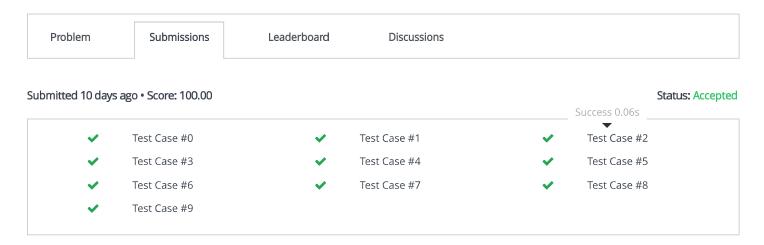
Mencari Teman Dekat



Submitted Code

```
Language: Python 3
                                                                                             P Open in editor
1 # Input: n = length of array
2 n = int(input())
4 # Input: arr = sorted array
5 arr = list(map(int,input().split()))
7 # Input: k = jumlah elemen terdekat, target = target value
8 k,target = map(int,input().split())
10 # Fungsi untuk menemukan titik crossover menggunakan pencarian biner
11 def find_crossover(arr,target):
       # Initialize low and high indices
12
      low = 0
13
      high = len(arr) - 1
14
15
       # Loop sampai low <= high
16
17
       while low <= high:
           # cari mid index
18
19
          mid = (low + high) // 2
20
           # If target is found, return mid index
21
           if arr[mid] == target:
22
23
               return mid
24
25
           # If target is smaller than mid element, move high to left
           elif arr[mid] > target:
26
27
               high = mid - 1
28
29
           # If target is larger than mid element, move low to right
           else:
30
               low = mid + 1
31
32
```

```
# Return low index as crossover point
33
34
       return low
35
36 # Function to find k closest elements using two pointers
37 def find_k_closest(arr,k,target):
38
       # Initialize result list
39
       result = []
40
       # Find crossover point index
41
42
       cross_index = find_crossover(arr,target)
43
44
       # Initialize left and right pointers around crossover point
45
       left = cross_index - 1
       right = cross_index
46
47
       # Loop until result list has k elements or pointers are out of bounds
48
      while len(result) < k and (left >= 0 or right < len(arr)):</pre>
49
50
           # If left pointer is valid and right pointer is invalid or left element is closer than right
51
  element
52
           if left >= 0 and (right >= len(arr) or abs(arr[left] - target) <= abs(arr[right] - target)):
53
54
               # Add left element to result list at front
55
               result.insert(0,arr[left])
56
57
               # Move left pointer to left
               left -= 1
58
59
           # Else if right pointer is valid
60
61
           elif right < len(arr):</pre>
62
63
               # Add right element to result list at back
64
               result.append(arr[right])
65
66
               # Move right pointer to right
67
               right += 1
68
69
       # Return result list
       return result
70
71
72 # Print the output by calling the function
73 print(*find_k_closest(arr,k,target))
```