**Note—**  you can use the Arrays.sort() method to sort arrays.

**Question 5:- void sort(int x[]);**

**Answer :-** In this program:

1. We create an integer array arr with some sample values.
2. The Arrays.sort(arr) method sorts the array in ascending order.
3. Finally, we print the sorted array.

The output will be:

The original array is: 5 -2 23 7 87 -42 509

The sorted array is: -42 -2 5 7 23 87 509

**Question 6:- void sort(int x[][]);**

**Answer :-** In this program:

Create a 2D array Iniialization

Initialise 2D array *int*[][] arr2D = {

                {13, 7, 6},

                {45, 21, 9},

                {101, 102}

            };

Print original 2D array

For loop

 System.out.println("\n\nOriginal 2D array:");

            for (*int*[] row : arr2D) {

                System.out.println(Arrays.toString(row));

            }

Sort array

// Sort each row individually

            for (*int*[] row : arr2D) {

                Arrays.sort(row);

            }

            System.out.println("\nSorted 2D array:");

            for (*int*[] row : arr2D) {

                System.out.println(Arrays.toString(row));

            }

OUTPUT

Original 2D array:

[13, 7, 6]

[45, 21, 9]

[101, 102]

Sorted 2D array:

[6, 7, 13]

[9, 21, 45]

[101, 102]

**Question 7:-** void mergesort(int x[],int y[]); using java example program

# Answer :-  the ****Merge** Sort** algorithm. Merge Sort is a divide-and-conquer algorithm that efficiently sorts an array by dividing it into smaller subarrays, sorting each subarray, and then merging them back together.

Output

Given Array: 12 11 13 5 6 7

Sorted array: 5 6 7 11 12 13

**Complexity:**

* Time Complexity: O(n log n)
* Auxiliary Space: O(n)

**Advantages of Merge Sort:**

1. **Stability**: Merge sort maintains the relative order of equal elements in the input array.
2. **Guaranteed worst-case performance**: It performs well even on large datasets.
3. **Parallelizable**: Merge sort can be easily parallelized to take advantage of multiple processors or threads.

**Question 8:-** void union(int x[],int y[]);

# Answer :-  create a method called unionArrays that takes two integer arrays as input and returns their union:

This represents the union of the arrays A and B, removing any duplicate elements. Feel free to modify the input arrays as needed!

**Question 9:-** void intersection(int x[],int y[]);

create a method called intersection that takes two integer arrays x and y as input and returns an array containing their common values.

Answer :-

# We create a new array result to store the common values.

# We iterate through the elements of array x.

# For each element in x, we check if it exists in array y.

# If it does, we add it to the result array.

# Finally, we resize the result array to the actual number of common values found.

**Question 10:-** void MatrixMult(int x[][],int y[][]);

# Answer :- multiply two matrices using **multidimensional arrays**. We’ll assume that the input matrices x and y are both square matrices (i.e., the number of rows equals the number of columns).

* We define two matrices x and y.
* We check if the matrices can be multiplied (i.e., the number of columns in x equals the number of rows in y).
* We initialize a result matrix to store the product.
* We perform matrix multiplication using nested loops.
* Finally, we print the resultant matrix.
* **Question 11:-** void intersection(int x[],int y[]);

Answer :-

**Question 12:-** void MatrixTranspose(int x[][]);

# Answer :- transpose a matrix using a 2D array. The goal is to convert rows into columns and columns into rows.

* We define a 2D array m representing the original matrix.
* The transpose method calculates the transposed matrix by swapping rows and columns.
* The printMatrix method displays the matrix.

**Question 14:-** void Triangle a sum

# Answer :- prints a triangle using a 2D array of characters. We’ll assume that the input int represents the height of the triangle.

**Question 144:-** void Triangle a sum

Answer /\*

A sum triangle is a pattern where each level contains the sum of consecutive two elements from the previous level. Here’s how we can do it:

 \*/

# Question 15:- void intersection(int x[],int y[]);

* Answer :- We create a 2D array called triangle to store the characters.
* We initialize the array with spaces.
* We then fill in the appropriate positions with asterisks to form the triangle.
* Finally, we print the triangle to the console.

When you run this program, it will produce the following output:

\*

\*\*\*

\*\*\*\*\*

\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*

**Question 20:-** void Shift(int x[]);

# Answer :-

# **Using a Loop:** You can shift elements in an array by moving each element one position to the right (or left).