**Bonus Question – Algorithm + Test Scenarios**

Write a program that takes in a non-empty list of non-empty sorted arrays of integers and returns a merged list of all of those arrays. The integers in the merged list should be sorted order.

* Algorithm

Here's a step-by-step algorithm for the program:

**Input:**

* Ask the user to enter the number of arrays.
* Initialize arrays to store the input arrays and arraySizes to store their sizes.

**For each array:**

* Ask the user to enter the size of the array (a positive integer).
  + If the input is not a positive integer, display an error message and ask for input again.
* Allocate memory for the array based on the given size.
  + If memory allocation fails, display an error message and exit.
* Ask the user to enter the elements of the array in sorted order.
* Repeat these steps for all input arrays.

**Merge the arrays:**

* Initialize a merged array to the first input array.
* Initialize a variable mergedSize to the size of the first array.
* For each of the remaining arrays:
  + Use the **mergeSortedArrays** function to merge the current merged array with the next input array.
  + Update the mergedSize to reflect the new size of the merged array.
  + Free the memory allocated for the current input array.

**Output:**

* Print the merged sorted array.
* Free the allocated memory for all arrays.

Here's the **pseudocode** for the algorithm:

Input: Number of arrays (numArrays)

For each array (i from 1 to numArrays):

Repeat until valid size (arraySizes[i]):

Ask for the size of array i

If size is not a positive integer:

Display error message

Allocate memory for array i of size arraySizes[i]

If memory allocation fails:

Display error message and exit

Ask for the elements of array i in sorted order

Initialize merged array to the first input array

Initialize mergedSize to the size of the first array

For each remaining array (i from 2 to numArrays):

merged = mergeSortedArrays(merged, mergedSize, array[i], arraySizes[i])

mergedSize += arraySizes[i]

Free memory for array[i]

Print "Merged Sorted Array:"

For each element in merged array:

Print the element

Free memory for all arrays

Exit

The algorithm ensures that the program accepts user input, merges the sorted arrays, and then prints the merged result while handling errors gracefully.

**Test Scenarios with Test Cases –**

**Scenario 1: Basic Test**

* Test Case 1:
  + Input:

Enter the number of arrays: 2

Enter the size of array 1: 4

Enter the elements of array 1 in sorted order: 1 5 9 21

Enter the size of array 2: 3

Enter the elements of array 2 in sorted order: -5 0 12

* + Expected Output

Merged Sorted Array: -5 0 1 5 9 12 21

**Scenario 2 : Empty Array**

* Test Case 1:
  + Input:

Enter the number of arrays: 3

Enter the size of array 1: 0

Enter the size of array 2: 2

Enter the elements of array 2 in sorted order: 4 7

Enter the size of array 3: 0

* + Expected Output

Merged Sorted Array:

**Scenario 3: Negative Numbers**

* Test Case 1:
  + Input:

Enter the number of arrays: 2

Enter the size of array 1: 3

Enter the elements of array 1 in sorted order: -10 -5 0

Enter the size of array 2: 4

Enter the elements of array 2 in sorted order: -20 -15 -8 -1

* + Expected Output

Merged Sorted Array: -20 -15 -10 -8 -5 -1 0

**Scenario 4 : Large Array**

* Test Case 1:
  + Input:

Enter the number of arrays: 2

Enter the size of array 1: 1000

Enter the elements of array 1 in sorted order: <1000 integers>

Enter the size of array 2: 500

Enter the elements of array 2 in sorted order: <500 integers>

* + Expected Output

Merged Sorted Array:<1500 integers in sorted order>

**Scenario 5 : Single Array**

* Test Case 1:
  + Input:

Enter the number of arrays: 1

Enter the size of array 1: 5

Enter the elements of array 1 in sorted order: 1 5 9 21 30

* + Expected Output

Merged Sorted Array: 1 5 9 21 30

In conclusion, we've developed a responsive and robust C program that efficiently merges multiple sorted arrays into a single sorted array. This program demonstrates the principles of dynamic memory allocation, user interaction, error handling, and algorithmic efficiency. By following best practices in C programming, we've created a reliable tool for merging and sorting data.

Remember, this program is not only a practical solution for merging sorted arrays but also serves as an example of how to design interactive and error-tolerant software. We hope this presentation has provided valuable insights into both the programming aspects and the importance of good software design.

Thank you for your attention, and if you have any questions or need further clarification, please feel free to ask