**1.You are given an array of integers and a target integer k. Write a function to find the element in the array that is closest to k. If there are multiple elements equidistant from k, return the smallest of them**.

**Input Format**

6

4 18 32 21 9 17

30

**Constraints**

The array may contain both positive and negative integers. The array is not sorted. 1 ≤ array length ≤ 10,000 -10,000 ≤ array elements, x ≤ 10,000

**Output Format**

32

**Sample Input**

6

4 18 32 21 9 17

30

**Sample Output**

32

**Explanation**

32 is nearest to 30

**Solution:**

def find\_closest\_element(arr, k):

closest\_element = arr[0]

smallest\_diff = abs(arr[0] - k)

for num in arr[1:]:

diff = abs(num - k)

if diff < smallest\_diff:

closest\_element = num

smallest\_diff = diff

elif diff == smallest\_diff and num < closest\_element:

closest\_element = num

return closest\_element

n = int(input())

arr = list(map(int, input().split()))

k = int(input())

print(find\_closest\_element(arr, k))

**2.Given an array of integers, write a function that sorts the array using the Bubble Sort algorithm. While sorting, count the total number of adjacent swaps made and return this count.**

**Input Format**

3

3 2 1

**Constraints**

The array may contain duplicate values. 1 ≤ array length ≤ 1,000 -1,000 ≤ array elements ≤ 1,000

**Output Format**

1 2 3

3

**Sample Input 0**

5

5 1 4 2 8

**Sample Output 0**

1 2 4 5 8 4

**Solution :**

def bubble\_sort\_and\_count\_swaps(arr):

n = len(arr)

swap\_count = 0

for i in range(n):

for j in range(0, n-i-1):

if arr[j] > arr[j+1]:

arr[j], arr[j+1] = arr[j+1], arr[j]

swap\_count += 1

return arr, swap\_count

n = int(input())

arr = list(map(int, input().split())

sorted\_arr, total\_swaps = bubble\_sort\_and\_count\_swaps(arr)

print(" ".join(map(str, sorted\_arr)))

print(total\_swaps)

**3.Given an array of integers and an integer K, write a function that partially sorts the array such that only the top K smallest elements are sorted in ascending order at the beginning of the array. The rest of the array can remain unsorted.**

**Input Format**

5

7 4 1 8 5

3

**Constraints**

1 ≤ K ≤ array length ≤ 1,000 -1,000 ≤ array elements ≤ 1,000

**Output Format**

1 4 5 8 7

**Sample Input 0**

6

6 3 2 10 8 1

4

**Sample Output 0**

1 2 3 6 8 10

**Solution :**

def partially\_sort\_array(arr, K):

sorted\_arr = sorted(arr)

top\_k\_sorted = sorted\_arr[:K]

result = top\_k\_sorted + sorted\_arr[K:]

return result

# Input reading

if \_\_name\_\_ == "\_\_main\_\_":

n = int(input())

arr = list(map(int, input().split()))

K = int(input())

# Ensure K is valid

if K < 1 or K > n:

print("Error: K must be between 1 and the length of the array.")

else:

result = partially\_sort\_array(arr, K)

print(" ".join(map(str, result)))

**4.Given an array of integers, write a function to sort the array in ascending order using the Insertion Sort algorithm. However, during the sorting process, ignore any element that is greater than a specified threshold T.**

**Input Format**

5

9 5 2 11 6

**Constraints**

1 ≤ array length ≤ 1,000 -1,000 ≤ array elements, T ≤ 1,000

**Output Format**

2 5 6 11 9

**Sample Input 0**

5

3 1 4 10 2

3

**Sample Output 0**

1 2 3 4 10

**Solution :**

def insertion\_sort\_with\_threshold(arr, T):

n = len(arr)

for i in range(1, n):

key = arr[i]

j = i - 1

if key <= T:

while j >= 0 and arr[j] > key:

arr[j + 1] = arr[j]

j -= 1

arr[j + 1] = key

return arr

if \_\_name\_\_ == "\_\_main\_\_":

n = int(input())

arr = list(map(int, input().split()))

T = int(input())

sorted\_arr = insertion\_sort\_with\_threshold(arr, T)

print(" ".join(map(str, sorted\_arr)))