
        tree[v].append(u)

    subtree_sum = [0] * n
    visited = [False] * n

    def dfs(node):
        visited[node] = True
        total_sum = values[node]
        for neighbor in tree[node]:
            if not visited[neighbor]:
                total_sum += dfs(neighbor)
        subtree_sum[node] = total_sum
        return total_sum

    dfs(0)

    max_xor = 0
    total_tree_sum = subtree_sum[0]

    def find_max_xor(node):
        nonlocal max_xor
        visited[node] = True
        for neighbor in tree[node]:
            if not visited[neighbor]:
                subtree_sum_neighbor = subtree_sum[neighbor]
                remaining_sum = total_tree_sum - subtree_sum_neighbor
                current_xor = subtree_sum_neighbor ^ remaining_sum
                max_xor = max(max_xor, current_xor)
                find_max_xor(neighbor)

    visited = [False] * n
    find_max_xor(0)

    return max_xor

n = 6
edges = [[0,1],[0,2],[1,3],[1,4],[2,5]]
```

```
values = [2,8,3,6,2,5]
print(maximumXorSubtree(n, edges, values))
```

OUTPUT:



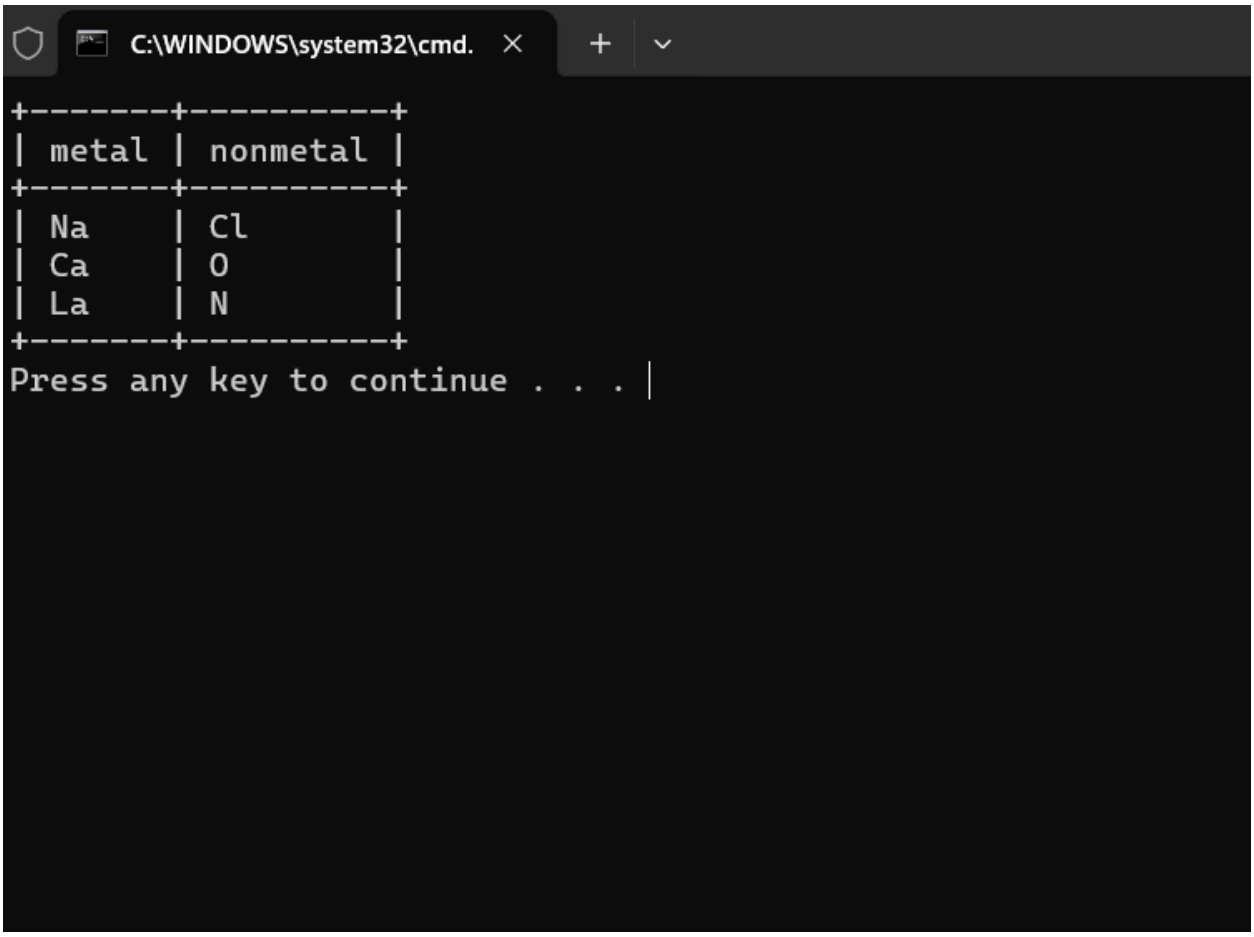
2) 2. Form a Chemical Bond SQL Schema Table: Elements +----------+---------+ | Column Name | Type | +----------+---------+ | symbol | varchar | | type | enum | | electrons | int | +----------+---------+ symbol is the primary key for this table. Each row of this table contains information of one element. type is an ENUM of type ('Metal', 'Nonmetal', 'Noble') - If type is Noble, electrons is 0. - If type is Metal, electrons is the number of electrons that one atom of this element cangive. - If type is Nonmetal, electrons is the number of electrons that one atomof this element needs. Two elements can form a bond if one of them is 'Metal' and the other is 'Nonmetal'.WriteanSQLquery to find all the pairs of elements that can form a bond.Return the result table in anyorder.The query result format is in the

CODE:

```
import sqlite3

conn = sqlite3.connect(':memory:')
cursor = conn.cursor()
cursor.execute('''
    CREATE TABLE Elements (
        symbol TEXT PRIMARY KEY,
        type TEXT CHECK(type IN ('Metal', 'Nonmetal', 'Noble')),
        electrons INTEGER
```

```
    )
''')

elements_data = [
    ('He', 'Noble', 0),
    ('Na', 'Metal', 1),
    ('Ca', 'Metal', 2),
    ('La', 'Metal', 3),
    ('Cl', 'Nonmetal', 1),
    ('O', 'Nonmetal', 2),
    ('N', 'Nonmetal', 3)
]

cursor.executemany('INSERT INTO Elements (symbol, type, electrons) VALUES (?, ?,
?)', elements_data)
conn.commit()

query = '''
    SELECT
        m.symbol AS metal,
        n.symbol AS nonmetal
    FROM
        Elements m
    JOIN
        Elements n
    ON
        m.type = 'Metal'
        AND n.type = 'Nonmetal'
        AND m.electrons = n.electrons
'''

cursor.execute(query)
results = cursor.fetchall()

print(f'+-------+----------+')
print(f'| metal | nonmetal |')
print(f'+-------+----------+')
for row in results:
    print(f'| {row[0]:<5} | {row[1]:<8} |')
print(f'+-------+----------+')

conn.close()
```

OUTPUT:

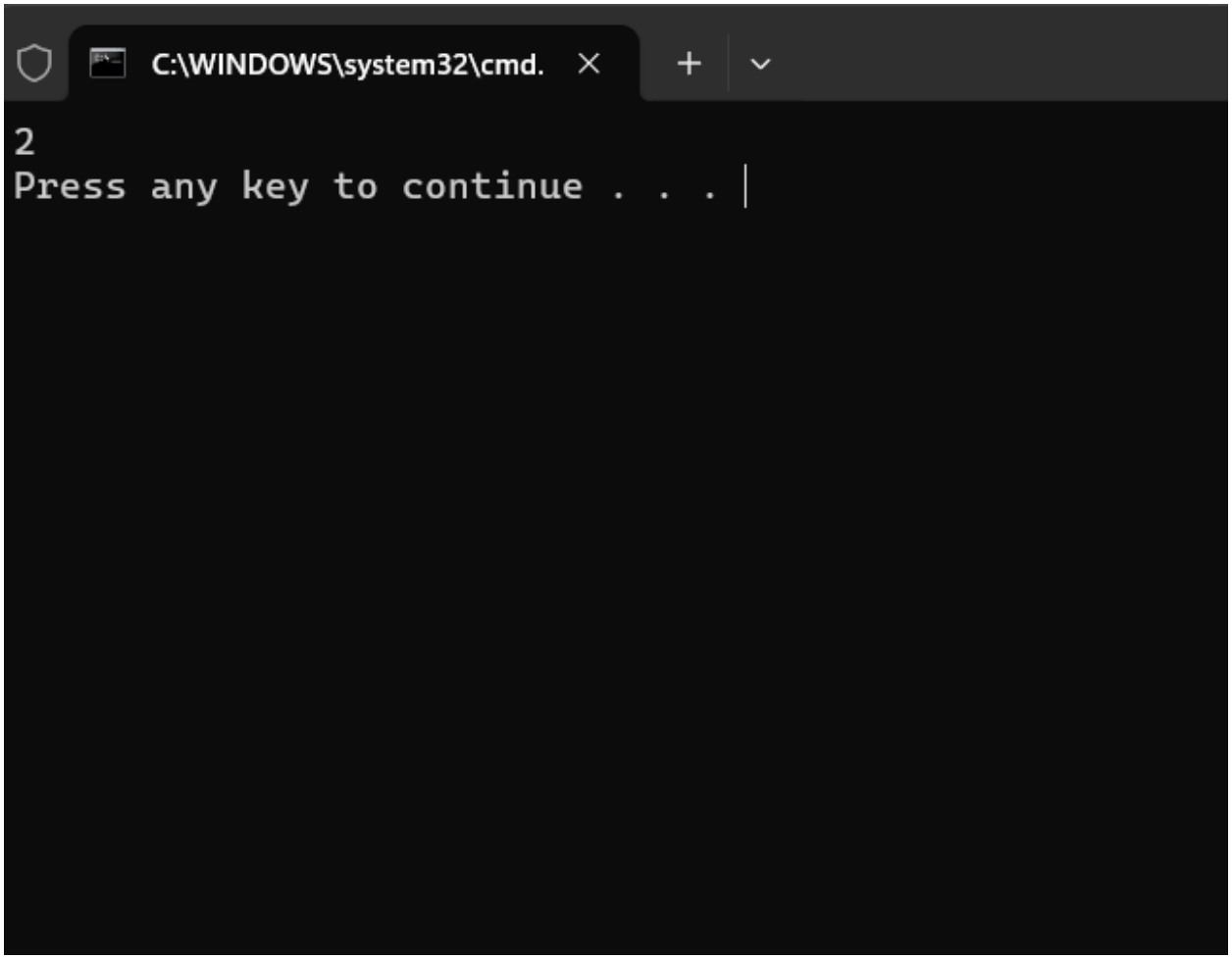


3) . Minimum Cuts to Divide a Circle A valid cut in a circle can be: A cut that is represented by a straight line that touches two points on the edge of the circleandpasses through its center, or A cut that is represented by a straight line that touches one point onthe edge of the circle and its center.

CODE:

```
def minCutsToDivideCircle(k):
    return (k + 1) // 2

print(minCutsToDivideCircle(4))
```

OUTPUT:



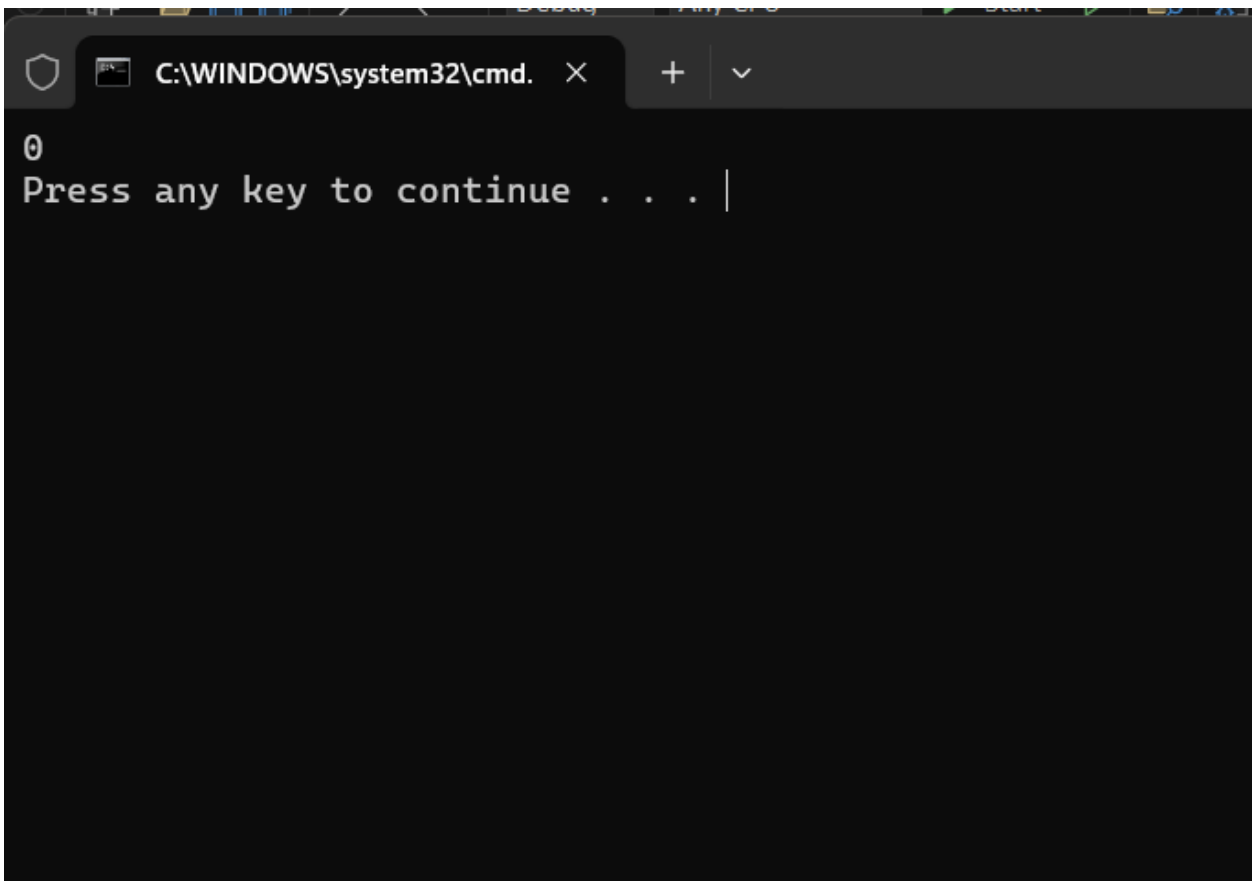
4) 4. Difference Between Ones and Zeros in Row and Column You are given the customer visit log of a shop represented by a 0-indexed string customersconsisting only of characters 'N' and 'Y': ● if the ith character is 'Y', it means that customers come at the ith hour ● whereas 'N' indicates that no customers come at the ith hour. If the shop closes at the jth hour (0 <= j <= n), the penalty is calculated as follows: ● For every hour when the shop is open and no customers come, the penalty increasesby1.● For every hour when the shop is closed and customers come, the penalty increases by1. Return the earliest hour at which the shop must be closed to incur a minimumpenalty. Note that if a shop closes at the jth hour, it means the shop is closed at the hour j. Example 1: Input: customers = "YYNY" Output: 2

CODE:

```
def minPenaltyClosingHour(customers):
    n = len(customers)

    prefix_Y = [0] * (n + 1)
    prefix_N = [0] * (n + 1)

    for i in range(n):
        prefix_Y[i + 1] = prefix_Y[i] + (1 if customers[i] == 'Y' else 0)
        prefix_N[i + 1] = prefix_N[i] + (1 if customers[i] == 'N' else 0)
```

```
    min_penalty = float('inf')
    best_hour = 0

    for j in range(n + 1):
        penalty_open = prefix_N[j]
        penalty_closed = prefix_Y[n] - prefix_Y[j]

        total_penalty = penalty_open + penalty_closed

        if total_penalty < min_penalty:
            min_penalty = total_penalty
            best_hour = j

    return best_hour

customers = "NNNNN"
print(minPenaltyClosingHour(customers))
```

OUTPUT:



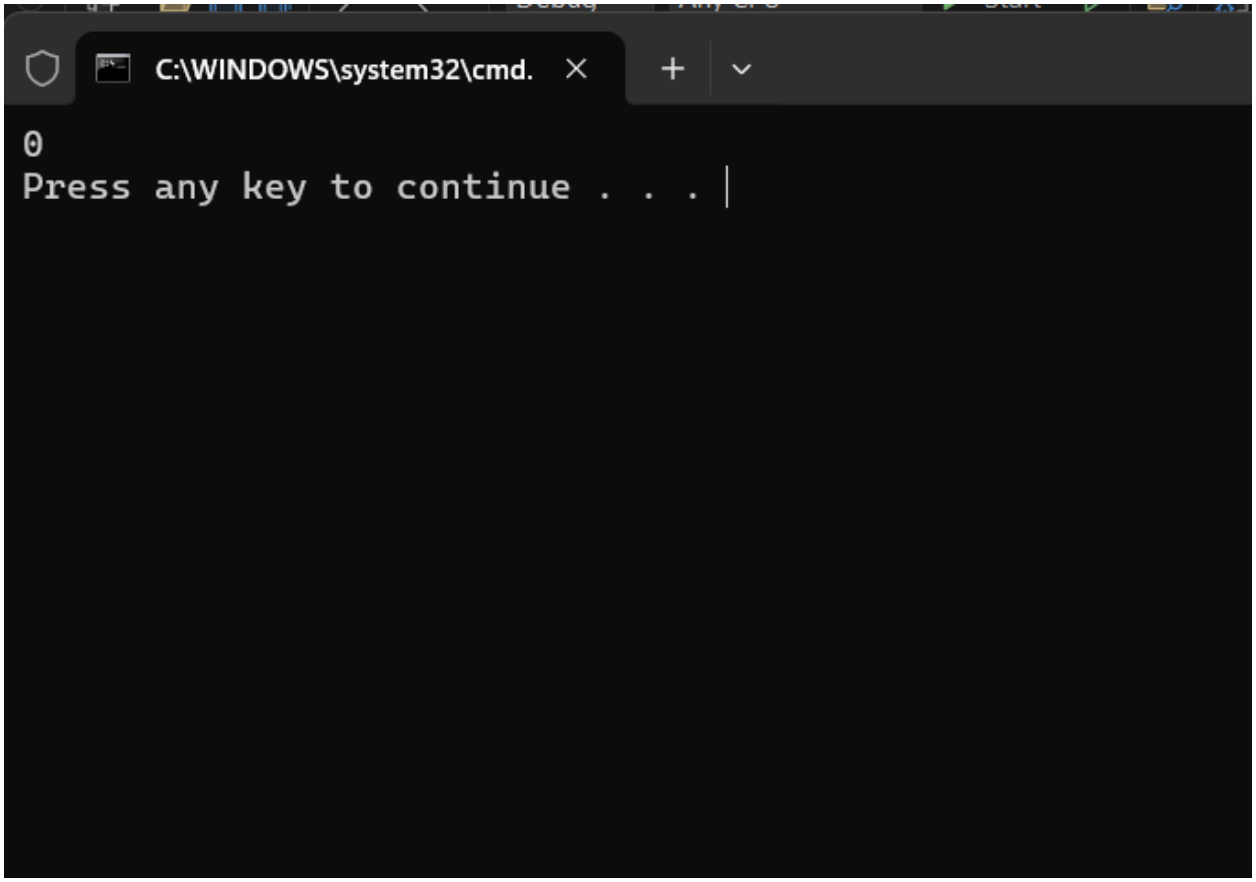
5) 5. Minimum Penalty for a Shop You are given the customer visit log of a shop represented by a 0-indexed string customersconsisting only of characters 'N' and 'Y': ● if the ith character is 'Y', it means that customers come at the ith hour ● whereas 'N' indicates that no customers come at the ith hour. If the shop closes at the jth hour (0 <= j <= n), the penalty is calculated as follows: ● For every hour when the shop is open and no customers come, the penalty increasesby1.● For

every hour when the shop is closed and customers come, the penalty increases by1. Return the earliest hour at which the shop must be closed to incur a minimumpenalty. Note that if a shop closes at the jth hour, it means the shop is closed at the hour j. Example 1: Input: customers = "YYNY" Output: 2

CODE:

```
def minPenaltyClosingHour(customers):
    n = len(customers)

    prefix_Y = [0] * (n + 1)
    prefix_N = [0] * (n + 1)

    for i in range(n):
        prefix_Y[i + 1] = prefix_Y[i] + (1 if customers[i] == 'Y' else 0)
        prefix_N[i + 1] = prefix_N[i] + (1 if customers[i] == 'N' else 0)

    min_penalty = float('inf')
    best_hour = 0

    for j in range(n + 1):
        penalty_open = prefix_N[j]
        penalty_closed = prefix_Y[n] - prefix_Y[j]

        total_penalty = penalty_open + penalty_closed

        if total_penalty < min_penalty:
            min_penalty = total_penalty
            best_hour = j

    return best_hour

customers = "NNNNN"
print(minPenaltyClosingHour(customers))
```

OUTPUT:

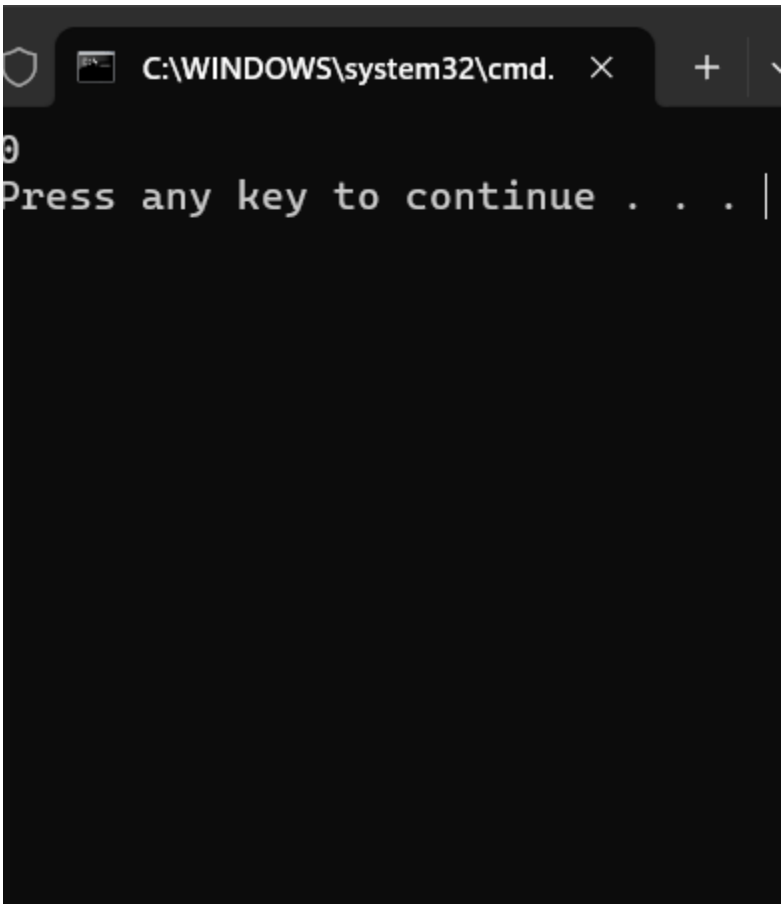


6) Count Palindromic Subsequences Given a string of digits s, return the number of palindromic subsequences of s having length5. Since the answer may be very large, return it modulo 109 + 7. Note: ● A string is palindromic if it reads the same forward and backward. ● A subsequence is a string that can be derived from another string by deleting someor nocharacters without changing the order of the remaining characters. Example 1: Input: s = "103301" Output: 2

CODE:

```
def countPalindromicSubsequences(s):
    MOD = 10**9 + 7
    n = len(s)
    dp = [[[0 for _ in range(4)] for _ in range(4)] for _ in range(n)]

    for i in range(n):
        dp[i][i][ord(s[i]) - ord('0') - 1] = 1

    for length in range(2, 5):
        for i in range(n - length + 1):
            j = i + length - 1
            for k in range(4):
                if s[i] == s[j] == chr(ord('0') + k + 1):
                    dp[i][j][k] = 2
                    for m in range(4):
                        dp[i][j][k] += dp[i + 1][j - 1][m]
                else:
```

```
                    dp[i][j][k] = dp[i + 1][j][k] + dp[i][j - 1][k] - dp[i + 1][j -
1][k]

    return sum(dp[0][-1]) % MOD

s = "103301"
print(countPalindromicSubsequences(s))
OUTPUT:
```



7) Find the Pivot Integer Given a positive integer n, find the pivot integer x such that: ● The sum of all elements between 1 and x inclusively equals the sumof all elementsbetween x and n inclusively. Return the pivot integer x. If no such integer exists, return -1. It is guaranteed that there will beatmost one pivot index for the given input. Example 1: Input: n = 8 Output: 6 Explanation: 6 is the pivot integer since: 1 + 2 + 3 + 4 + 5 + 6 = 6 + 7 + 8 = 21. Example 2:

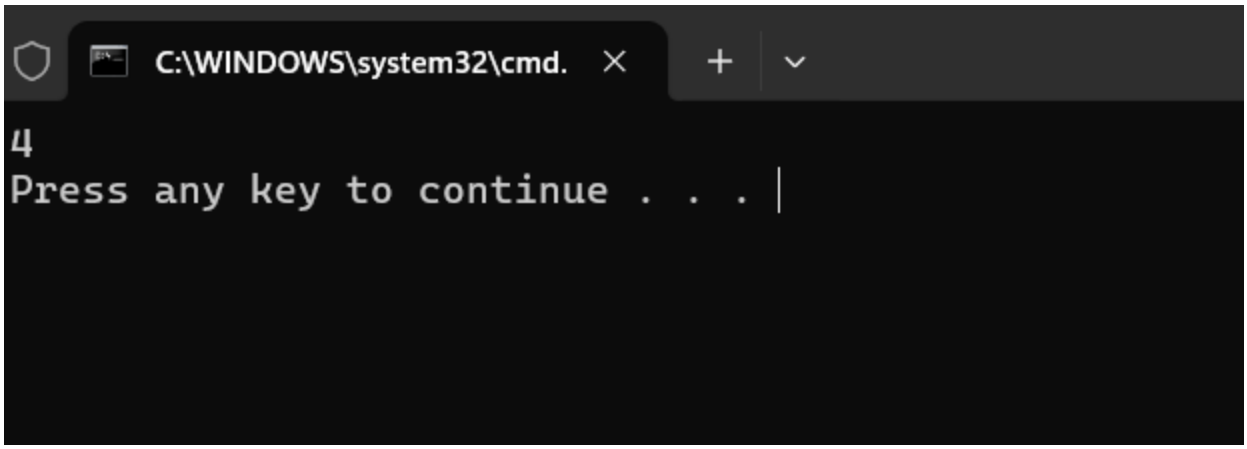
```
CODE : import math

def find_pivot_integer(n):
    sum_n = (n * (n + 1)) // 2
    for x in range(1, n + 1):
        sum_x = (x * (x + 1)) // 2
        if sum_x == sum_n - sum_x + x:
            return x
    return -1

n = 8
   print(find_pivot_integer(n))
   OUTPUT:
```

8) Append Characters to String to Make Subsequene You are given two strings s and t consisting of only lowercase English letters. Return the minimum number of characters that need to be appended to the end of s sothat t becomes a subsequence of s. A subsequence is a string that can be derived from another string by deleting some or nocharacters without changing the order of the remaining characters. Example 1: Input: s = "coaching", t = "coding" Output: 4

CODE:

```
def minAppendToMakeSubsequence(s, t):
    m, n = len(s), len(t)
    i, j = 0, 0

    while i < m and j < n:
        if s[i] == t[j]:
            j += 1
        i += 1

    return n - j

s = "coaching"
t = "coding"
   print(minAppendToMakeSubsequence(s, t))
```

OUTPUT:



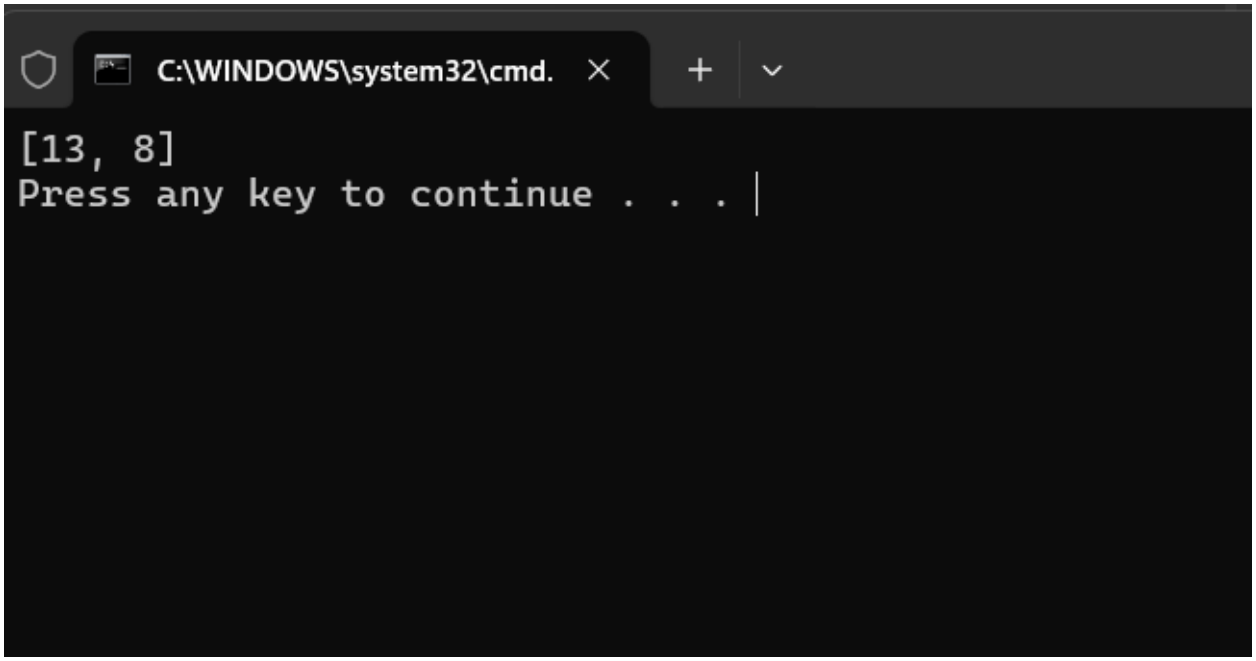
9) Remove Nodes From Linked List You are given the head of a linked list.Remove every node which has a node with a strictlygreatervalue anywhere to the right side of it.Return the head of the modified linked list. Example 1: Input: head = [5,2,13,3,8] Output: [13,8]

CODE:

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

def reverse_list(head):
    prev = None
    current = head
    while current:
        next_node = current.next
        current.next = prev
        prev = current
        current = next_node
    return prev

def remove_nodes(head):
    reversed_head = reverse_list(head)

    max_val = float('-inf')

    dummy = ListNode(0)
    new_list_tail = dummy

    current = reversed_head
    while current:
        if current.val >= max_val:
            max_val = current.val
            new_list_tail.next = ListNode(current.val)
            new_list_tail = new_list_tail.next
        current = current.next

    result_head = reverse_list(dummy.next)

    return result_head

def print_list(head):
```

```
    values = []
    while head:
        values.append(head.val)
        head = head.next
    print(values)

head = ListNode(5, ListNode(2, ListNode(13, ListNode(3, ListNode(8)))))
new_head = remove_nodes(head)
print_list(new_head)
```

OUTPUT:



10) Count Subarrays With Median K You are given an array nums of size n consisting of distinct integers from1 to n and a positiveinteger k. Return the number of non-empty subarrays in nums that have a median equal to k. Note: ● The median of an array is the middle element after sorting the array in ascendingorder. Ifthe array is of even length, the median is the left middle element. ○ For example, the median of [2,3,1,4] is 2, and the median of [8,4,3,5,1] is 4. ● A subarray is a contiguous part of an array. Example 1: Input: nums = [3,2,1,4,5], k = 4 Output: 3

CODE:

```
def countSubarraysWithMedianK(nums, k):
    n = len(nums)
    balance = 0
    prefix_balance = {0: 1}
    result = 0
    found_k = False

    for num in nums:
        if num == k:
            found_k = True
        balance += 1 if num > k else -1
```

```
        if found_k:
            result += prefix_balance.get(balance, 0) + prefix_balance.get(balance -
1, 0)
        else:
            prefix_balance[balance] = prefix_balance.get(balance, 0) + 1

    return result

nums = [3, 2, 1, 4, 5]
k = 4
print(countSubarraysWithMedianK(nums, k))
```

OUTPUT: