Q1. Write a function *maxInEachRow* that takes a 2D array of integers of dimensions *4x4* and returns a 1D array containing the maximum element from each row. In the main function, print the returned array to check the maximum values from each row. For example, for array int array[4][4] = { {3, 5, 7, 2}, {1, 6, 4, 8}, {9, 2, 3, 5}, {4, 8, 1, 7} }, the function should return 1d array consisting of 7 8 9 8 values.

Q2. Create a function swapElements that takes two pointers to integers as parameters and swaps their values. In the main function, declare an array of integers, then call swapElements to swap two elements within the array using pointers. Print the array before and after the swap to verify the changes.

Q3. Write a function concatenateArrays that takes two 1D arrays of integers. The function should return a new array that concatenates the two arrays. For example, if the arrays are {1, 2, 3} and {4, 5, 6}, the result should be {1, 2, 3, 4, 5, 6}. In the main function, print the concatenated array to verify correctness.

Q4. Write a recursive function `count7()' that given a non-negative integer n, returns the count of the occurrences of 7 as a digit, so for example count7(717) yields 2. Following are some more examples count7(7170123) should return 2, count7(7) should return 1, count7(123) should returns 0 Hint: Note that mod (%) by 10 yields the rightmost digit (126%10 is 6), while divide (/) by 10 removes the rightmost digit (126/10 is 12).

Q5. Write a recursive function towerOfHanoi() that solves the Tower of Hanoi problem for n disks. The function should print the moves required to transfer n disks from source peg to destination peg using an auxiliary peg.

**Explanation:**

**Base Case:** If there is only one disk (n == 1), the disk is moved directly from the source peg to the destination peg. This is the simplest operation.

**Recursive Case:**

Move the top n-1 disks from the source peg to the auxiliary peg, using the destination peg as a temporary storage.

Move the nth disk (the largest disk) directly from the source peg to the destination peg.

Move the n-1 disks from the auxiliary peg to the destination peg, using the source peg as the temporary storage.

**Function Call:**

The recursive function towerOfHanoi(n, source, auxiliary, destination) is called with the correct arguments to simulate moving the disks according to the rules of the Tower of Hanoi puzzle.

**Example Output:**

For n = 3, the output will be:

Enter the number of disks: 3

Move disk 1 from A to C

Move disk 2 from A to B

Move disk 1 from C to B

Move disk 3 from A to C

Move disk 1 from B to A

Move disk 2 from B to C

Move disk 1 from A to C