TEST QUESTIONS

1) minimum length

PROGRAM:

```
def min_length_after_operations(nums):
        return max(0, len(nums) - 2 * (len(nums) // 2))
# Example usage:
nums = [1,2,3,4]
print(min_length_after_operations(nums))
```

OUTPUT:

```
0
```

2) A substring is a contiguous sequence of characters within a string.

PROGRAM:

```
def find_substrings(words):
    substrings = []
    for word in words:
        for other_word in words:
            if word != other_word and word in other_word:
                substrings.append(word)
                break
    return substrings
words = ["mass","as","hero","superhero"]
print(find_substrings(words))
```

OUTPUT:

```
['as', 'hero']
```

3)BINARY MATRIX

PROGRAM:

```
from collections import deque

def update_matrix(mat):
    rows, cols = len(mat), len(mat[0])
    dist = [[float('inf')] * cols for _ in range(rows)]
    q = deque()
    for r in range(rows):
        for c in range(cols):
            if mat[r][c] == 0:
                dist[r][c] = 0
                q.append((r, c))
    directions = [(1, 0), (-1, 0), (0, 1), (0, -1)]
    while q:
        r, c = q.popleft()
        for dr, dc in directions:
            nr, nc = r + dr, c + dc
            if 0 <= nr < rows and 0 <= nc < cols and dist[nr][nc] > dist[r][c] + 1:
                dist[nr][nc] = dist[r][c] + 1
                q.append((nr, nc))

    return dist

mat1 = [[0,0,0],[0,1,0],[0,0,0]]
mat2 = [[0,0,0],[0,1,0],[1,1,1]]
print(update_matrix(mat1))
print(update_matrix(mat2))
```

OUTPUT:

```
[[0, 0, 0], [0, 1, 0], [0, 0, 0]]
[[0, 0, 0], [0, 1, 0], [1, 2, 1]]
```

4)MINIMUM NO.OF OPERATIONS:

PROGRAM:

```
def min_operations_to_increase(arr1, arr2):

    arr2.sort()

    def find_next(arr, target):
        left, right = 0, len(arr) - 1
        while left < right:
            mid = (left + right) // 2
            if arr[mid] <= target:
                left = mid + 1
            else:
                right = mid
        return left if arr[left] > target else -1
    count, prev = 0, float('-inf')

    for i in range(len(arr1)):
        if arr1[i] > prev:
            prev = arr1[i]
        else:
            index = find_next(arr2, prev)
            if index == -1: return -1
            arr1[i] = arr2[index]
            count += 1
            prev = arr2[index]

    return count

arr1 = [1,5,3,6,7]
arr2 = [1,3,2,4]
print(min_operations_to_increase(arr1, arr2))
```

OUTPUT:

```
==== RESTART: C:\Use
-1
```

5)STRING INTO SUBSTRING:

PROGRAM:

```
def repeated_string(a, b):
    if set(b) - set(a):
        return -1
    repeated_a = a
    count = 1
    while len(repeated_a) < len(b):
        repeated_a += a
        count += 1
    if b in repeated_a:
        return count
    if b in repeated_a + a:
        return count + 1

    return -1

a = "abcd"
b = "cdabcdab"
print(repeated_string(a, b))
```

OUTPUT:

```
3
```

6)MISSING NUMBER IN THE ARRAY:

PROGRAM:

```
def missing_number(nums):
    n = len(nums)
    expected_sum = n * (n + 1) // 2
    actual_sum = sum(nums)
    return expected_sum - actual_sum

nums = [3, 0, 1]
print(missing_number(nums))
```

OUTPUT:

```
2
```

7)INTEGER MATRIX

PROGRAM:

```
def largest_local(grid):
    n = len(grid)
    maxLocal = [[0] * (n - 2) for _ in range(n - 2)]

    for i in range(1, n - 1):
        for j in range(1, n - 1):
            max_value = 0
            for x in range(i - 1, i + 2):
                for y in range(j - 1, j + 2):
                    max_value = max(max_value, grid[x][y])
            maxLocal[i - 1][j - 1] = max_value

    return maxLocal
grid = [
    [9,9,8,1],
    [5,6,2,6],
    [8,2,6,4],
    [6,2,2,2]
]
print(largest_local(grid))
```

OUTPUT:

```
[[9, 9], [8, 6]]
```

8)PREIFX OF A STRING.

PROGRAM:

```
def count_prefixes(words, pref):
    return sum(word.startswith(pref) for word in words)

words = ["pay","attention","practice","attend"]
pref = "at"
print(count_prefixes(words, pref))
```

OUTPUT:

```
2
```

9)MXN INTEGER MATRIX

PROGRAM:

```
def set_zeroes(matrix):
    rows, cols = len(matrix), len(matrix[0])
    row_zero = False
    for i in range(rows):
        for j in range(cols):
            if matrix[i][j] == 0:
                matrix[i][0] = 0
                if j == 0:
                    row_zero = True
                else:
                    matrix[0][j] = 0
    for i in range(1, rows):
        for j in range(1, cols):
            if matrix[i][0] == 0 or matrix[0][j] == 0:
                matrix[i][j] = 0
    if matrix[0][0] == 0:
        for j in range(cols):
            matrix[0][j] = 0
    if row_zero:
        for i in range(rows):
            matrix[i][0] = 0

matrix = [[1,1,1],[1,0,1],[1,1,1]]
set_zeroes(matrix)
print(matrix)
```

OUTPUT:

```
---- RESTART: C:\Users\Sai17\AppData
[[1, 0, 1], [0, 0, 0], [1, 0, 1]]
```

10)TWO INTEGER ARRAY.

PROGRAM:

```
def intersection(nums1, nums2):
    return list(set(nums1) & set(nums2))

nums1 = [1,2,2,1]
nums2 = [2,2]
print(intersection(nums1, nums2))
```

OUTPUT:

```
[2]
```