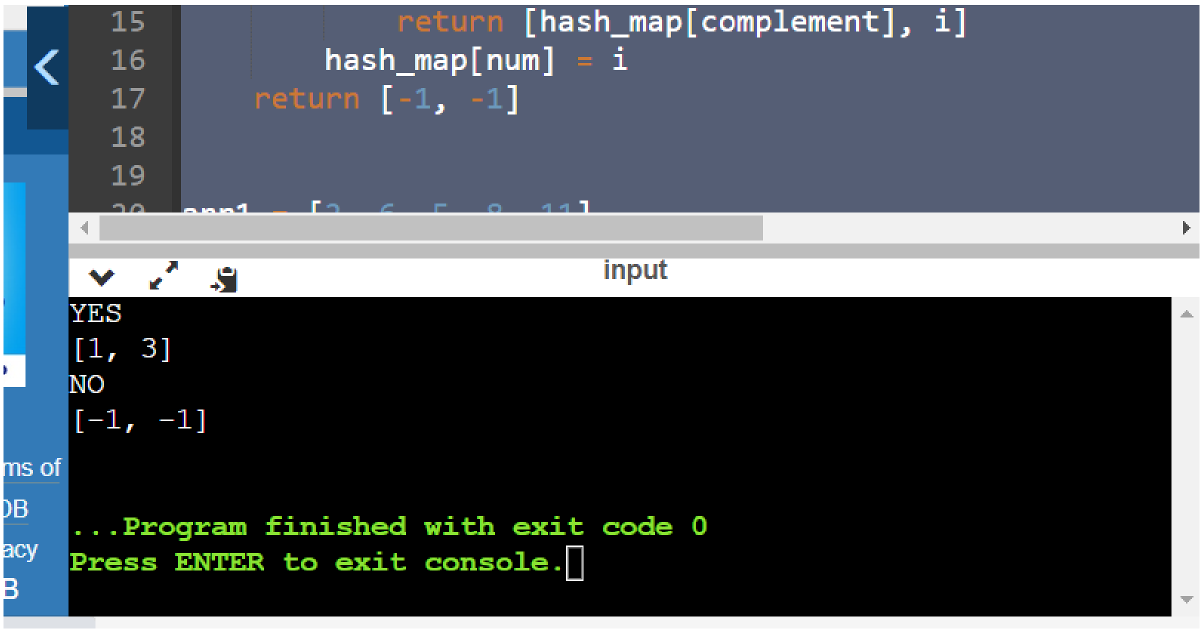
Tanish Mittal
IT-C
2000290130174

Day – 4 : Arrays- IV

Problem 1: **Two Sum : Check if a pair with given sum exists in Array**

```
def find_two_numbers_with_sum(arr, target):
    hash_set = set()
    for num in arr:
        complement = target - num
        if complement in hash_set:
            return "YES"
        hash_set.add(num)
    return "NO"


def find_indices_of_two_numbers_with_sum(arr, target):
    hash_map = {}
    for i, num in enumerate(arr):
        complement = target - num
        if complement in hash_map:
            return [hash_map[complement], i]
        hash_map[num] = i
    return [-1, -1]



arr1 = [2, 6, 5, 8, 11]
target1 = 14
print(find_two_numbers_with_sum(arr1, target1))
print(find_indices_of_two_numbers_with_sum(arr1, target1))


arr2 = [2, 6, 5, 8, 11]
target2 = 15
print(find_two_numbers_with_sum(arr2, target2))
print(find_indices_of_two_numbers_with_sum(arr2, target2))
```



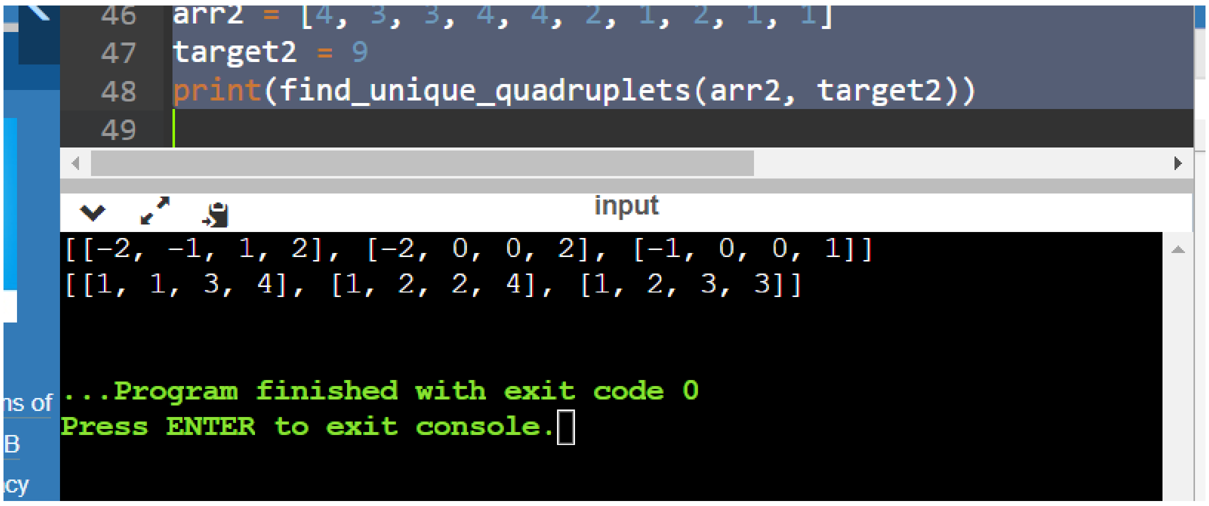
```
            return [hash_map[complement], i]
        hash_map[num] = i
    return [-1, -1]
```

```
YES
[1, 3]
NO
[-1, -1]


...Program finished with exit code 0
Press ENTER to exit console.
```

Problem -2: Given an array of N integers, your task is to find unique quads that add up to give a target value. In short, you need to return an array of all the unique quadruplets [arr[a], arr[b], arr[c], arr[d]] such that their sum is equal to a given target

```
def find_unique_quadruplets(arr, target):
    n = len(arr)
    arr.sort()
    result = []

    for a in range(n - 3):
        # Skip duplicate elements for a
        if a > 0 and arr[a] == arr[a - 1]:
            continue

        for b in range(a + 1, n - 2):
            # Skip duplicate elements for b
            if b > a + 1 and arr[b] == arr[b - 1]:
                continue
            left = b + 1
            right = n - 1

            while left < right:
                quad_sum = arr[a] + arr[b] + arr[left] + arr[right]

                if quad_sum == target:
                    result.append([arr[a], arr[b], arr[left], arr[right]])

                    # Skip duplicate elements for left and right
                    while left < right and arr[left] == arr[left + 1]:
                        left += 1
                    while left < right and arr[right] == arr[right - 1]:
                        right -= 1

                    left += 1
                    right -= 1
                elif quad_sum < target:
                    left += 1
                else:
                    right -= 1

    return result



arr1 = [1, 0, -1, 0, -2, 2]
target1 = 0
print(find_unique_quadruplets(arr1, target1))
```

```
arr2 = [4, 3, 3, 4, 4, 2, 1, 2, 1, 1]

target2 = 9

print(find_unique_quadruplets(arr2, target2))
```



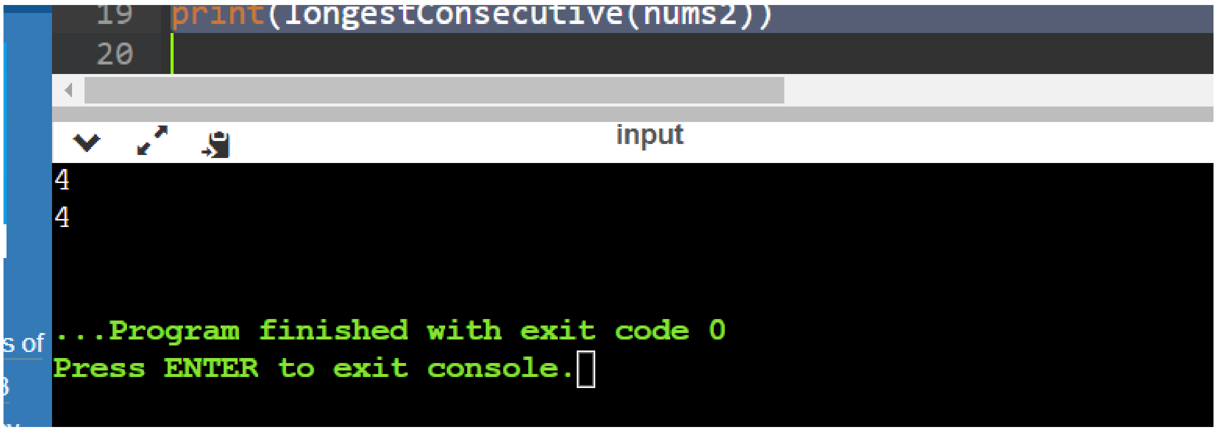
```
arr2 = [4, 3, 3, 4, 4, 2, 1, 2, 1, 1]
target2 = 9
print(find_unique_quadruplets(arr2, target2))
```

```
[[-2, -1, 1, 2], [-2, 0, 0, 2], [-1, 0, 0, 1]]
[[1, 1, 3, 4], [1, 2, 2, 4], [1, 2, 3, 3]]


...Program finished with exit code 0
Press ENTER to exit console.
```

Problem 3: you are given an array of 'N' integers. You need to find the length of the longest sequence which contains the consecutive elements.

```
def longestConsecutive(nums):

    numSet = set(nums)

    maxLen = 0


    for num in nums:

        if num - 1 not in numSet:

            currLen = 1

            while num + 1 in numSet:

                num += 1

                currLen += 1

            maxLen = max(maxLen, currLen)


    return maxLen


nums1 = [100, 200, 1, 3, 2, 4]
```

```
print(longestConsecutive(nums1))


nums2 = [3, 8, 5, 7, 6]

print(longestConsecutive(nums2))
```

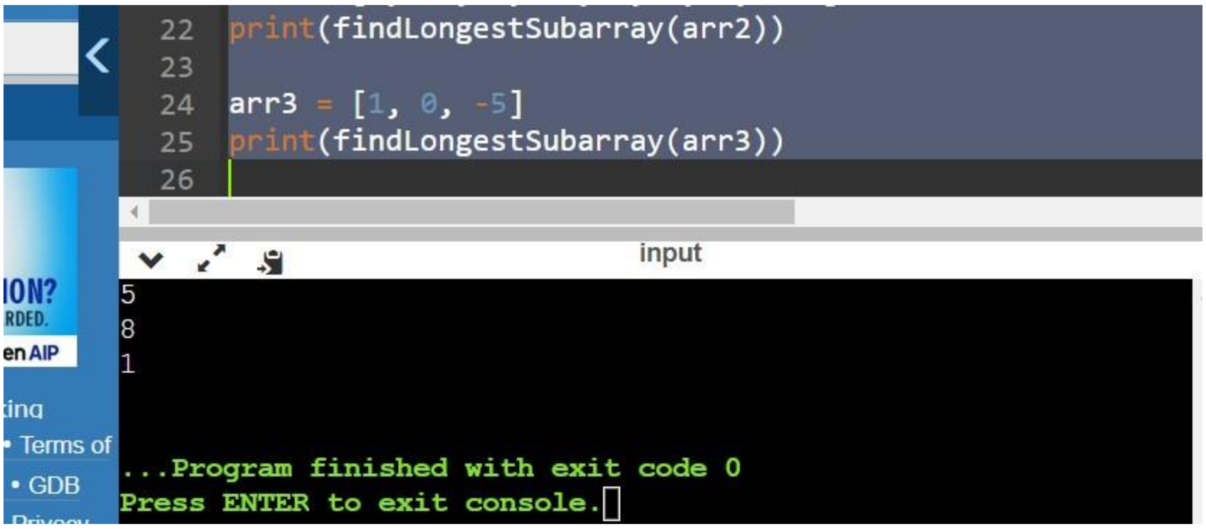


```
print(longestConsecutive(nums2))
```

```
4
4


...Program finished with exit code 0
Press ENTER to exit console.
```

Problem 4: Given an array containing both positive and negative integers, we have to find the length of the longest subarray with the sum of all elements equal to zero.

```
def findLongestSubarray(arr):

maxLen = 0

curSum = 0

sumDict = {}


for i in range(len(arr)):

    curSum += arr[i]


    if curSum == 0:

        maxLen = i + 1
        if curSum in sumDict:
            maxLen = max(maxLen, i - sumDict[curSum])
        else:
            sumDict[curSum] = i

    return maxLen
arr1 = [9, -3, 3, -1, 6, -5]
print(findLongestSubarray(arr1))

arr2 = [6, -2, 2, -8, 1, 7, 4, -10]
print(findLongestSubarray(arr2))

arr3 = [1, 0, -5]
print(findLongestSubarray(arr3))
```

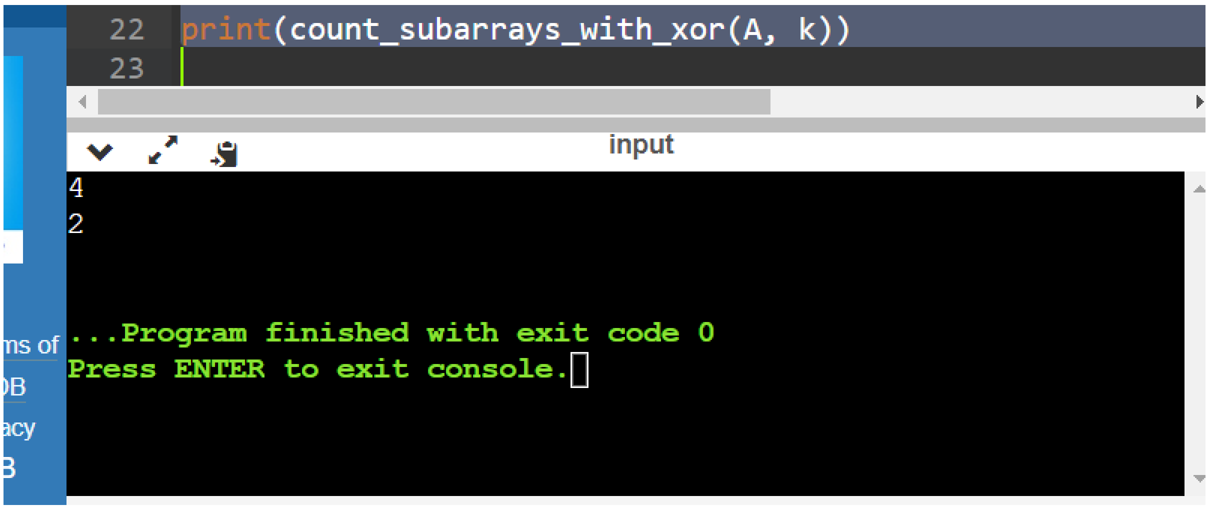


```
print(findLongestSubarray(arr2))

arr3 = [1, 0, -5]
print(findLongestSubarray(arr3))
```

```
5
8
1


...Program finished with exit code 0
Press ENTER to exit console.
```

Problem 5: Given an array of integers A and an integer B. Find the total number of subarrays having bitwise XOR of all elements equal to k.

```
def count_subarrays_with_xor(A, k):
    count = 0
    prefix_xor_count = {0: 1}
    prefix_xor = 0

    for num in A:
        prefix_xor ^= num
        desired_xor = prefix_xor ^ k

        if desired_xor in prefix_xor_count:
            count += prefix_xor_count[desired_xor]

        prefix_xor_count[prefix_xor] = prefix_xor_count.get(prefix_xor, 0) + 1

    return count
A = [4, 2, 2, 6, 4]
k = 6
print(count_subarrays_with_xor(A, k))

A = [5, 6, 7, 8, 9]
k = 5
print(count_subarrays_with_xor(A, k))
```



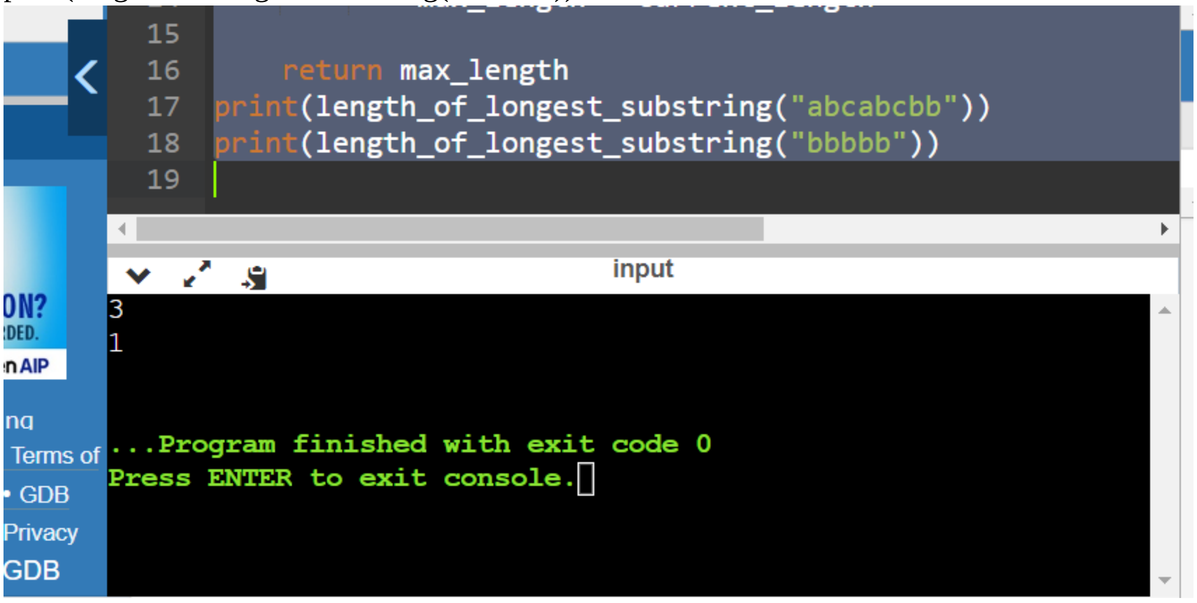
```
print(count_subarrays_with_xor(A, k))
```

```
4
2


...Program finished with exit code 0
Press ENTER to exit console.
```

Problem 6: Given a String, find the length of longest substring without any repeating character.

```
def length_of_longest_substring(s):

    max_length = 0

    char_map = {}

    start = 0


    for end in range(len(s)):

        if s[end] in char_map and char_map[s[end]] >= start:

            start = char_map[s[end]] + 1


        char_map[s[end]] = end

        current_length = end - start + 1


        if current_length > max_length:

            max_length = current_length


    return max_length

print(length_of_longest_substring("abcabcbb"))
```

print(length_of_longest_substring("bbbbb"))



```
    return max_length
print(length_of_longest_substring("abcabcbb"))
print(length_of_longest_substring("bbbbb"))
```

```
3
1


...Program finished with exit code 0
Press ENTER to exit console.
```