Recursive Functions Lab 20-3-2023

1.	Write a C++ Program which inputs a sorted array and tells whether the key searched is present in array or not using Binary Search Algorithm **recursively**.
2.	You have a list arr of all integers in the range [1, n] sorted in a strictly increasing order. Apply the following algorithm **recursively** on arr: • Starting from left to right, remove the first number and every other number afterward until you reach the end of the list. • Repeat the previous step again, but this time from right to left, remove the rightmost number and every other number from the remaining numbers. • Keep repeating the steps again, alternating left to right and right to left, until a single number remains. Given the integer n, return *the last number that remains in* arr. **Input:** n = 9 **Output:** 6 **Explanation:** arr = [1, 2, 3, 4, 5, 6, 7, 8, 9] arr = [2, 4, 6, 8] arr = [2, 6] arr = [6]
3.	**Recursive function** to find common elements in three sorted array? (solution) Now we are coming on territory of tough array questions. Given three arrays sorted in non-decreasing order, print all common elements in these arrays. Examples: input1 = {1, 5, 10, 20, 40, 80} input2 = {6, 7, 20, 80, 100} input3 = {3, 4, 15, 20, 30, 70, 80, 120} Output: 20, 80
4.	Given an array **A** of size **N**. **Find the minimum number of operations needed to convert the given array to 'Palindromic Array'**. The only allowed operation is that you can merge two adjacent elements in the array and replace them with their sum. **Example:** Input array: 5 3 3 3 No of minimum operation: 3 (don't worry, you will know the reason soon!!) Input array: 5 3 3 5 No of minimum operation: 0 (It's palindromic array!!)
5.	Suppose A, B, C are arrays of integers of size M, N, and M + N respectively. The numbers in array A appear in ascending order while the numbers in array B appear in descending order. Write a user defined **recursive** function to produce third array C by merging arrays A and B in ascending order with unique elements. Use A, B and C as arguments in the function. **Output:** Enter number of elements of first array: 5 Enter element in ascending order: 10 26 30 40 45 Enter number of elements of second array: 4 Enter element in descending order 44 40 34 26 The Merged Array in ascending Order: 10 26 30 34 40 44 45

6.	Given two sorted arrays of integers, **recursively** find a maximum sum path involving elements of both arrays whose sum is maximum. We can start from either array, but we can switch between arrays only through its common elements. For example, Input: X = { 3, 6, 7, 8, 10, 12, 15, 18, 100 } Y = { 1, 2, 3, 5, 7, 9, 10, 11, 15, 16, 18, 25, 50 } The maximum sum path is: 1 —> 2 —> 3 —> 6 —> 7 —> 9 —> 10 —> 12 —> 15 —> 16 —> 18 —> 100 The maximum sum is 199		
7.	Check if a given set of moves is circular or not. A move is circular if its starting and ending coordinates are the same. The moves can contain instructions to move one unit in the same direction (M), to change direction to the left of current direction (L), and to change direction to the right of current direction (R). Assume that the initial direction is North. For example, Set of moves MRMRMRM is circular Set of moves MRMLMRMRMMRMM is circular Algorithm – start with (0, 0) as the starting coordinates and North as the starting direction and linearly read each instruction from the input string. For every instruction, update the coordinates of the current location (x, y) if the instruction is MOVE or update the current direction if the instruction is GO LEFT or GO RIGHT. The move is circular if we are back to the starting coordinates (0, 0) in the end.		
8.	Write a C program to find reverse of any number using recursion.		
9.	Write a C program to check whether a number is palindrome or not using recursion.		
10.	Write a recursive function which prints sum of factors of a given number n as shown below		
	Input n = 4 output 1 + 1 + 1 + 1 1 + 1 + 2 1 + 3 2 + 2 4	Input n = 6 output 1 + 1 + 1 + 1 + 1 + 1 1 + 1 + 1 + 1 + 2 1 + 1 + 1 + 3 1 + 1 + 2 + 2 1 + 1 + 4 1 + 2 + 3 1 + 5 2 + 2 + 2 2 + 4 3 + 3 6	Input n = 8 output 1 + 1 + 1 + 1 + 1 + 1 + 1 + 1 1 + 1 + 1 + 1 + 1 + 1 + 2 1 + 1 + 1 + 1 + 1 + 3 1 + 1 + 1 + 1 + 2 + 2 1 + 1 + 1 + 1 + 4 1 + 1 + 1 + 2 + 3 1 + 1 + 1 + 5 1 + 1 + 2 + 2 + 2 1 + 1 + 2 + 4 1 + 1 + 3 + 3 1 + 1 + 6 1 + 2 + 2 + 3 1 + 2 + 5 1 + 3 + 4 1 + 7 2 + 2 + 2 + 2 2 + 2 + 4 2 + 3 + 3 2 + 6 3 + 5 4 + 4 8