Problem: Merging 2 arrays

Merging two 1D arrays involves combining the elements of both arrays into a single array while maintaining their original order.

Here's a step-by-step process for merging two 1D arrays:

- Create a New Array: Create a new array that is large enough to hold the combined elements of both arrays. The size of the new array should be the sum of the sizes of the two original arrays.
- Copy Elements: Iterate through the elements of the first array and copy them to the new array.

  Then, iterate through the elements of the second array and copy them to the new array after the elements of the first array.
- Result: The new array now contains all the elements from both original arrays, merged in the desired order.

Task 1:

You are given a template code in the IDE.

Update the code to merge the 2 arrays based on the process defined above.

```
#include <bits/stdc++.h>
using namespace std;

int main(){
    int arr1[100] = {2, 4, 6}; // First
    array int size1 = 3;

    int arr2[100] = {8, 10, 12, 14}; // Second array
    int size2 = 4;

    // Update the code below to solve the problem
```

```
    // Print the merged array
    for(int i=0 ; i<mergedSize ; i++){
        cout<<merged[i]<<" ";
    }

    return 0;
}
```

Problem: Sum of Array elements

Given an array A, Output the sum of all elements in A.

Input Format

- The first line of input will contain a single integer N denoting the number of elements in A.
- the second line contains N space-separated integers denoting elements of the array A.

Output Format

Output a single integer, sum of all the elements in the array A.

Sample 1:

Input

8 2 4 1 4

19

Ou

tpu

t 5

Task 2:

Now complete the following program:

```
#include <bits/stdc++.h>
using namespace std;

int main() {
    // your code goes here

}
```

Problem: Find maximum in an Array

Given a list of N integers, representing height of mountains. Find the height of the tallest mountain.

Input:

- First line will contain *T*, number of testcases. Then the testcases follow.
- The first line in each testcase contains one integer, N.
- The following line contains N space separated integers: the height of each mountains.

Output:

For each testcase, output one line with one integer: the height of the tallest mountain for that test case.

Constraints

- $1 \leq T \leq 10$
- $1 \leq N \leq 100000$
- $0 \leq$ height of each mountain $\leq 10^9$

Task 3:

Now complete the following program:

```
#include <bits/stdc++.h>
using namespace std;

int main() {
    // your code goes here

}
```

Problem: MIN To MAX

You are given an array **A** of size **N**.

Let **M** be the minimum value present in the array initially.

In one operation, you can choose an element $\mathbf{A_i}$ ($1 \le i \le N$) and an integer **X** ($1 \le X \le 100$), and set $\mathbf{A_i}$ **= X**. Determine the minimum number of operations required to make **M** the maximum value in the array **A**.

Input Format

- The first line of input will contain a single integer **T**, denoting the number of test cases.
- Each test case consists of multiple lines of input:
    - The first line of each test case contains a single integer **N** - the size of the array.
    - The next line of each test case contains **N** space-separated integers $\mathbf{A_1}$, $\mathbf{A_2}$, …, $\mathbf{A_n}$ - the elements of the array.

Output Format

For each test case, output on a new line, the minimum number of operations required to make **M** the maximum value in the array **A**.

Constraints

- $1 \leq \mathbf{T} \leq 100$
- $1 \leq \mathbf{N} \leq 100$
- $1 \leq \mathbf{A_i} \leq 100$

Samp

le 1

Input

:

3

2

1 2

4

2 2 3 4

1

1

Output:

1

2

0

Explanation:

Test case 1:

The minimum value in the array, **M**, is initially **1**. We can use one operation as follows:

- Choose $\mathbf{A_2}$ and set it as **X = 1**. Thus, the final array becomes **[1, 1]**.
- Since all elements of the final array are **1**, the maximum value of the array is now **1**.
- It can be shown that this is the minimum number of operations required.

Test case 2:

The minimum value in the array, **M**, is initially **2**. We can use two operations as follows:

- Choose $\mathbf{A_4}$ and set it as **X = 2**. The array becomes **[2, 2, 3, 2]**.
- Choose $\mathbf{A_3}$ and set it as **X = 2**. The array becomes **[2, 2, 2, 2]**.

- Since all elements of the final array are **2**, the maximum value of the array is now **2**.

Test case 3:

The minimum value in the array, **M**, is initially **1**. It is also the maximum value of the array. Hence, no operations are required.

Task 4:

Now complete the following program:

```
#include <bits/stdc++.h>
using namespace std;

int main() {
    int t;
    cin>>t;
    while(t-
    -){
        int n;
        cin>>n;
        int
        a[n];
        for(int i=0;i<n;i++){
            cin>>a[i];
        }
        // your code goes here
    }

}
```

Pseudocode: Grade School Integer Multiplication

```
Algorithm GradeSchoolMultiplication(A, B)
    Input: Two integers A and B (both non-negative)
    Output: The product of A and B

    1. Convert A and B into digit arrays (rightmost digit is at index 0)
    2. Let Result = 0
    3. Let PlaceValue = 1   // Tracks place value (1, 10, 100, ...)

    4. For each digit b in B (starting from rightmost digit):
        5. Let PartialSum = 0
        6. Let Carry = 0

        7. For each digit a in A (starting from rightmost digit):
            8. Multiply a and b → Product = (a * b) + Carry
            9. Extract last digit of Product → Digit = Product mod 10
           10. Update Carry = Product ÷ 10
           11. Append Digit to PartialSum (shift left)

        12. If Carry > 0, append Carry to PartialSum
        13. Multiply PartialSum by PlaceValue and add it to Result
        14. Update PlaceValue = PlaceValue × 10 (shift left)

    15. Return Result
```

Explanation

- We multiply each digit of **B** by every digit of **A**, like in long multiplication.
- Each partial result is adjusted for its place value and added to the final result.
- Carry is handled at every step to ensure correct arithmetic.
- The final sum gives the correct product.

Task 5:

Now complete the following program:

```
#include <iostream>
#include <cstring> // For memset
using namespace std;

#define MAX 200 // Maximum digits to handle large numbers

class BigIntMultiplication {
private:
    int numA[MAX], numB[MAX], result[MAX];
    int lenA, lenB;

public:
    // Constructor to initialize arrays
    BigIntMultiplication() {
        memset(numA, 0, sizeof(numA));
        memset(numB, 0, sizeof(numB));
        memset(result, 0, sizeof(result));
        lenA = lenB = 0;
    }

    // Function to convert an integer into a digit array (least
significant digit first)
    void storeNumber(int num, int arr[], int &length) {
        while (num > 0) {
            arr[length++] = _________;  // store
            digit num /= 10;
        }
    }

    // Function to multiply two integers using grade school
multiplication
    void multiply(int A, int B)
        { if (A == 0 || B == 0)
        {
            cout << "0" << endl; // If either number is 0, the
product is 0
            return;
        }
```

```
        // Store numbers in digit arrays
        storeNumber(A, numA, lenA);
        storeNumber(B, numB, lenB);

        // Perform grade school multiplication
        for (int i = 0; i < lenB; i++) {
            // Perform grade school multiplication
        }

        printResult();
    }

    // Function to print the final result
    void printResult() {
        int lenResult = lenA + lenB;
        while (lenResult > 1 && result[lenResult - 1] ==
            0) { lenResult--; // Remove leading zeros
        }

        // Print the result in correct order (reverse of storage)
        for (int i = lenResult - 1; i >= 0; i--) {
            cout << result[i];
        }
        cout << endl;
    }
};

// Driver
Code int
main() {
    int A, B;
    cout << "Enter two integers: ";
    cin >> A >> B;

    BigIntMultiplication
    multiplier; cout << "Product:
    "; multiplier.multiply(A, B);

    return 0;
}
```