**Task - 5. Implement various Searching and Sorting Operations in python programming**

**5.a. A company stores employee records in a list of dictionaries, where each dictionary contains id, name, and department. Write a function find\_employee\_by\_id that takes this list and a target employee ID as arguments and returns the dictionary of the employee with the matching ID, or None if no such employee is found.**

**Sample Input and Output:**

**Input:**

**employees = [**

**{"id": 1, "name": "Riya", "department": "Marketing"},**

**{"id": 2, "name": "Arjun", "department": "Sales"},**

**{"id": 3, "name": "Meera", "department": "Engineering"},**

**{"id": 4, "name": "Kiran", "department": "Design"}**

**]**

**Output:**

**{'id': 203, 'name': 'Meera', 'department': 'Engineering'}**

**Program:**

def find\_employee\_by\_id(employees, target\_id):

for employee in employees:

if employee['id'] == target\_id:

return employee

return None

# Test the function

employees = [

{'id': 1, 'name': 'Alice', 'department': 'HR'},

{'id': 2, 'name': 'Bob', 'department': 'Engineering'},

{'id': 3, 'name': 'Charlie', 'department': 'Sales'},

]

print(find\_employee\_by\_id(employees, 3))

**5.b. You are developing a grade management system for a school. The system maintains a list of student records, where each record is represented as a dictionary containing a student's name and score. The school needs to generate a report that displays students scores in ascending order. Your task is to implement a feature that sorts the student records by their scores using the Bubble Sort algorithm.**

**Sample Input and Output:**

**Before sorting:**

**{'name': 'Alice', 'score': 88}**

**{'name': 'Bob', 'score': 95}**

**{'name': 'Charlie', 'score': 75}**

**{'name': 'Diana', 'score': 85}**

**After sorting:**

**{'name': 'Charlie', 'score': 75}**

**{'name': 'Diana', 'score': 85}**

**{'name': 'Alice', 'score': 88}**

**{'name': 'Bob', 'score': 95}**

**Program:**

def bubble\_sort\_scores(students):

n = len(students)

for i in range(n):

# Track if any swap is made in this pass

swapped = False

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

if students[j]['score'] > students[j+1]['score']:

# Swap if the score of the current student is greater than the next

students[j], students[j+1] = students[j+1], students[j]

swapped = True

# If no two elements were swapped, the list is already sorted

if not swapped:

break

# Example usage

students = [

{'name': 'Alice', 'score': 88},

{'name': 'Bob', 'score': 95},

{'name': 'Charlie', 'score': 75},

{'name': 'Diana', 'score': 85}

]

print("Before sorting:")

for student in students:

print(student)

bubble\_sort\_scores(students)

print("\nAfter sorting:")

for student in students:

print(student)

**5.c. Riya has two friends who each gave her a list of exam scores. Both lists are already sorted in non-increasing order (from highest to lowest). Riya wants to combine these two lists into a single list that is also sorted in non-increasing order. Since the lists are big and she doesn’t want to do it manually, she asks for your help to merge them into one sorted list.**

**Sample Input**

1

4 5

9 7 5 3

8 6 4 2 0

**Sample Output**

9 8 7 6 5 4 3 2 0

**Program:**

def merge\_descending(list1, list2):

i, j = 0, 0

merged = []

# Merge both lists

while i < len(list1) and j < len(list2):

if list1[i] >= list2[j]:

merged.append(list1[i])

i += 1

else:

merged.append(list2[j])

j += 1

# Add remaining elements

while i < len(list1):

merged.append(list1[i])

i += 1

while j < len(list2):

merged.append(list2[j])

j += 1

return merged

# ---- Driver Code ----

t = int(input()) # number of test cases

for \_ in range(t):

n, m = map(int, input().split())

list1 = list(map(int, input().split()))

list2 = list(map(int, input().split()))

result = merge\_descending(list1, list2)

print(\*result)