Week-10

1. Write a Python program to sort a list of elements using the merge sort algorithm.

**For example:**

| **Input** | **Result** |
| --- | --- |
| 5  6 5 4 3 8 | 3 4 5 6 8 |

Program:

def mergeSort(arr):

if len(arr) > 1:

mid = len(arr) // 2

left\_half = arr[:mid]

right\_half = arr[mid:]

mergeSort(left\_half)

mergeSort(right\_half)

i = j = k = 0

while i < len(left\_half) and j < len(right\_half):

if left\_half[i] < right\_half[j]:

arr[k] = left\_half[i]

i += 1

else:

arr[k] = right\_half[j]

j += 1

k += 1

while i < len(left\_half):

arr[k] = left\_half[i]

i += 1

k += 1

while j < len(right\_half):

arr[k] = right\_half[j]

j += 1

k += 1

def printList(arr):

for i in range(len(arr)):

print(arr[i], end=" ")

print()

# Input

n = int(input())

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

# Sorting

mergeSort(arr)

# Output

printList(arr)

output:

|  | **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- | --- |
|  | 5  6 5 4 3 8 | 3 4 5 6 8 | 3 4 5 6 8 |  |
|  | 9  14 46 43 27 57 41 45 21 70 | 14 21 27 41 43 45 46 57 70 | 14 21 27 41 43 45 46 57 70 |  |
|  | 4  86 43 23 49 | 23 43 49 86 | 23 43 49 86 |  |

Passed all tests!

**Correct**

2. An list contains N numbers and you want to determine whether two of the numbers sum to a given number K. For example, if the input is 8, 4, 1, 6 and K is 10, the answer is yes (4 and 6). A number may be used twice.

**Input Format**

The first line contains a single integer n , the length of list

The second line contains n space-separated integers, list[i].

The third line contains integer k.

**Output Format**

Print Yes or No.

**Sample Input**

7

0 1 2 4 6 5 3

1

**Sample Output**

Yes

**For example:**

| **Input** | **Result** |
| --- | --- |
| 5  8 9 12 15 3  11 | Yes |
| 6  2 9 21 32 43 43 1  4 | No |

Program:

def hasSumPair(arr, k):

# Create an empty set to store visited elements

visited = set()

# Iterate through the array

for num in arr:

# Calculate the difference required to reach the target sum

diff = k - num

# If the difference is already in the visited set, a pair with the sum exists

if diff in visited:

return "Yes"

# Add the current number to the visited set

visited.add(num)

# If no such pair exists

return "No"

# Input

n = int(input())

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

k = int(input())

# Check if a pair with sum k exists in the list

print(hasSumPair(arr, k))

output:

|  | **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- | --- |
|  | 5  8 9 12 15 3  11 | Yes | Yes |  |
|  | 6  2 9 21 32 43 43 1  4 | No | No |  |
|  | 6  13 42 31 4 8 9  17 | Yes | Yes |  |

Passed all tests!

**Correct**

3. Bubble Sort is the simplest sorting algorithm that works by repeatedly swapping the adjacent elements if they are in wrong order. You read an list of numbers. You need to arrange the elements in ascending order and print the result. The sorting should be done using bubble sort.

**Input Format:**The first line reads the number of elements in the array. The second line reads the array elements one by one.

**Output Format:** The output should be a sorted list.

**For example:**

| **Input** | **Result** |
| --- | --- |
| 6  3 4 8 7 1 2 | 1 2 3 4 7 8 |
| 5  4 5 2 3 1 | 1 2 3 4 5 |

Program:

def bubbleSort(arr):

n = len(arr)

# Traverse through all array elements

for i in range(n):

# Last i elements are already in place, so we can skip them

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

# Traverse the array from 0 to n-i-1

# Swap if the element found is greater than the next element

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

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

# Input

n = int(input())

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

# Sorting using bubble sort

bubbleSort(arr)

# Output

for num in arr:

print(num, end=" ")

output:

|  | **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- | --- |
|  | 6  3 4 8 7 1 2 | 1 2 3 4 7 8 | 1 2 3 4 7 8 |  |
|  | 6  9 18 1 3 4 6 | 1 3 4 6 9 18 | 1 3 4 6 9 18 |  |
|  | 5  4 5 2 3 1 | 1 2 3 4 5 | 1 2 3 4 5 |  |

Passed all tests!

**Correct**

4. Write a Python program for binary search.

**For example:**

| **Input** | **Result** |
| --- | --- |
| 1,2,3,5,8  6 | False |
| 3,5,9,45,42  42 | True |

Program:

def binary\_search(arr, target):

left, right = 0, len(arr) - 1

while left <= right:

mid = (left + right) // 2

if arr[mid] == target:

return True

elif arr[mid] < target:

left = mid + 1

else:

right = mid - 1

return False

arr\_input = input()

target\_input = input()

arr = list(map(int, arr\_input.split(',')))

target = int(target\_input)

arr.sort()

result = binary\_search(arr, target)

print(result)

output:

|  | **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- | --- |
|  | 1,2,3,5,8  6 | False | False |  |
|  | 3,5,9,45,42  42 | True | True |  |
|  | 52,45,89,43,11  11 | True | True |  |

Passed all tests!

**Correct**

5. To find the frequency of numbers in a list and display in sorted order.

**Constraints:**

1<=n, arr[i]<=100

**Input:**

1 68 79 4 90 68 1 4 5

**output:**

 1 2

 4 2

 5 1

 68 2

 79 1

90 1

**For example:**

| **Input** | **Result** |
| --- | --- |
| 4 3 5 3 4 5 | 3 2  4 2  5 2 |

Program:

def frequencySort(arr):

# Create an empty dictionary to store the frequency of each number

frequency = {}

# Calculate the frequency of each number

for num in arr:

frequency[num] = frequency.get(num, 0) + 1

# Sort the dictionary based on keys (numbers)

sorted\_frequency = sorted(frequency.items())

# Print the sorted frequency

for num, freq in sorted\_frequency:

print(num, freq)

# Input

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

# Display frequency of numbers in sorted order

frequencySort(arr)

output;

|  | **Input** | **Expected** | **Got** |  |
| --- | --- | --- | --- | --- |
|  | 4 3 5 3 4 5 | 3 2  4 2  5 2 | 3 2  4 2  5 2 |  |
|  | 12 4 4 4 2 3 5 | 2 1  3 1  4 3  5 1  12 1 | 2 1  3 1  4 3  5 1  12 1 |  |
|  | 5 4 5 4 6 5 7 3 | 3 1  4 2  5 3  6 1  7 1 | 3 1  4 2  5 3  6 1  7 1 |  |

Passed all tests!

**Correct**