

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 an edge between nodes ai and bi in the tree. The root of the tree is the node labeled 0 . Each node has an associated value. You are given an array `values` of length n , where `values[i]` is the value of the i th node. Select any two non-overlapping subtrees. Your score is the bitwise XOR of the sum of the values within those subtrees. Return the maximum possible score you can achieve. If it is impossible to find two nonoverlapping subtrees,

return 0

PROGRAM:

```
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)

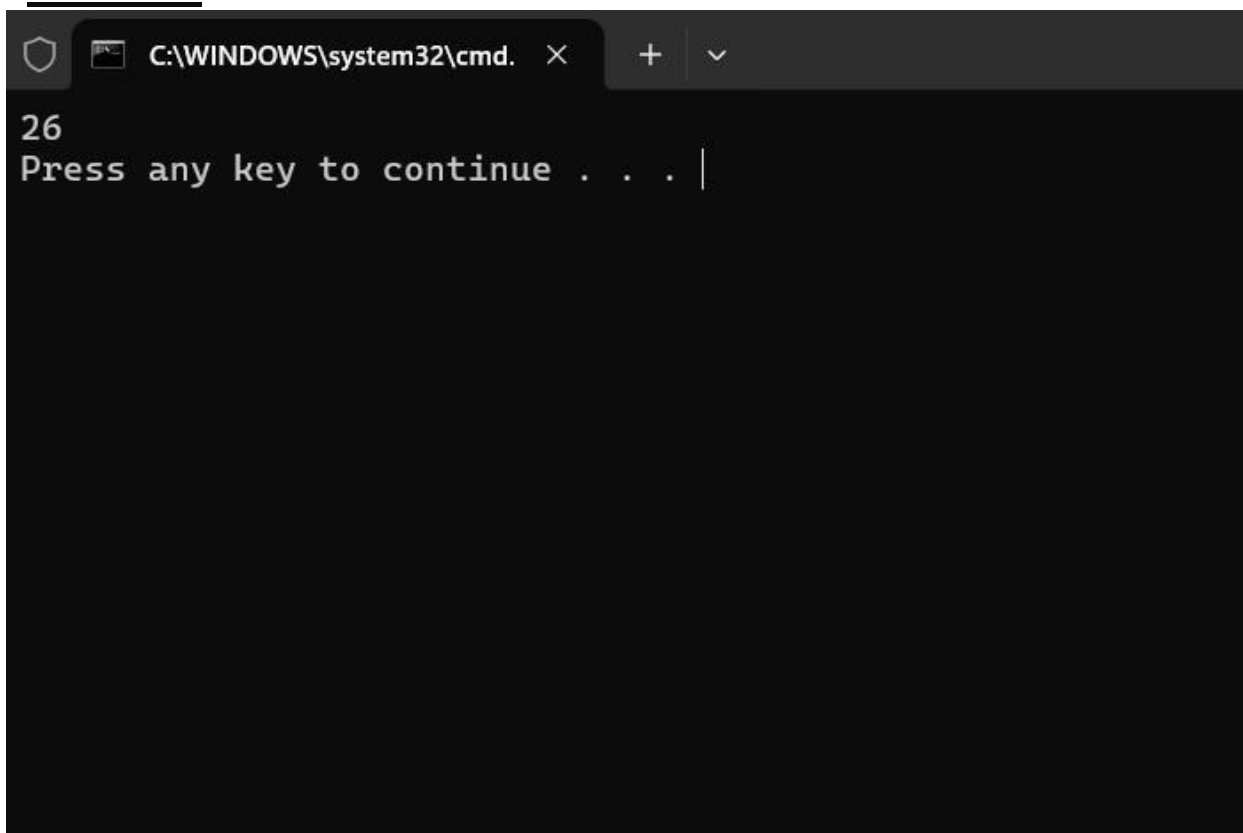
    visited = [False] * n    find_max_xor(0)

    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:



```
C:\WINDOWS\system32\cmd. 26
Press any key to continue . . . |
```

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 import sqlite3

PROGRAM:

```
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:

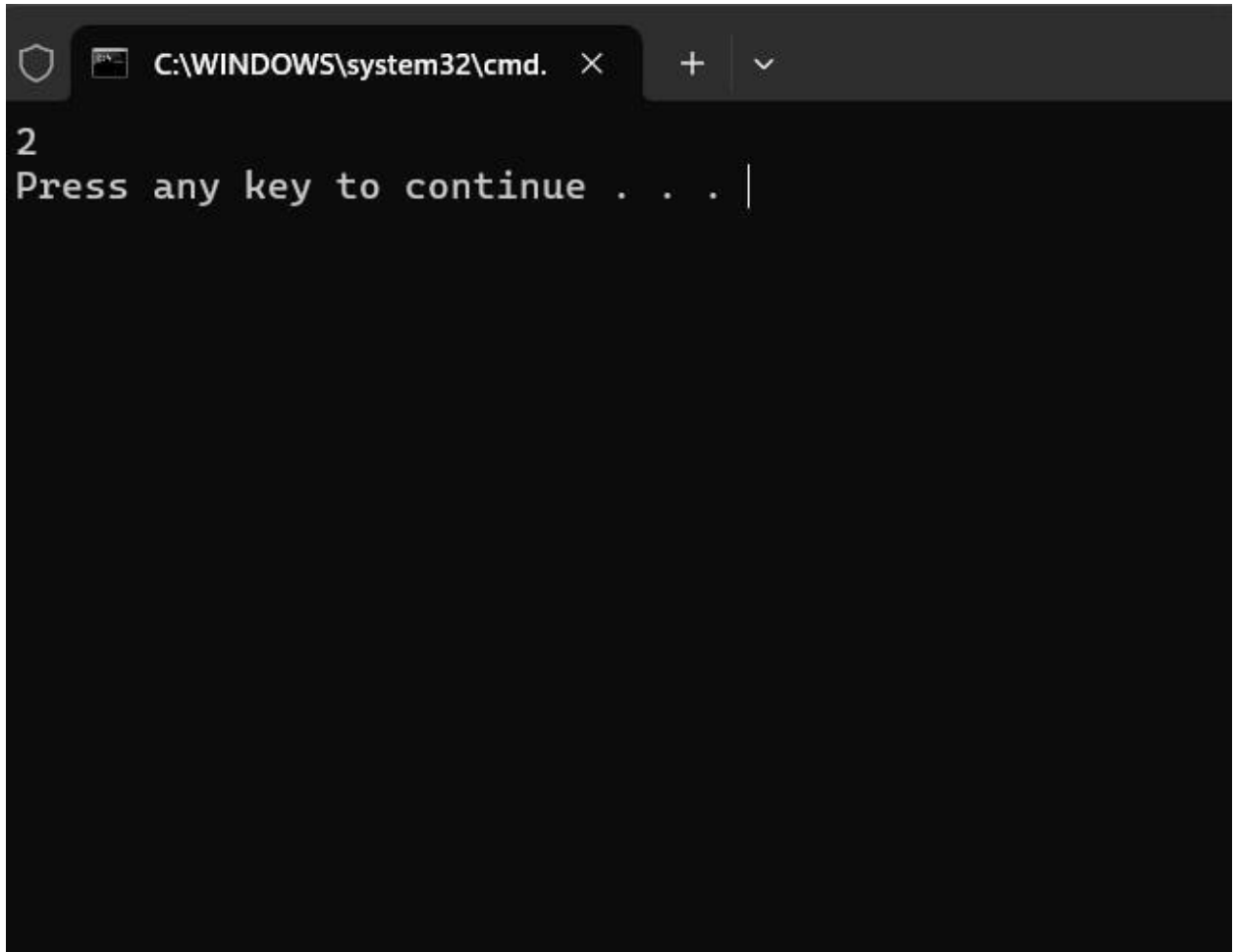
```
C:\WINDOWS\system32\cmd. X + v
+-----+-----+
| metal | nonmetal |
+-----+-----+
| Na    | Cl      |
| Ca    | O       |
| La    | N       |
+-----+-----+
Press any key to continue . . . |
```

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 circle and passes through its center, or A cut that is represented by a straight line that touches one point on the edge of the circle and its center.

PROGRAM:

```
Def
minCutsToDivideCircle(k)
:    return (k + 1) // 2
print(minCutsToDivideCircle(4))
```

OUTPUT:



```
C:\WINDOWS\system32\cmd.  X  +  v
2
Press any key to continue . . . |
```

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 `customers` consisting only of characters 'N' and 'Y':

- if the i th character is 'Y', it means that customers come at the i th hour
- whereas 'N' indicates that no customers come at the i th hour.

If the shop closes at the j th hour ($0 \leq j \leq n$), the penalty is calculated as follows:

- For every hour when the shop is open and no customers come, the penalty increases by 1.
- For every hour when the shop is closed and customers come, the penalty increases by 1.

Return the earliest hour at which the shop must be closed to incur a minimum penalty. Note that if a shop closes at the j th hour, it

means the shop is closed at the hour j. Example 1: Input:
customers = "YYNY" Output: 2

PROGRAM:

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

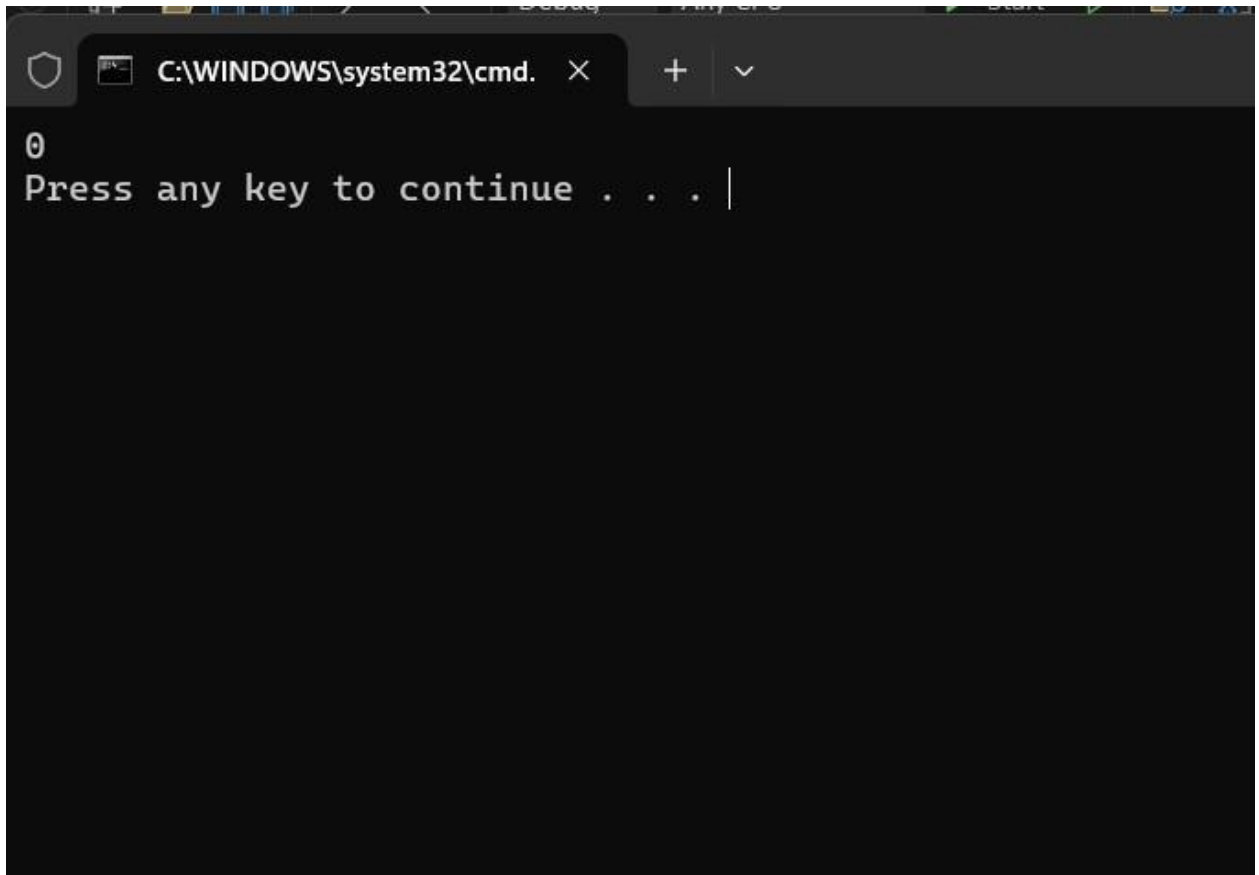
    prefix_Y = [0] * (n + 1)
    prefix_N = [0] * (n + 1)
    for i in range(n):    prefix_Y[i + 1] = prefix_Y[i] + (1 if customers[i] ==
'Y'
else 0)    prefix_N[i + 1] = prefix_N[i] + (1 if customers[i] == 'N' else 0)
    min_penalty = float('inf')
    best_hour = 0
    for j in range(n + 1):
        penalty_open =
        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:



```
0
Press any key to continue . . . |
```

- 5. Minimum Penalty for a Shop** You are given the customer visit log of a shop represented by a 0indexed string **customers** consisting only of characters 'N' and 'Y':
- if the i th character is 'Y', it means that customers come at the i th hour
 - whereas 'N' indicates that no customers come at the i th hour.
- If the shop closes at the j th hour ($0 \leq j \leq n$), the penalty is calculated as follows:
- For every hour when the shop is open and no customers come, the penalty increases by 1.
 - For every hour when the shop is closed and customers come, the penalty increases by 1.
- Return the earliest hour at which the shop must be closed to incur a minimum penalty.

**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**

PROGRAM:

```
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)

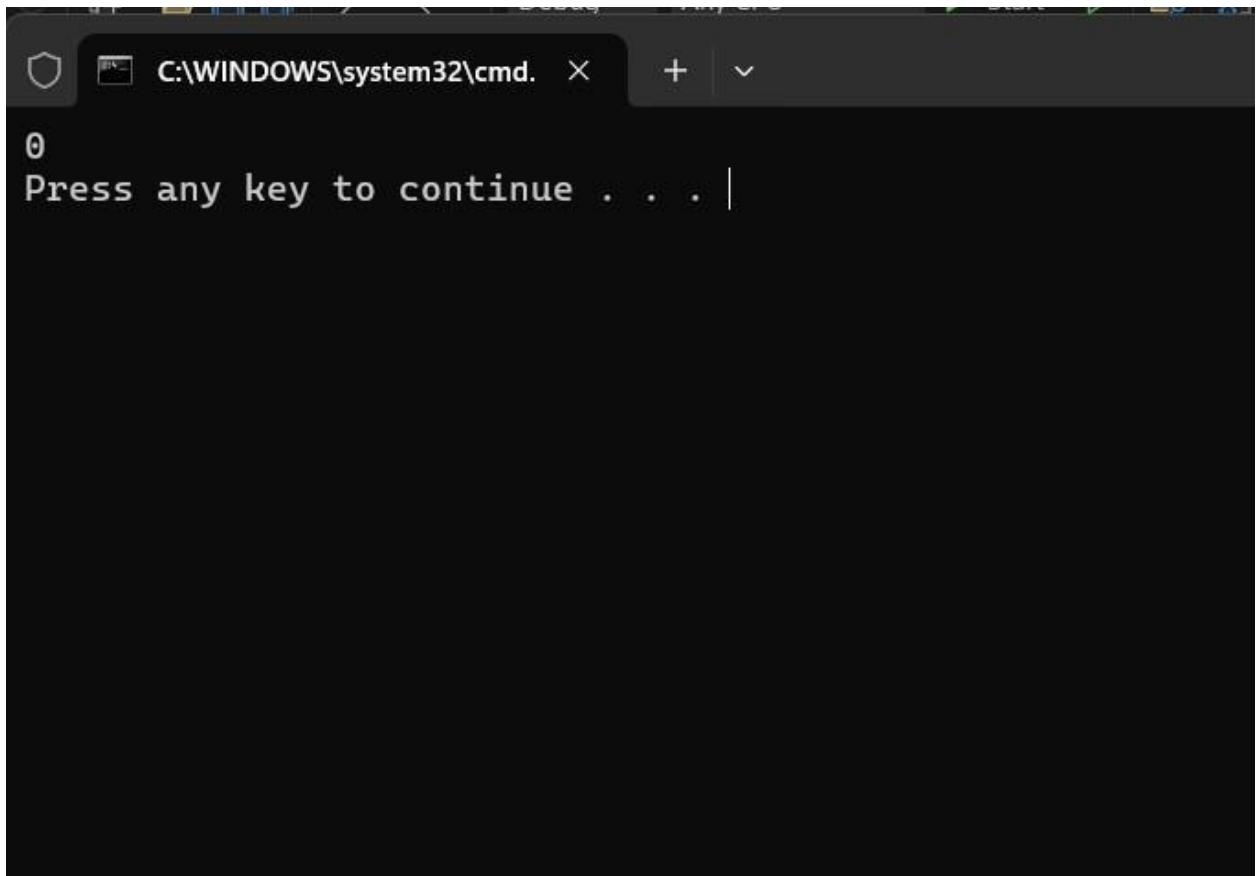
    min_penalty =
    float('inf')
    best_hour = 0
    for j in range(n + 1):
        penalty_open =
        prefix_N[j]
        penalty_closed = prefix_Y[n] - prefix_Y[j]

        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:



```
0
Press any key to continue . . . |
```

6.Count Palindromic Subsequences Given a string of digits s , return the number of palindromic subsequences of s having length 5. Since the answer may be very large, return it modulo $10^9 + 7$. 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 some or no characters without changing the order of the remaining characters.

Example 1: Input: $s = "103301"$ Output: 2

PROGRAM:

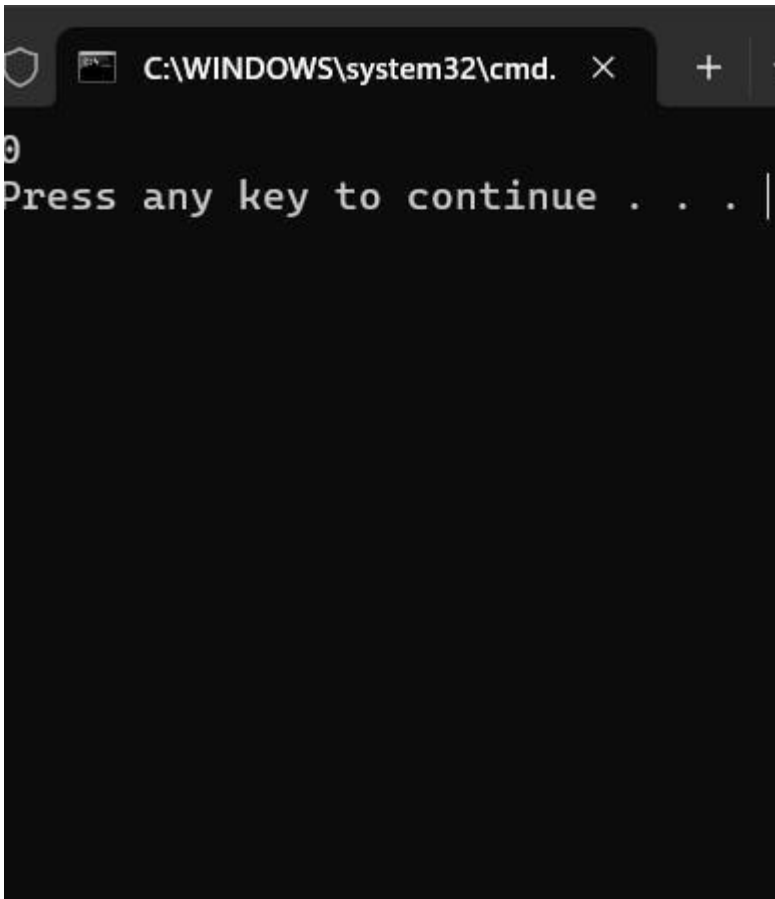
```
def countPalindromicSubsequences(s): MOD = 10**9 + 7
n = len(s)
dp = [[[0 for _ in range(4)] for _ in range(4)] for _ in range(n)]
for i in range(n):
```

```

        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 sum of all elements between x and n inclusively. Return the pivot integer x. If no such integer exists, return -1. It is guaranteed that there will be at most one pivot index for the given input. Example 1: Input:

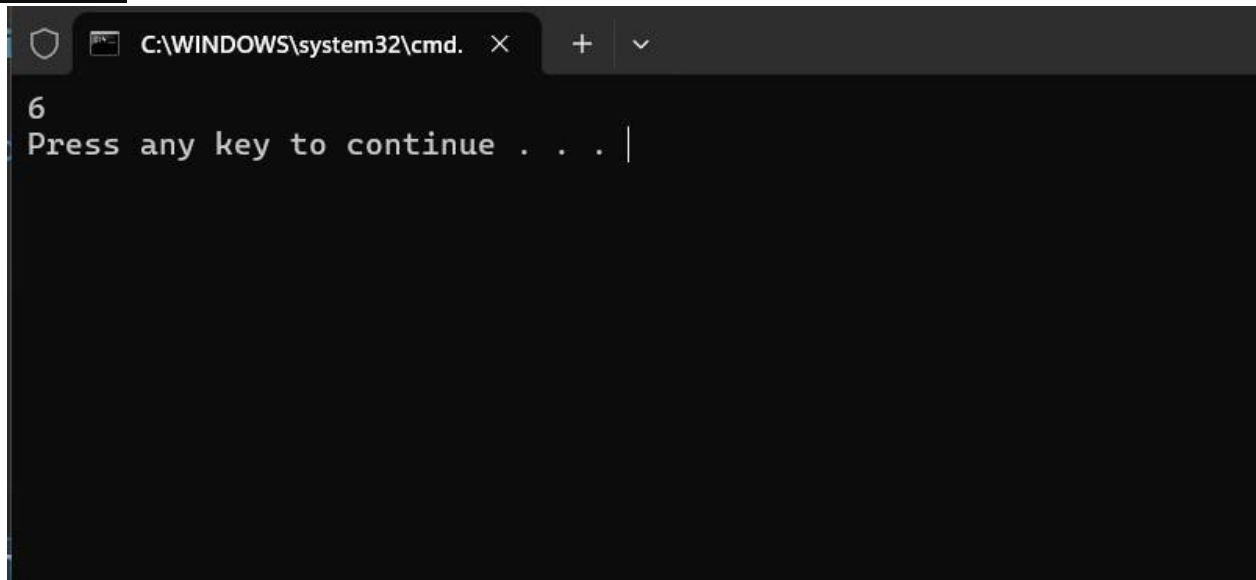
n = 8 Output: 6

Explanation: 6 is the pivot integer since: $1 + 2 + 3 + 4 + 5 + 6 = 6 + 7 + 8 = 21$. Example 2:

PROGRAM:

```
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_n - sum_x == x:
            return x
    return -1
n = 8
print(find_pivot_integer(n))
```

OUTPUT:

A screenshot of a Windows command prompt window. The title bar shows the path 'C:\WINDOWS\system32\cmd.' and standard window controls. The command prompt displays the number '6' on the first line, followed by the prompt 'Press any key to continue . . . |' on the second line, where the cursor is positioned.

8. Append Characters to String to Make Subsequence 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* so that *t* becomes a subsequence of *s*. A subsequence is a string that can be derived from another string by deleting some or no characters without changing the order of the remaining characters. Example 1: Input: *s* = "coaching", *t* = "coding" Output: 4

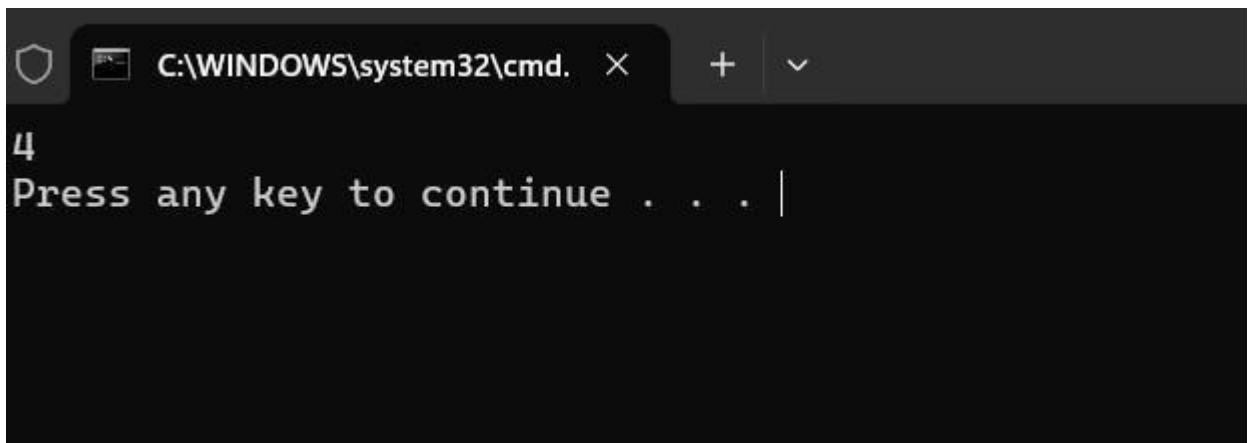
PROGRAM:

```
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:

A screenshot of a Windows command prompt window. The title bar shows the path 'C:\WINDOWS\system32\cmd.' and standard window controls. The command prompt displays the number '4' on the first line, followed by the prompt 'Press any key to continue . . . |' on the second line, where the cursor is positioned.

9. Remove Nodes From Linked List You are given the head of a linked list. Remove every node which has a node with a strictly greater value 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]

PROGRAM:

```
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)
```

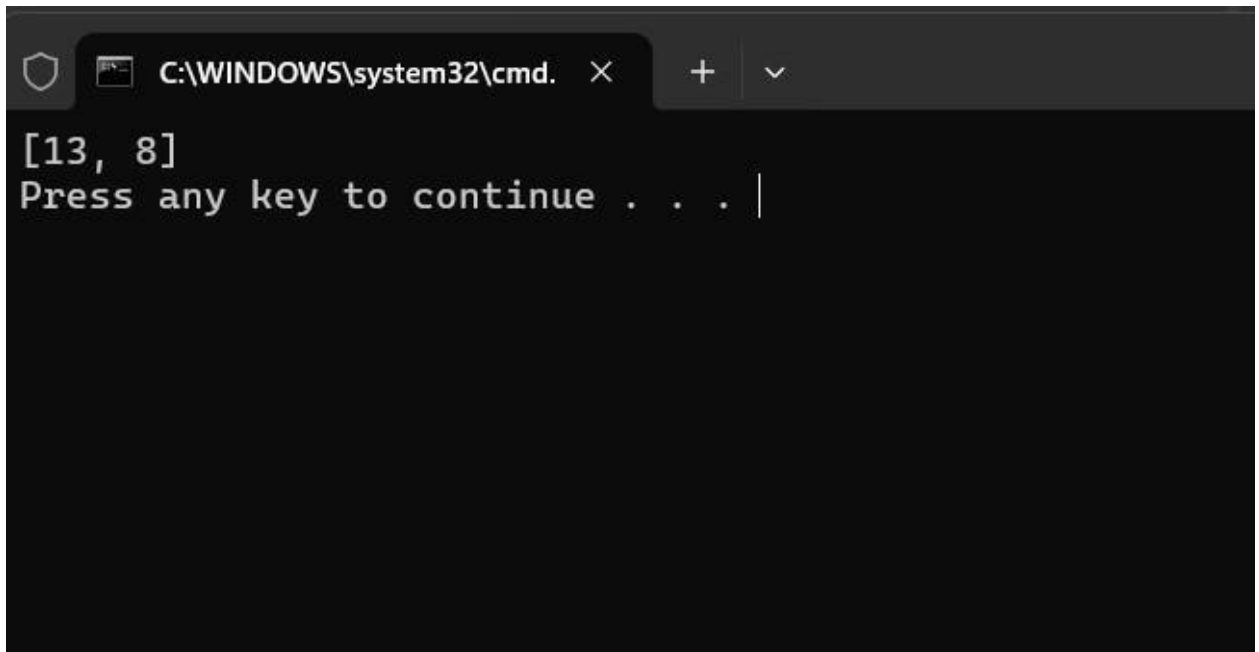
```

    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:



```

C:\WINDOWS\system32\cmd.
[13, 8]
Press any key to continue . . . |

```

10. Count Subarrays With Median K You are given an array `nums` of size `n` consisting of distinct integers from 1 to `n` and a positive integer `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 ascending order. If the 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

PROGRAM:

```
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:

