Topic: You can do it!

Task1: Write a program that will take positive integer values as input and store them in an array. It will stop taking input for any negative number or zero. The program will print how many times each of the input numbers occurs. (Don't show anything for the number that didn't occur.)

Sample Input: 3 4 5 7 8 9 2 3 3 4 4 5 6 0	Sample Output: 3 occurs:3 times 4 occurs: 3 times 5 occurs: 2 times 7 occurs: 1 time 8 occurs: 1 time 9 occurs: 1 time 2 occur: 1 time

[**Note**: The output is sorted based on the occurrence of each number]

Task 2: Write a program where the main function will take an integer number as input. The program will also contain another function that will recursively calculate the factorial of the number and return to the main function.

Sample Input: 5 Sample Input: 10	Sample Output: 120 Sample Input: 3628800

Task 3: The binary search algorithm is an algorithm that is based on compare and split mechanism. This algorithm is also known as half-interval search, logarithmic search or binary chop. The objective is to search for the position of the target value in a sorted array. It compares the target value with the middle element of the array. If the element is equal to the target element, then the algorithm returns the index of the found element. And if they are not equal, the searching algorithm uses a half section of that array, based on the comparison value, the algorithm uses either the first half (when the value is less than the middle) or the second half (when the value is greater than the middle). And does the same for the next half.

Your task is to implement this algorithm in both iterative and recursive fashion.

Sample Input: 0 2 6 11 12 18 34 45 55 99 -1 55	Sample Output: 8
Sample Input: 25 33 37 43 55 60 -1 42	Sample Output: -1
Sample Input: 2 3 7 13 15 20 25 33 -1 -1	Sample Output: -1

[**Note**:

- The input array is given as sorted sequence of numbers.
- **Using iterations**− this means using a loop inside the function that checks for the equality of the middle element.
- **Using recursion**- In this method, the function calls itself again and again with a different set of values.
- Array indexing starts from 0.
- It will stop taking input for any negative number.]