Q1: The Student Merit System

Description:

Define a structure Student with fields: ID(int), Name(string), and Marks(float array of 3 subjects). Write a program to:

1. Read details for N students.
2. Calculate the **Average** for each student.
3. Print a "Merit List" of students who scored **above 75%** average.
4. Identify and print the details of the **Topper** (Highest Average).

Input Format:

- N (Number of students).
- For each student: ID, Name, Mark1, Mark2, Mark3.

Sample Test Case 1:

Input:
2
101 Alice 80 90 85
102 Bob 60 50 55

Output:
Merit List:
101 Alice (Avg: 85.00)
Topper: Alice

Sample Test Case 2:

Input:
1
500 John 40 40 40

Output:
Merit List:
(No Output)
Topper: John

Q2: Matrix Multiplication

Description:

Write a program to multiply two matrices.

1. Read two matrices A and B
2. Check if multiplication is possible (Condition: C1 == R2). If not, print "Not Possible".
3. Implement a function multiplyMatrices() that computes the product.
4. Print the resulting matrix.

Input Format:

- Dimensions of A (R1, C1), followed by elements of A.
- Dimensions of B (R2, C2), followed by elements of B.

Sample Test Case 1:

Input:
2 2
1 2
3 4
2 2
1 0
0 1

Output:
1 2
3 4

Sample Test Case 2:

Input:
2 2
1 2
3 4
3 2
1 1
1 1
1 1

Output:
Not Possible

Q3: The Text Analyzer

Description:

Write a program that accepts a sentence string and performs three distinct analyzes using helper functions:

1. **Word Count:** Count total words (based on whitespace).
2. **Vowel Count:** Count total vowels (a, e, i, o, u).
3. **Reverse:** Print the reversed string.

Input Format:

- A single string (may contain spaces).

Sample Test Case 1:

Input:
Hello World

Output:
Words: 2
Vowels: 3
Reversed: dlroW olleH

Sample Test Case 2:

Input:
C Programming

Output:
Words: 2
Vowels: 3
Reversed: gnimmargorP C

Q4: Set Operations

Description:

Given two arrays A and B of distinct integers, implement two functions:

1. findUnion(): Print elements present in **either** A or B (no duplicates).
2. findIntersection(): Print elements present in both A and B.
   Note: The output order does not matter.

Input Format:

- Size of A (N1), elements of A.
- Size of B (N2), elements of B.

Sample Test Case 1:

Input:
3
1 2 3
3
2 3 4

Output:
Union: 1 2 3 4
Intersection: 2 3

Sample Test Case 2:

Input:
2
10 20
2
30 40

Output:
Union: 10 20 30 40
Intersection: (None)

Q5: Merge Sorted Arrays

Description:

Given two sorted arrays, merge them into a single sorted array.

Constraint: You must use the efficient "Two Pointer" method, not just concatenating and running Bubble Sort.

1. Compare the current elements of both arrays.
2. Pick the smaller one and add it to the result.
3. Repeat until one array is empty, then copy the rest.

Input Format:

- Size N1, sorted elements of Array 1.
- Size N2, sorted elements of Array 2.

Sample Test Case 1:

Input:
3
1 3 5
3
2 4 6

Output:
1 2 3 4 5 6

Sample Test Case 2:

Input:
2
10 50
3
20 30 40

Output:
10 20 30 40 50

Q6: Frequency Counter

Description:

Given an array of N integers, write a program to count the frequency of each distinct element.

- You must handle negative numbers.
- The output should display the number and how many times it appears.
- *Hint: You can sort the array first to make counting easier, or use a visited array.*

Input Format:

- N(Size), followed by N integers.

Sample Test Case 1:

Input:
8
10 20 20 10 10 20 5 20

Output:
10 -> 3
20 -> 4
5 -> 1

Sample Test Case 2:

Input:
3
1 1 1

Output:
1 -> 3

Q7: String Compression

Description:

Write a program to compress a string by replacing consecutive identical characters with the character followed by its count.

- Example: aaabbcbecomes a3b2c1.
- If the compressed string is not smaller than the original (eg, abc-> a1b1c1), print the original string.

Input Format:

- A single string (no spaces).

Sample Test Case 1:

Input:
aaabbc

Output:
a3b2c1

Sample Test Case 2:

Input:
aabb

Output:
a2b2

Q8: The Array Rotator

Description:

Implement a menu-driven program (or two separate functions) that can rotate an array to the Left or to the Right by K positions.

1. rotateLeft(arr, k): Shift elements left. First element moves to end.
2. rotateRight(arr, k): Shift elements right. Last element moves to front.
   Note: K can be larger than N.

Input Format:

- N(Size)
- N integers
- D(Direction: 0 for Left, 1 for Right).
- K(Number of shifts).

Sample Test Case 1:

Input:
5
1 2 3 4 5
0
1 (Left Rotate by 1)

Output:
2 3 4 5 1

Sample Test Case 2:

Input:
3
10 20 30
1
2 (Right Rotate by 2)

Output:
20 30 10